Askia Training

Askiascript 2.0

Introductory training

Course 150

Participant's Coursebook

With Tutor's Notes

Contents

Introduction

Format

This course comprises five flexible modular sessions, which permit different learning pathways through the training course. The course has been designed for several alternative delivery methods, to offer maximum flexibility to learners:

- As a taught course lasting a little under one day
- For online delivery as a series of webinar tutorials
- For one-to-one delivery (e.g. peer-to-peer training)

In each case, the course is delivered as a series of separate modules or sessions. Each session is intended to last no more than an hour, though it may take longer, if additional time is required to complete the practical work.

Each session follows the same format:

1. Introduction (by tutor) 2-3 minutes
2. Tutorial and demonstration 15-20 minutes
3. Summary (by tutor) 2 minutes
4. Practical exercises variable
5. Recap, feedback and questions

Audience and course pre-requisites

This course is primarily intended for experienced Askia scriptwriters and programmers who should already be familiar with the Askia software and have a working knowledge of all the material covered in the Askia Design course (Course 100).

Module topics

Session 150-1	Askiascript 2.0 Basics: Objects and Operators
Session 150-2	Logical structures
Session 150-3	Variables and keywords
Session 150-4	Advanced loops

Related course

In addition, the single session short course Data Editing (Course 410), combines learning from both this course and also the Introduction to Askia Tools (Course 400). For audiences already familiar with the content of the Tools course, it may be useful to continue immediately on to Course 410 after completing session 150-4 in this course.

Recommended learning pathways

All the modules of this course are considered to be 'core modules' and should be followed in sequence.

Face-to-face delivery

If delivered face-to-face, the course may either be delivered as a short one-day course, or spread out over two half days, as follows:

One day plan	*Two day plan*
Morning	**Day** 1 (morning or afternoon)
Module 1	Module 1
Module 2	Module 2
Afternoon	**Day 2** (morning or afternoon)
Module 3	Module 3
Module 4	Module 4

Online delivery

The recommended method for online delivery is that a tutor delivers the content in the context of a series of webinar sessions. Participants then complete the practical exercises at their own pace in time for the next webinar session.

Webinars could be spaced out over a period of time, e.g. at the same time each day over a period of five days, with the first two sessions covered on day one, and a final follow-up session after presentation of all the new on day five. Other configurations are also possible, e.g. three sessions per day, to achieve completion of the course over days.

After the first module, each webinar will begin with a review of the practical work and an opportunity for questions and answers (Q&A) from participants.

The webinar should be delivered using an appropriate screensharing or web-conferencing tool (e.g. GoToMeeting or Webex) with video sharing and audio conference. Ideally, the tutor should be able to allow participants to share their screen too, e.g. when reviewing exercises completed by them.

Group sizes should be restricted to six participants, so that the tutor is able to review each participants work and provide specific feedback.

Webinar 1	Introduction Presentation of module 1
Webinar 2	Review exercise from module 1; Q&A Presentation of module 2

Webinar 3	Review exercise from module 2; Q&A Presentation of module 3
Webinar 4	Review exercise from module 3 Questions and answers Presentation of module 4
Webinar 5	Review exercise from module 4 Questions and answers Further learning using the Knowledge Base

Below are some possible timetables for delivery:

Two day plan	*Three day plan*	*Four day plan*
Day 1	**Day 1**	**Day 1**
Webinar 1	Webinar 1	Webinar 1
Webinar 2	Webinar 2	Webinar 2
	Day 2	**Day 2**
Day 2	Webinar 3	Webinar 3
Webinar 3	Webinar 4	
Webinar 4		**Day 3**
Webinar 5	**Day 3**	Webinar 4
	Webinar 5	
		Day 4
		Webinar 5
		Day 5
		Webinar 6

Preparation

When preparing for this course, you will need to agree the content you will be presenting with your client contact.

If possible, provide the training materials in advance so your client can prepare copies for each participant.

Course agenda and timetable

Once you have agreed the timetable or programme with your client contact, prepare a course agenda showing the start and end times of each session, as well as any breaks, for the modes that you will be covering.

You should add these timings to the introductory PowerPoint, as well as providing it by email or on paper. You should modify one of the four suggested learning pathways (tracks 1-4). Remove the other tracks not being followed from the PowerPoint, to avoid confusion.

While updating the PowerPoint, modify the title page of the slides to show your name and, if possible, the client company's logo.

Prepare a one-sheet hand-out for the participants that give them the course timings – or email this to them. It is a good idea to have some printed copies. You may also have other specific items that you wish to add to this sheet or provide as separate sheets – e.g. guidance on their company-specific

example project (see below) to be used in the exercises, info on how to access the system, and so on.

Example questionnaires

This course uses the example Power Tools survey.

Documents and files provided

The example questionnaires are provided as three files, as follows:

- **Power Tools survey.docx** Word file of the Power Tools survey, also included at the end of this booklet. We recommend printing this out for reference during the course.
- **PowerTools_with_hh_loop_WITH_routings,qex**
- **PowerTools_without_hh_loop_WITH_routings.qex** Askia file for the tutor only, containing worked examples of scripts
- **Model answers.zip** QEX files for the tutor only, containing model answers for the exercises.

Using examples from the example power tools QEX

This booklet describes many examples using the power tools QEX. In several of these cases, you will need to create a new script. To save a great deal of time during the course, we recommend that you keep a copy of this document open on your computer, so that you copy and paste the scripts directly into the example QEX.

Using the model answers for the participant exercises

For your reference, the file **model answers.zip** contains several QEX files which have suggested solutions to the exercises. These are not intended to be given to the students, but are provided for your reference during the exercises, for example when you are assisting students who have questions about the problems posed in the exercises.

Working with a client-provided surveys or sample files

It can also be useful to work with some sample files and survey files provided by the client. Generally, we recommend that these should be relatively short surveys, so that participants can complete test interviews relatively quickly.

As a preliminary step to this activity, you (as tutor) will need to review the questionnaire and identify any topics that will require additional training or explanation. Generally, these should take place *after* this introductory training course has been delivered and the exercises completed, in order to ensure that training participants receive a consistent, controlled training experience.

Preparing for face-to-face delivery

Setting up the room

Arrive early. Ensure you arrive to give the training in plenty of time, so that you have time to set up the room as you would like it. Make it as tidy as possible – it is distracting to be learning in a room full of clutter.

Arrange the seating. Arrange all of the seats so that your group have a clear view of the screen and also that you have a clear view of them too – so you have facial contact with each of them. Set out enough seats for the group, and put away, or move to the back or against the wall, seats you don't want to be used, so it is obvious where you want everyone to sit.

Flipchart or whiteboard. Ideally, make sure you have a flipchart or whiteboard that you can also write up important messages or notes, and the right kind of pens (indelible pens for paper; dry-wipe pens for the whiteboard).

Heat. Make sure the room is **warm**, but not too warm, and that your group will be comfortable.

Noise. Ensure the door can be closed, so that the group will not be distracted by external **noise**, and also that you won't be distracting others in the office with your training session.

Light. Adjust the **lighting** – ensure the screen is bright and visible but not dazzled from overhead lights or daylight. Don't over-dim the room: you also need to be visible to the group and you need to be able to see them.

Course materials. Set out the coursebooks for everyone – and if possible, provide them with an Askia pen to use too (or at least, pens from the stationery cupboard). Also lay out the course timetable/agenda you have created, or any other supplementary sheets you have prepared (e.g. on the company-specific example project).

Demo screen and system access. Ensure you know how to work the data projector or large screen, that you have access to the system, to the Askia software and to the directories containing the training files. After any testing, close any apps so you are starting from a 'cold start' in your training session.

PowerPoint. Finally, set the display screen to show the first slide of the course PowerPoint set – in full screen mode – so that this is on the screen when participants enter for their training.

Introductory presentation

Before you start the main training you should:

Do introductions of everyone in the group. If time permits, also:

- Ask each to say what they do
- Ask each what experience they have in (a) Askia and (b) analysis and reporting of market research surveys
- Ask each to say what they hope to get out of the course

Explain the structure of the course – use the first three slides in the PowerPoint presentation *Course100.ppt.*

Explain the timings: start times and end times – and when break times will occur

Explain the materials and resources they will be using:

- A participant's coursebook each
- Access to AskiaDesign
- Access to the training files

If necessary, explain where they will be able to find the files

Check that everyone has the materials and resources they need

Ask if there are any questions

Continue on to Session 101

Checklists

This check-list summarises the points described above. Please refer to the more detailed descriptions above, for more information.

During preparation

- ☐ Discuss course timings and participants with client and agree schedule
- ☐ Update agenda and timings in training course PowerPoint
- ☐ Discuss the example projects with client (you can forward the Word files) and review whether additional time should be allowed after the course to work on a "real" project.
- ☐ Obtain the example project and work through which questions and examples will be used in each exercise (see content checklist, above)
- ☐ If necessary, where no QEX file exists, import the survey and/or program those parts needed for the exercises to work.
- ☐ Agree who will print and bind the coursebooks for participants (double-sided and bound or in a folder is the preferred presentation)
- ☐ Ensure that there are printed copies of the Mobile Internet survey, for use by participants during the exercises.
- ☐ If you are covering askia**word**, ensure that participants will have access to the askia**word** version of the Mobile Internet survey, with the questions and responses already marked up, so that they can open it quickly in Word.
- ☐ Prepare and print any supplementary notes – agenda, example project info, etc.
- ☐ Sort out access to the system, server etc at the client company – will you be using their PC or yours?
- ☐ Establish who will be able to update the IE settings on participant's PCs to access askiavista.
- ☐ Check the training room will have a large screen or projector you can use; Internet access for you to use; whiteboard or flipchart and pens
- ☐ Check there will be one PC for each participant or each pair of participants

- ☐ If you are not familiar with the system configuration and how Askia is deployed, check with the Askia team member responsible for the set-up, or with your client. Verify, in particular, where/how askiavista is deployed, and if projects are stored in SQL.
- ☐ Check any other arrangements with your client contact immediately before your visit – explain that you would like to have access to the room at least 30 minutes before your first session begins, in order to get it ready.

Using an alternative project for participant exercises

If the students are going to be using an alternative to the Mobile Internet survey for the exercises (e.g. the client's own project), it should include **at least** the following items:

- ☐ Two single-coded questions.
- ☐ Two multi-coded questions, preferably with at least one including an other (specify) type response.
- ☐ An open text question.
- ☐ An open numeric question.
- ☐ A grid.
- ☐ Chapters (if not present in study provided, you can add these in as appropriate).
- ☐ Demographic questions.
- ☐ A structure where respondents might skip at least one question, depending on earlier answer/s given.
- ☐ Also ensure that:
- ☐ you have a version of this study in Microsoft Word format;
- ☐ you have marked it up in askia**word**, leaving a few questions for the participants to mark up for themselves;
- ☐ participants will have access to an electronic copy of the marked-up file, so that they can open it in Word;
- ☐ you have printed copies to give to the students for use during the exercises.

On the day

- ☐ Arrive early
- ☐ Arrange the seating and tidy away unwanted seats, clutter etc – make the room as tidy as possible
- ☐ Adjust the lighting and temperature as appropriate
- ☐ Check for noise or other distractions; check you will be able to keep the door closed
- ☐ Set out coursebooks, agendas and pens for all participants
- ☐ Check you have access to the software, the Internet, the example projects you need and any passwords required
- ☐ Check you have the latest version of askia installed on your PC and on the participants' PCs (see Quick installation, below)

- ☐ Check whiteboard or flipchart – and that you have the right pens for each
- ☐ Ensure you have a glass or bottle of water to hand
- ☐ Ensure you know how to reach your client contact from the training room, in case of any problem
- ☐ Check PowerPoint slides load and have been edited for this client (and not for another client!)
- ☐ Finally, have PowerPoint open in presentation mode. This should be open, showing the welcome screen, when participants arrive

Software update, if required

The best way to ensure you have the latest Askia version of software, including the latest templates is to download and install the latest version. This can be found at http://installers.askia.com/helpdesk/askiasuite/. When installing the new version, make sure you uninstall the previous version of AskiaSuite (through control panel) before performing the installation.

Preparing for online delivery

Preparation

Trial run. If you are not familiar with the screen sharing or conference software you are using, do a trial run first with a colleague, to check that you understand the controls, how to show your screen, how to pass control to another participant etc.

Ensuring good quality audio. If you are using the audio provided with the online conference tool, you must use a USB headset. This will eliminate most ambient noise from being broadcast, which is distracting for participants. Use a quiet place from which to deliver your webinar – a meeting room or private room rather than an open-plan office.

Sharing your desktop. Close other applications and preferably switch off alerts (e.g. email or calendar alerts) that will interrupt your broadcast. Pay attention to what you may be displaying when you share your desktop, and ensure that any confidential material is not in sight.

Send invites and reminders. Ideally send meeting invites as calendar invites inviting an acknowledgement that they will be attending. Ensure this contains the instructions for accessing the webinar session and also the audio. You may need to offer telephone as well as web audio access. It is also wise to send an email reminder to participants on the day of the session.

Participant's materials. Send the materials for participants in advance, as a PDF, and encourage them to print these out so they have the hard copy to refer to during the webinar session, as it contains details of the exercises.

Availability of the software. Verify with each participant that they have access to the AskiaDesign software in advance of the training. You can cover this again at the end of the first webinar session. However, the first session does not require that they have access to askiascript or askiadesign on their own machines. It is only necessary when they start the first exercise.

Delivery

Open early and start promptly. It's good practice to start the call ten minutes early, to allow people to assemble. Present a "welcome" holding screen, with a message to confirm the webinar, and say that the session will be starting shortly. Aim to start each session punctually – otherwise if you start each session late, waiting for the last person to arrive, it will only encourage latecomers to each successive session.

Encourage questions to be put by chat. You can choose whether to place your participants on mute or to allow them to speak over the audio feed at any time. For larger groups, it is better to place everyone on mute – and it may be better to do that with small groups too, if there is background noise breaking in from any one participant. You should suggest that people put their questions via the chat facility anyway. That way, you will have a record of the questions, and can address them when appropriate. You can also choose when you break off from the material you are presenting to deal with questions.

During preparation

- ☐ Structure the delivery so that participants will have time to complete the exercise between each webinar session. Consider the time-zones of all participants.
- ☐ Update agenda and timings in training course PowerPoint file – and show the times in the attendee's time zones.
- ☐ Identify the roles and responsibilities of those attending: are they CAI scriptwriters, programmers, DP or survey technicians? You may wish to prepare a short pre-joining questionnaire by email to collect this information.
- ☐ Distribute course materials with explanatory instructions.
- ☐ Prepare any supplementary notes – agenda, example project info, etc.
- ☐ Email all participants to confirm the times of the course presentations, the materials they will need to print out, the files they will need to download and information on the software they will need to have installed on their PC; also explain to the participants which version of the software they need to use and how to install it (see ***Error! Reference source not ound.*** on p. **Error! Bookmark not defined.**).
- ☐ Familiarise yourself with the webinar delivery platform, and perform a test run (dress rehearsal) to check the presentation of the slides, your screen and the audio feed, with a colleague in a different location.
- ☐ Send out the webinar joining instructions several days before the course.
- ☐ Send out a reminder the day before the webinar.
- ☐ Check you have the latest version of askia installed on your PC.

On the day

- ☐ Be ready early: at least one hour before the webinar, to perform these checks.
- ☐ Check you have all of the files you will need to demonstrate: example QEX or QES files, PowerPoint, Word files etc.

- ☐ Check that your PC is only displaying the information you wish participants to see, and that any client confidential or other material is closed and away from view. Close your email client and any web pages not related to your presentation.
- ☐ Open any applications you intend to use during your presentation, such as AskiaDesign, MS Excel, MS Word etc. Minimise these ready for your start.
- ☐ Check that your desktop is neutral.
- ☐ Ensure you have a glass or bottle of water to hand
- ☐ Check for noise or other distractions; check you will not be interrupted or disturbed – if necessary, display a "Do not Disturb" sign to alert colleagues that you are conducting a **live web broadcast**.
- ☐ Try to turn off any alerts that will interrupt the screen while you are sharing your screen.
- ☐ Turn off your mobile phone completely.
- ☐ Around 10 minutes before the start of the session, open the Webinar, open PowerPoint and display a holding slide that announces the name of the Webinar session, and a welcome text that explains that "this webinar will start shortly".
- ☐ Start on time. Introduce yourself and if necessary, introduce who else is participating in the session.
- ☐ Explain to the group how you want questions to be handled.

Session 150-1 Askiascript 2.0 Basics: Objects and Operators

Outline

Topics presented

In this session, we will introduce you to:

- The concept of an askiascript
- Objects and Properties
- Useful operators

Learning outcomes

At the end of this session you will understand:

- How askiascript interacts with the different Askia modules
- How and where to add askiascript definitions
- How to use questions as objects in your scripts
- Which properties to use when referring to question objects
- How to define and use operators within scripts

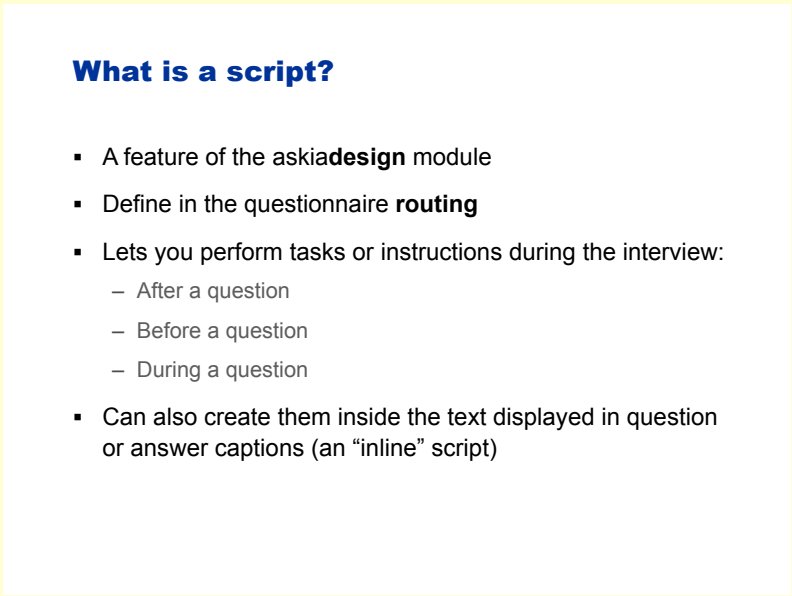
Tutorial

What is a script?

Askiascript is an extension to askiadesign that allows you to program Askia at the time it is executing a questionnaire, so that you can enhance how questionnaires are delivered or how they collect data, and you can use them to automate tasks which may otherwise require manual intervention to achieve.

Askiascript consists of an easy-to-learn set of syntax commands and keywords which interact with the normal variables or questions that you create in askiadesign. They allow you to perform tasks before or after questions are displayed, or at the beginning or the end of an interview. There are many different applications for scripts, and many different occasions when they can be useful.

→ Show PowerPoint Slide "What is a script?"



→ Explain that askiascript contains hundreds of keywords, properties and methods, and we cannot cover all of them today. However, complete documentation, with all keywords, and examples for each of them, is available in:

- the Knowledge Base (at the following link): https://support.askia.com/hc/en-us/articles/200003251-AskiaScript-2-0-specification
  - Demonstrate how to navigate to the above Knowledge Base page (e.g. go to the Knowledge Base and search for askiascript).
- and the askiadesign Assistant.

Notes for participants

You can define scripts in the askia**design** module, in the questionnaire routing. Your scripts can perform tasks or instructions during the interview, either before, during or after a question. You can also create scripts “inline” inside the text displayed in question or answer captions.

Scripts defined in the routing

→ *Show PowerPoint Slide “Scripts defined in the routing”*

Scripts defined in the routing

Two places where you can define a script in the routing:

- To create a condition
  - Used to make a routing decision
  - These scripts must end up returning a *true* or a *false* result, to control whether the routing is followed or skipped
- To create a value
  - Often used to set the value of a variable
  - It must return the right kind of value, e.g. *set*, *numeric* or *string*

*NB. A special **Edit** version of routing scripts (covered in 150-5) lets you edit data after the interview is complete, to correct the data*

→ *Show PowerPoint Slide “Example scripts for conditions or values”*

Example scripts for conditions or values

Scripts for values

These scripts need to deliver a number, a text response (called a “string”) or set of codes (a “set”), according to the context

```
Lastname.Value
```
A text *string*

```
Q1.Answers.Value
```
A *set* of codes

Scripts for conditions

Scripts that create conditions must result in a *true* or *false* value (called a “boolean” result)

```
Q1.Answers[1].Value = 3
```
T or F (*boolean*)

[n] lets us refer to a specific position in the set

→ Explain the difference between the part of the routing script that defines the value and the part which contains the condition

→ Demo opening *PowerTools_without_hh_loop_WITH_routings.qex*.

- Switch to routing mode
- From the tree view on the left, select S3
- Click **Ins**
- Write a script here

→ Show the scripts that exist already (e.g. the script for the routings Q1) – and explain that these are the same as the advanced conditions covered in the Design course – the "ignore responses" script – and which they may also have defined in any projects they have worked on since.

Notes for participants

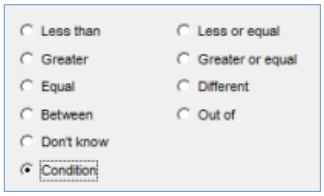
You can create a script at two places in a routing instruction:

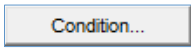
- To make a routing position; these scripts must end up returning a *true* or a *false* result, to control whether the routing is followed or skipped.
- To create a value; this is often to set the contents of a variable. It must return the right kind of value (e.g. numeric or string) to match the variable whose contents it is setting.

Note: *The above script types are executed during the interview. A special* ***edit*** *version of routing scripts (covered in 150-5) lets you edit data after the interview is complete, to correct the data.*

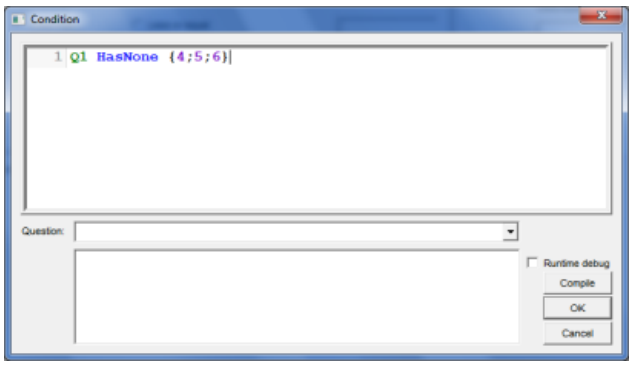
To create a script in the routing:

1. In the toolbar, click **routing mode**, if you are not already in this mode.
2. Select **condition**:





3. Click the **condition…** to open the script editor.
4. In the script editor, enter *your script*:



5. Click **OK**.

Unconditional scripts

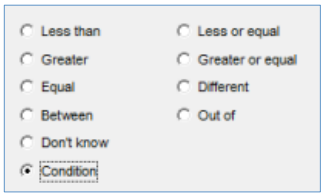
Routing scripts that you want to execute on every occasion should be given the condition *true*, so that no logical test is applied.

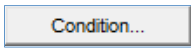
→ Show how to write a true condition into the condition field for a routing.

Notes for participants

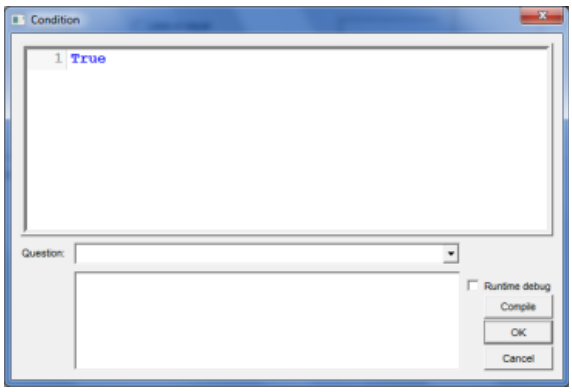
To create an unconditional script:

1. In the toolbar, click **routing mode**, if you are not already in this mode.
2. Select **condition**:





3. Click the **condition…** to open the script editor.
4. In the script editor, enter *True*:



5. Click **OK**.
6. In **action**, select **set value**.
7. In **target question**, select the question whose value you want to set in your script.
8. Next to **value if true**, click … to open the script editor.
9. Enter your script.
10. Click **OK**.

Questions as Objects

Objects, properties and methods

Questions, and their properties and methods.

→ Show PowerPoint Slide “Objects, Properties and Methods”

→ Explain that any question can be treated as an Object in Askia

- Objects, properties and methods are widely used today in programming
- Properties return information about different aspects of an object.
- Methods allow you to modify the result given when interrogating the property of any object

→ Point out how to reference a question with its shortcut.

- Explain that shortcuts containing spaces should be surrounded by carets (^). For example: ^Q2. Age^.

Notes for participants

In askiascript, any question can be handled as an object. You can always reference a question with its shortcut, for example:

Q1
Shortcuts containing spaces should be surrounded by carets (^). For example:

^Q2. Age^.
Objects have properties and methods:

- Object **properties** return information about different aspects of an object (e.g. the question type);
- Object **methods** allow you to modify the result when interrogating the property of an object.

Properties and methods are referenced by using the question name, then a dot and the property or method name. For example:

Q1.Value

Working with Values

Value property

→ *Show PowerPoint Slide "Value property"*

- Value is the property you will use most when writing script
- In some contexts, value is implicit
- At other times, you will need to write .value,
  - e.g. when adding one or more methods to the value property

→ Explain the examples given:

- LastName value is implicit – it would return the name. Other examples: Gender.Value might return 1, City.Value might return 'Paris')
- LongCaption and ShortCaption are properties that return the question's long caption and short caption respectively.
- LongCaption and ShortCaption are properties that relate to the Design definition and not to the response

- LastName.Value.ToUpperCase() – here you must specify *Value*, as it cannot be implicit if you add a method. The method always comes after the property.

Values: a reminder

→ *Show PowerPoint Slide "Values: a reminder"*

There are several different kinds response values in Askia:

- **Numeric**: *numbers*
- **String**: *open text – letters, numbers, punctuation, spaces*
- **Sets**: *coded answers to single and multiple questions*
- **Date**, **Time**, and **Date and Time**
- **Boolean:** logical values *true* or *false*

→ Explain that you must always match the right type of value with the operation you are performing, the variable you are referring to, or to which you are assigning a value.

Notes for participants

Value is the property you will use most often when writing scripts. In some cases, value is implicit. At other times, you will need to write .value (e.g. when adding one or more methods to the value property).

For example:

Gender.Value
will return 1 if the question contains the value 1 in the current interview.

- **LongCaption** and **ShortCaption** are properties that return the question's long caption and short caption respectively. They return information from the question definition, as set up in askiadesign, not the respondent's answer. For example:
  - `Q3.LongCaption()`
  - `Q3.Shortcaption ()`
- `LastName.Value.ToUpperCase()` – here you must specify Value, as it cannot be implicit if you add a method. The method always comes after the property. For example:
  - `Q3.Value.ToLowerCase() = "bosch"`
  - `Q3.Value.ToUpperCase() = "BOSCH"`

There are several different kinds response values in Askia:

- **Numeric**: *numbers*
- **String**: *open text – letters, numbers, punctuation, spaces*
- **Sets**: *coded answers to single and multiple questions*
- **Date**, **Time**, and **Date and Time**
- **Boolean:** logical values *true* or *false*

You must always match the right type of value with the operation you are performing, the variable you are referring to, or to which you are assigning a

value. If you do not, the script will not execute, and a warning message will appear.

Sets and arrays

The responses to single and multiple questions are different from numeric or text questions in that they consist of a series of answer possibilities, and in the case of multiple questions, can also contain more than one answer after the question has been asked.

→ Refer participants to the example Power Tools survey.

→ *Show PowerPoint Slide "Sets and arrays" – this animates the selection of question 1 and shows the responses being stored as an array*

■ Notes for participants

The responses to single and multiple questions are different from numeric or text questions in that they consist of a series of answer possibilities. In the case of multiple questions, they can also contain more than one answer after the question has been asked.

Response data for multiple questions can be referred to in scripts as **sets**. This consists of the response numbers, separated by semi-colons, within curly brackets. For example, if a respondent selected responses 1, 3 and 5, this would be referred to as:

{1; 3; 5}

Askiascript provides functions for comparing two or more sets, and for manipulating the values in a set. For details, please *see* 3. Comparing sets of values on page 28 and Working with sets and arrays on page 48.

Response properties

Responses, and their methods and properties.

→ *Show PowerPoint Slide "Response properties for questions with sets of answers"*

→ Explain that responses are also objects

→ These special properties allow you to access much more information about single and multiple questions:

- **Responses** (returns the entire list of responses for a closed question);
- **AvailableResponses** (returns the list of available responses in the order of display);
- **Answers** (returns the list of responses selected by the respondent, in the order selected).

→ *Show PowerPoint Slide "Response properties: examples"*

→ As an example, in the Power Tools project, switch to question mode and show the question Q1. Explain that if *Lawn edger* and *Chain saw* and were selected by the respondent in that order, the various properties would return the following:

- `Q1.Responses.Index` will return 1;2;3;4;5;6;7;8;9
- `Q1.AvailableResponses.Index` will return 3;1;4;2;8;6;9;7;5 (assuming this was the randomised order in which they were presented to the respondent).
- `Q1.Answers.Index` will return 6;4
- `Q1.Answers.EntryCode` will return 21; 3 (because the entry code for *Lawn edger* is 21)
- `Q1.Answers.Caption` will return Lawn edger;Chain saw
- `Q1.Answers[2].Index` will return 4 (the second answer given)

→ Explain that by using the square brackets `[]`, you can refer to a specific item in an array. This can be used for `Responses`, `AvailableResponses`, `Answers`, etc.

→ `Q1.Value` will return 6;4 (Q1 will also return 6;4 – as we mentioned earlier, the `Value` property is implicit).

Notes for participants

Like questions, responses are objects, and have various properties and methods.

The following special properties allow you to access much more information about single and multiple questions:

- **Responses** (returns the entire list of responses for a closed question);
- **AvailableResponses** (returns the list of available responses in the order of display);
- **Answers** (returns the list of responses selected by the respondent, in the order selected).

For example, if the responses 6 and 4 (“Lawn edger” and “Chain saw”) are selected at Q1, these script snippets will give the following results:

Script	Result	Notes
`Q1.Responses.Index`	1;2;3;4;5;6;7;8;9	This is all of the responses defined in the question, no matter what the respondent selected.
`Q1.AvailableResponses.Index`	3;1;4;2;8;6;9;7;5	Assuming the response order is randomised, and this was the order they were presented to the current respondent.
`Q1.Answers.Value`	6;4	The responses selected by

Script	Result	Notes
		the respondent.
`Q1.Answers.EntryCode`	21; 4	If the entry code for response 6 is 21.
`Q1.Answers.Caption`	Lawn edger; Chain saw	
`Q1.Answers[2].Index`	4	The second answer selected by the respondent. By using square brackets, you can refer to a specific item in an array. This can be used for `Responses`, `AvailableResponses`, `Answers`, etc.
`Q1.Value`	6;4	
`Q1`	6;4	This is equivalent to `Q1.Value`. The `Value` property is implicit.

Operators

1. Comparing numeric values

Operators that allow you to compare numeric values.

→ *Show PowerPoint Slide "1. Comparing numeric values"*

→ Explain each of the operators in the following list. For the purposes of the explanation, create a new routing, as follows.

- Use Age as the start question.
- Select **condition**, and click **condition…** to edit the logical expression. For example, enter `If Age>24` to check that respondent is over 24.
- When you have explained/demonstrated these operators, you can delete the new routing, as it is used only for this explanation.

■ Notes for participants

The following operators allow you to compare numeric values:

Equal to (=)

True if the first value is equal to the second value. For example:

```
Age = 24
```

is true if the value in Age is 24.

Not equal to (<>)

True if the first value is not equal to the second value.

```
Age <> 24
```

is true if the value in Age is *not* 24.

Greater than (>)	True if the first value is greater than the second value. `Age > 24` returns is true if the value in Age is greater than 24.
Greater than or equal to (>=)	True if the first value is greater than or equal to the second value. `Age >= 24` is true if the value in Age is greater than or equal to 24.
Less than (<)	True if the first value is less than the second value. `Age < 24` returns true if the value in Age is less than 24.
Less than or equal to (<=)	True if the first value is less than or equal to the second value. `Age <= 24` is true if the value in Age is less than or equal to 24.

2. Comparing string values

Operators that allow you to compare string values, typically in open-ended questions.

→ *Show PowerPoint Slide "2. Comparing string values"*

Typically used on text questions

Operators

- Equal
- Not equal

Methods

- Trim()
- Left()
- ToLowerCase()
- ToUpperCase()
- IsNumber()

String comparisons are case sensitive!

→ *Then show the slide "String comparisons are case sensitive!"*

→ Explain that string comparisons are **case sensitive**. You can, however, make it case insensitive by using the `ToLowerCase()` or `ToUpperCase()` methods. For example:

```
o  Q3.Value.ToLowerCase() = "bosch"

o  Q3.Value.ToUpperCase() = "BOSCH"
```

- `Q3.Value.Left(3) = "bos"`
- `Q3.Value.IsNumber() = False`
- `Q3.Value.Trim() = "bosch"`

"= equal to" and "<> not equal to" string comparisons

→ *Show the slides "= Equal to" and "<> Not equal to".*

→ Add a new routing along the following lines:

```
If (Q1.Value.Trim().ToLowerCase() = "string trimmer")
Then
  Return True
Else
  Return False
EndIf
```

→ Explain the operators in the table below (under participant notes)

→ Delete the new routing, as it is used only for this example.

→

Useful object methods for strings

An **Object Method** provides something that you can perform on an object, such as to provide a value or perform a test. Object methods are always specific to the type of object being referenced. Here we describe several methods that are useful when working with string questions.

→ Show PowerPoint Slide "Other useful object methods for strings (1)"

→ Explain the following methods:

- Left()
- Right()
- IsNumber()

Notes for participants

The following methods are useful when working with string questions:

Equal to (=)	True if the first string is equal to the second string. For example: Q1.Value.Trim().ToLowerCase() = “string trimmer” is true if the string in Q6 is “bosch” and false if the string in Q6 is “black & decker”.
Not equal to	<> is true if the first string is not equal to the second string. Q1.Value.Trim().ToLowerCase() <> “string trimmer” is true if the string in Q6 is not “bosch”.
Left()	Returns the specified number of characters from the left of the string. For example: `Q6.Value.Left(3)` Returns “bos” if the string in Q6 is “bosch”
Right()	Returns the specified number of characters from the left of the string. For example: `Q6.Value.Right(2)` Returns “ch” if the string in Q6 is “bosch”
IsNumber()	True if the string contains a pure number; false if it does not. Q6.Value.IsNumber() Is true if the string in Q6 is “5”.

Other useful object methods for strings

→ *Show PowerPoint* Slide *“Other useful object methods for strings (2)”*

→ Explain that you can use regular expressions with the `IsMatch` method.

→ Explain that this course does not teach regular expressions. If nobody in the group is familiar with regular expressions, you can omit this section, or explain it very briefly.

Notes for participants

If you are familiar with regular expressions, you can use them in your scripts to perform complex comparisons. *Note that regular expressions are not part of the scope of this course.*

Two examples

1. To check the format of an email address:

```
Dim email = "askia@askia.com"
email.IsMatch("\w+@\w+\.[a-z]+")
```

Note: there is built-in method for this so `email.IsEmail()` will produce the same result

2. This script will check if a string is a proper UK number:

```
Dim rgUkPhone = "(\+|00)\s*\d{2}(\s?\d{3}){2}\s?\d{4}"
("0044 207 689 5492").IsMatch(rgUkPhone) ' => true
```

3. Comparing sets of values

→ *Show PowerPoint Slide "3. Comparing and testing sets of values"*

→ *Then the five slides, one for each operation: "Has", HasNone", "HasAll" and "HasAndNoOther", "HasAllAndNoOther or ="*

Operators that compare specific groups of items in a routing condition. Explain the following.

→ There is a group of operators that is useful for referring to multi-coded questions in a routing condition. Equals (=) would not work with a multi, because more than one response might be selected.

→ We want our routing instruction to return 1 (true) when the set of answers has been selected, and 0 (false) when it has not been.

→ Explain that to refer to a set of value(s) we use curly brackets {} and separate the values using semi colons ";" (e.g. {1;2;5;8}).

→ We can also use the "`To`" keyword to specify ranges of numbers, so {1 to 5;7;9 to 12;15} is equivalent to {1;2;3;4;5;7;9;10;11;12;15}.

→ To demonstrate, create a routing to demonstrate each of the operators in the following list. Explain the logic used, and run the questionnaire to show the effect of the logic.

- Use Q1 as the start question.
- Set the **routing action** as *go to* and the **target question** as *{end}*.
- Select **condition**, and click the **condition…** button to define the logic.
- When you have explained/demonstrated these operators, you can delete the new routing, as it is used only for this explanation.

■ Notes for participants

Some operators allow you to compare specific groups of items in multi-coded questions. A simple equals comparison (e.g. `Q1 = 1`) will not work with a multi, because more than one response might be selected.

To refer to a set of values we use curly brackets {} and separate the values inside with semi colons. For example:

```
{1;2;5;8}
```

We can use the "To" keyword to specify ranges of numbers:

```
{1 to 5;7;9 to 12;15}
```

is equivalent to

```
{1;2;3;4;5;7;9;10;11;12;15}
```

The operators are as follows:

Has	Checks that **at least one** of the referenced response items was selected. If this is the case, the function returns 1; otherwise, it returns 0. `Q1 Has {4;5;6}` will return 1 if either the fourth, fifth or sixth were selected.
HasNone	The opposite of "has". It returns 1 if **none** of the specified response items were selected by the respondent. `Q1 HasNone {4;5;6}` will return 1 if none of the fourth, fifth and sixth responses were selected.
HasAll	Checks that **all** of the referenced response items were selected. Note that other responses may be selected too, and the function will still return 1. For example: `Q1 HasAll {4;5;6}` will return 1 if the respondent answered 4, 5 and 6.
HasAndNoOther	Checks that **at least one** of the specified responses was selected by the respondent. However, if any responses not in the list were selected, then the function returns 0. For example: `Q1 HasAndNoOther {4;5;6}` returns 1 if the respondent selected response 4. If she selected responses 1 and 5, the function will return 0 (because 1 is not in the list).
HasAllAndNoOther or =	Similar to "`HasAll`", but it returns 0 if the respondent selected any responses other than those in the list. In other words, the selected response must match the list exactly. So: `Q1 HasAllAndNoOther {4;5;6}` will return 1 if the respondent answered 4, 5 and 6, but 0 if she answered 1,4,5 and 6 (because 1 is not in the list of qualified values). Note that the above example is equivalent to `Q1 = {4;5;6}`.

Recap

→ Provide a one-minute summary of the topics covered and then prepare participants for the exercise.

→ *Show PowerPoint Slide "Summary"*

In this session, we have:

- Looked at the question and response objects.
- Seen some of the operator comparisons that can be made in your scripts.

- Looked at some of the useful functions for examining string variables
- Seen how regular expressions can be combined with askiascript expressions.

Practical exercises

→ Ask participants to complete these exercises. They can start with a blank questionnaire.

→ If time is short, and you want the participants to complete these exercises more quickly than normal, direct them to the `Trim` and `ToUpperCase` functions for exercise 2. The exact keywords needed are:
Step 3: `RespName.Value.Trim()`
Step 6: `RespName.Value.ToUpperCase().Trim()`

Exercise 1: Using script to set a variable

Follow these steps:

1. Open askia**design**. A new, blank QEX will be created.
2. Create a **Gender** question (closed) with the responses Male and Female.
3. Create an **Age** question (numeric).
4. Create an **AgeGroup** question (closed) with the responses *Younger than 25, 25 to 34* and *35 or older* as the answers.
5. Now define routing instructions to set the appropriate category in the **Age Group** question, according to the value in **Age**. Do this by creating three separate routing instructions, each one checking for a specific age range in **Age**, and setting the value in **AgeGroup** accordingly. So, your first routing will set **AgeGroup** to 1, where appropriate, the second will set the value to 2, and so on.

   Note: Later in the course, you will learn how to do this with only one routing condition, but for now create three separate routing instructions.

Exercise 2: Editing your script

Follow these steps:

1. Add a **RespName** question (type: open), and position it at the start of your QEX.

   Note that we should not call a question "Name", as this is a reserved word in Askia, so we should use an alternative (in this case, "RespName").
2. Add a second question **RespNameTrimmed** (type: open) and place it after *RespName*. Ensure that the question's **visible during data entry** property is *not* selected, so that it is hidden during interviewing.
3. Add a routing as follows, to remove any spaces from the start and end of the name given at *RespName*, and store that result in *RespNameTrimmed*.

When defining your routing, click **condition…**. Then, simply enter **true**. This ensures that the condition is always executed. Click **OK**.

Set up the other settings of the condition as follows:
Start question: RespName

Action: set value

Target question: *RespNameTrimmed*

Value if true: *write an appropriate expression here*

4. Later in the QEX, add a routing to display the contents of *RespNameTrimmed*:

   Condition: true (ensures the routing is always run)

   Action: Show message

   Message text: *Name is ??RespNameTrimmed??.*

5. Test your questionnaire. You should see the contents of the *RespNameTrimmed* variable displayed by your second routing.
6. Now go back into your first routing, and change it to convert the answer to capital letters.
7. Test the questionnaire, to see the results of your change.
8. If time permits, try to repeat steps 6 and 7 two more times, using different string handling operations each time, as suggested by the routing editor.

Session 150-2 Logical structures

Outline

Topics presented

In this session, we will introduce you to:

- **If/Then/Else** structures
- The **Return** statement
- Comments in script

Learning outcomes

At the end of this session you will be able to:

- Create complex logical structures using If/Then/Else
- Write more script that is more efficient to write and easier to understand

Tutorial

If ...Then ... Else structures

Using If/Then/Else structures

→ Show PowerPoint Slide "If/Then/Else Syntax"

→ Explain the purpose of If/Then/Else structures:

- they allow you to write clearer logical statements;
- they allow you to write complex conditions more easily.
- For example, you might want to execute certain code only if a specific combination of answers has been given by the respondent, and otherwise execute another set of code. This can be done clearly and easily with an If/Then/Else structure.

■ Notes for participants

`If/Then/Else` structures allow you to execute blocks (sections) of script according to specific conditions. For example, you might want to execute a block of code only if a specific combination of answers has been given by the respondent, and otherwise execute another set of code.

By using `If/Then/Else` structures, you can write more complex logical statements, and your logical statements will be easier to understand when you look at the script.

Syntax

The syntax is as follows:

```
If logical_expression Then
      'block of script to execute if logical_expression
is True
EndIf

Or
If logical_expression Then
      'block of script to execute if logical_expression
is True
Else
      'block of script to execute if logical_expression
is False
EndIf
```

Note: *Else* is always optional; you do not have to include it in an If structure.

Return statement

→ *Show PowerPoint Slide "Return"*

→ When a `Return` expression is reached, it immediately breaks the script flow (all following lines are ignored), and returns the expression associated.

Notes for participants

When a `Return` expression is reached, it immediately breaks the script flow (all following lines are ignored), and returns the expression associated. For examples:

```
Return
Return True
Return 1
```

Worked examples of *If … Then … Else*

If … Endif

→ *Explain that if the condition is satisfied, any code inside the block is executed. In this example, the return statement causes the result of the script to be the value 1; return statements also cause the script to stop executing immediately.*

If with *else*

→ Explain the concept of `Else`, and modify your condition as follows:

```
If(Q1 has {6}) Then

  Return 1

Else

  Return 0

EndIf
```

→ Explain that `Else` is optional; you do not have to include it in an `If` structure.

→ Remove this new routing now, as it is only used to explain this concept.

Example for participants

```
If Q1 Has {4;5;6} Then
      '4, 5 or 6 was selected at Q1
      Return True
Else
      'none of these responses were selected at Q1
      Return False
EndIf
```

If with *elseif*

→ Explain `ElseIf`, and show the **value if true** in the routing for **Age**:

```
If (Age >= 18 And Age <= 25) Then
  Return 1
ElseIf (Age >= 26 And Age <= 35) Then
  Return 2
ElseIf (Age >= 36 And Age <= 45) Then
  Return 3
ElseIf (Age >= 46 And Age <= 55) Then
  Return 4
ElseIf (Age >= 56) Then
  Return 5
EndIf
```

→ Explain that `ElseIf` is optional; you do not have to include it in an If structure.

→ Explain that you have as many `ElseIf` clauses as you want in a structure

→ Explain that if you use `ElseIf` and `Else` together, the `Else` clause must be the last one in the structure.

→ Change the example to use `Else` at the end:

```
ElseIf (Age >= 56) Then
  Return 5

Else
  Return 5
```

■ Note and example for participants

`ElseIf` can be used in an `If/Then/Else` block in order to create logical expressions with multiple outcomes. For example:

```
If (Age >= 18 And Age <= 25) Then
  Return 1
ElseIf (Age >= 26 And Age <= 35) Then
  Return 2
ElseIf (Age >= 36 And Age <= 45) Then
  Return 3
ElseIf (Age >= 46 And Age <= 55) Then
  Return 4
ElseIf (Age >= 56) Then
  Return 5
EndIf
```

Nesting *If* within *If*

`IF` structures can be nested inside each other.

→ Create this routing as an example, and explain that you are using comments to make it easier to understand

```
If (Q1 Has {1}) Then

      If (Age >=56) Then
        Return True
      EndIf

ElseIf (Q1 Has {2}) Then

      If (Age >=56) Then
        Return True
      EndIf

EndIf
```

■ Note and example for participants

`If` structures can be nested inside each other to create complex logical expressions. For example:

```
If (Q1 Has {1}) Then
      '1 is selected at Q1

      If (Age >=56) Then
            '1 is selected at Q1 and Age>=56
            Return True
      EndIf

ElseIf (Q1 Has {2}) Then
      '2 (but not 1) is selected at Q1

      If (Age >=56) Then
            '2 (but not 1) is selected at Q1 and Age>=56
            Return True
      EndIf

EndIf
```

Comments

A single quotation mark or apostrophe (') introduces a comment.

→ Explain that comments should be used widely in scripts to make the meaning clearer to anyone who has to review or maintain the script in future.

→ Amend the previous example to incorporate comments:

```
If (Q1 Has {1}) Then
  'Push Lawn mower owners

  If (Age >=56) Then
    'Push Lawn mower owners aged 56 and over
    Return True
  EndIf
```

```
ElseIf (Q1 Has {2}) Then
  'Riding Lawn mower owners

  If (Age >=56) Then
    'Riding Lawn mower owners aged 56 and over
    Return True
  EndIf

EndIf
```

■ Notes for participants

You can include comments in your scripts to make them easier for you or others to understand. A single quotation mark or apostrophe (') introduces a comment. The rest of the line is not executed. You should use comments widely in order to make the meaning of the script to anyone who has to review or maintain the script in future. For example:

```
'this script returns True if 4, 5 or 6 is selected at Q1
```

Throughout the syntax explanations and code examples in this booklet, you will see comments that help explain the code being described.

Using *Has*, *HasAll* and *HasNone* in If blocks

Conditions referring to single or multiple questions are best used with these operators because they are dealing with sets of answers.

→ Step through the following examples:

```
If (Q1 Has {1 to 5}) Then
  'User selected at least one of the first 5 responses
  Return 1
EndIf

If (Q1 Has {1 to 95}) Then
   'User selected at least one of the first 95 responses
  Return 2
EndIf
```

→ Explain the following additional examples:

```
  If (Q1 HasAll {3;4;5;6}) Then
  If (Q1 HasNone {8;9}) Then
```

■ Notes for participants

Conditions that refer to responses to a single or multiple questions, are best used with these operators because they all deal with sets of answers.

Examples:

```
If (Q1 Has {1 to 5}) Then
```

```
      'User selected at least one of the first 5
responses
      Return 1
EndIf

If (Q1 Has {1 to 95}) Then
      'User selected at least one of the first 95
responses
      Return 2
EndIf

If (Q1 HasAll {3;4;5;6}) Then
      'User selected responses 3, 4, 5 and 6
      Return True
EndIf

If (Q1 HasNone {8;9}) Then
      'User didn't select responses 8 or 9
      Return 1
EndIf
```

→ Delete the routing or routings you just created.

Expressions

Brackets in expressions

→ *Return to the PowerPoint presentation here, and show the slide "Brackets in expressions"*

→ Explain that curly brackets {} are used to enclose response values, for example ranges like {1 to 5}.

→ Explain that normal brackets are also used as part of askiascript syntax, for example `Q1.Value.Trim()`.

→ Explain that normal brackets can also be used to improve the clarity of logical expressions (for example to surround `IF` conditions), but this is not mandatory.

■ Notes for participants

Curly brackets `{}` are used to enclose response values, for example ranges. Examples:

```
{3;4;5;6}
{1 to 5}
{1 to 6;8;9}
```

Note that normal brackets are also used as part of Askia Script syntax, for example `Q1.Value.Trim()`.

Normal brackets can also be used to improve the clarity of logical expressions (for example to surround IF conditions), but this is not mandatory. For example:

```
((Gender Has {1}) And (Q1 Has {9})) Or ((Gender Has {2})
And (Q1 Has {9}))
```

Logical expressions: using AND, OR and NOT

→ Note that there is no slide for this concept: use the QEX again to demonstrate, as described below.

→ Explain that (`AND`, and `OR`) determine how multiple expressions are combined.

→ **`AND`**: the condition is *true* if both expressions are true.

→ Show the routing based on the question Age:

```
If (Age >= 18 And Age <= 25) Then
```

This is true if the value in Age is equal to or greater than 18 **AND** it is less than or equal to 25.

→ **`OR`**: the condition is *true* if either expression is true.

→ Create a new routing based on Q1. Use the following syntax:

```
Q1 has {1} OR Q1 has {2}
```

This is true if the respondent selected either response 1 or response 2 at Q1.

→ Explain that the **`NOT`** operator allows us to reverse an expression, so the condition will evaluate true if the expression is false, and vice versa.

→ Amend our new routing so that is says:

```
NOT(Q1 has {1} OR Q1 has {2})
```

`The expression is reversed so that` it is true if neither response 1 nor 2 was selected at Q1.

→ Delete the routing or routings you just created.

→ Explain that some more advanced expressions will be introduced in subsequent sessions.

Notes for participants

AND, OR and NOT determine how multiple expressions are combined.

AND The condition is *true* if **both** expressions are true. For example:

```
If (Age >= 18 And Age <= 25) Then
```

This is true if the value in Age is equal to or greater than 18 **AND** it is less than or equal to 25.

OR The condition is *true* if **either** expression is true. For example:

```
Q1 has {1} OR Q1 has {2}
```

This is true if the respondent selected either response 1 or response 2 at Q1.

NOT Allows you to reverse an expression, so the condition will evaluate true if the expression is false, and vice versa. For example:

```
NOT(Q1 has {1} OR Q1 has {2})
```

The expression is reversed so that it is true if neither response 1 nor 2 was selected at Q1.

Note: more advanced expressions are covered in the subsequent sessions of this course.

Return with logical or Boolean values

→ Create a new routing to demonstrate `Return`, with the **start question** being *Gender*.

→ When teaching this section, copy and paste the script from this Word file, to save time.

→ Use the following script:

```
If ((Gender Has {1}) And (Q1 Has {9})) Then
  'Male sander owners
  Return True  ' Immediately return True
ElseIf ((Gender Has {1}) And (Q1 Has {8})) Then
  'Male drill owners
  Return False ' Immediately return False
EndIf
```

→ Delete the routing you just created.

Inline script

Inline script allows you to include askiascript code snippets in other contexts, such as question captions, response captions or online interview screen definitions.

→ *Return to the PowerPoint slides here, and show the slide "Inline script"*

You will already know this syntax for text substitution in a question or answer caption:

```
??Q1??
```

This will display the answers given to Q1

You can also write script to create the value to display directly within a question or answer caption:

```
{% If condition Then %}
  'Text alternative 1 to display
{% Else %}
   'Text alternative 2 to display
{% EndIf %}
```

→ Explain that you can also use the askiascript engine using {% askiascript %} to place askiascript code into these controls.

→ Explain how to use {% askiascript %} to execute the script without output and use {%= askiascript %} to execute the script and write the result.

→ You can demonstrate this within a screen:

→ Set up a screen for Q1, and add a label to it.

→ Open the label's **properties** page.

→ Select **caption update during entry**.

→ In the **caption** box, paste the following code:

```
{% If (Q1 HasAllAndNoOther {1;2}) Then %}
   'Has both types of mower, but no other tools
{% ElseIf (Q1 HasAndNoOther {1;2}) Then %}
  'Has only a mower (of either type, or both)
{% ElseIf (Q1 HasAll {1;2}) Then %}
   'Has both types of mower and may have other tools
{% ElseIf (Q1 Has {1;2}) Then %}
   'Has a mower (or either type, or both) and may have
other tools
{%ElseIf (Q1 HasNone {1;2}) Then %}
   'Does not own either type of mower
{% EndIf %}
```

→ Test the project *IN CATI TEST MODE*!!!. The contents of the box will change, according to the response given to Q1, so you can quickly demo the effect of the script.

→ Test the project. **You must do this in CATI Test Mode**, as **caption update during entry** only works in this mode.

The contents of the box will change, according to the response given to Q1, so you can quickly demo the effect of the script.

Notes for participants

You can also use the askiascript engine using {% askiascript %} to place askiascript code into a question or answer caption.

You can use `{% askiascript %}` to execute a script without output and `{%= askiascript %}` to execute a script and write the result.

For example:

```
{% If (Q1 HasAllAndNoOther {1;2}) Then %}
      Has both types of mower, but no other tools
{% ElseIf (Q1 HasAndNoOther {1;2}) Then %}
      Has only a mower (or either type, or both)
{% ElseIf (Q1 HasAll {1;2}) Then %}
      Has both types of mower and may have other tools
{% ElseIf (Q1 Has {1;2}) Then %}
'Has a mower (or either type, or both) and may have other
tools
{%ElseIf (Q1 HasNone {1;2}) Then %}
      Does not own either type of mower
{% EndIf %}
```

The contents of the caption will change according to the response given to Q1.

Recap

→ Provide a one-minute summary of the topics covered and then prepare participants for the exercise.

→ *Show PowerPoint Slide "Summary"*

In this session, we have:

- Created control structures using **If/Then/Else**
- Looked at the syntax required to define conditions
- Introduced the **Return** statement
- Added comments to script
- Inserted askiascript snippets into text as "inline script"

Practical exercise

→ Ask participants to complete this exercise

Writing control structures

Follow these steps:

Follow these steps:

1. Delete the routings you created in the first exercise.

2. Create just one routing to set the value of the **Age Group** variable, according to the value in **Age**. You will need to use an `If/Then/Else` structure.
3. Now put another control structure around the `If/Then/Else`, to test that **Age** contains a valid response between 16 and 99, and only recode the variable when it is.
4. Create a **Return** which will be True if **Age** was coded into the new variable (i.e. it was in the range 16..99) and False if it was not. Think about the best way to create that control structure.
5. Add the following question to your survey:

   Shortcut: Q1

   Long caption: Which of the following financial products do you have?

   Type: closed multi

   Responses: Investment plan, Life insurance, Mortgage, Pension plan
6. Add the following question:

   Shortcut: Q2

   Long caption: Which of the following financial products are you considering buying in the future?

   Type: closed multi

   Responses: Life insurance, Pension plan, None of these
7. Add a new routing containing a script that means that Q2 is asked only if either *life insurance* or *pension plan* were selected at Q1, but not both. It should also be asked if neither response was selected at Q1.
8. It is especially important with scripting to carry out thorough testing, to ensure that your scripts have the effect you want. Test your script to ensure that is working, and amend it if necessary.
9. Add an inline script into Q2's long caption, so that the caption varies according to the number of responses selected at Q1 from *Life insurance* and *Pension plan*:
   - **If one response is being presented**: Are you considering buying the following financial product in the future?
   - **If both responses are being presented**: Are you considering buying either of the following financial products in the future?
10. Now go through the script you have created and add comments, to make it more understandable to anyone else who has to read it.
11. *Optional*: If time permits, try to experiment by creating another version of this routing using different logical options, e.g. reverse the logic using *Not*, or using `And` and `Or`. See if you can improve on your first version.

Session 150-3 Variables and keywords

Outline

Topics presented

In this session, we will introduce you to:

- Virtual variables
- **Intersection** and **Union** set functions
- **Count** property
- The **SelectRandom** function

Learning outcomes

At the end of this session you will know:

- How to create new array variables in your script
- How to perform logical tests and combinations on arrays
- How to detect the number of elements in an array
- How to use scripts to make random selections from questions

Tutorial

Virtual variables

→ Explain the following

Virtual variables (also known as "declarative" variables in some programming languages) allow you to create additional variables for use inside the script which only exist within the script:

→ *Show PowerPoint Slide "Virtual variables"*

Virtual variables

- Virtual variables allow you to create additional variables for use inside the script which only exist within the script
- Known in some programming languages as *declarative variables*
- They are temporary variables and the value assigned to them is not stored in the data record
- Limited scope
  - Because their "scope" is limited to the current script, you can have another virtual variable with the same name in a different script and it will be a different virtual variable
  - For inline scripts, the scope is the entire caption containing the script

Creating a virtual variable

→ *Show PowerPoint Slide "Creating a virtual variable"*

Creating a virtual variable

Use **Dim** to declare the variable and optionally give it an initial value:

```
Dim variable_name
Dim variable_name = 1
```

Names for virtual variables must…

- start with a letter or "_"
- be composed of letters, numbers or "_" and no spaces
- not be the same as a reserved keyword or question shortcut

Assigning values and performing comparisons

→ *Show PowerPoint Slide "Comparison versus Assignment"*

Comparison versus Assignment

You can assign a value to a virtual variable using "=":

```
LoopCount = 0
```
Set LoopCount to zero

But "=" is also used in askiascript 2.0 to denote a logical comparison":

```
Age.Value = 3
```
Check if Age.Value is 3

Therefore, to use a virtual variable in a logical comparison, you need to put the expression between "(" and ")"

```
(LoopCount = 0)
```
Check if LoopCount is zero

Defining the virtual variable type

→ *Show PowerPoint Slide "Defining the virtual variable type"*

Defining the virtual variable type

- The type of a virtual variable is determined when you first assign a value to it
- Subsequently, you can only assign values of the same type to it

Type	Example
Numeric	`Dim my_var = 0`
String	`Dim my_var = " "`
Array (for sets)	`Dim my_var = {}`
Date	`Dim my_var = #25/03/2011#`
Time	`Dim my_var = #16:32:00#`
Date and time	`Dim my_var = #25/03/2011 16:32:00#`
Boolean (logical)	`Dim my_var = true`

- The type of a virtual variable is determined when you first assign a value to it.
- Subsequently, you can only assign values of the same type to it; if you try to assign a value of a different type, you will get an error message.

Type	Example
Numeric	`Dim my_var = 0`

String	`Dim my_var = ""`
Array	`Dim my_var = {}`
Date	`Dim my_var = #25/03/2011#`
Time	`Dim my_var = #16:32:00`
Date and time	`Dim my_var = #25/03/2011 16:32:00#`
Boolean (logical)	`Dim my_var = true`

→ Explain that a Boolean value takes one of only two values: true or false. True is given the internal value 1 and false is given the internal value 0. This means it is possible treat a Boolean virtual variable like a numeric variable – but it will only ever have the value 1 or 0.

■ Notes for participants

Note that Boolean variables accept one of only two values: true or false. True is given the internal value 1 and false is given the internal value 0. This means it is possible treat a Boolean virtual variable like a numeric variable – but it will only ever have the value 1 or 0.

Working with sets and arrays

Useful keywords for working with sets and arrays in the askiascript language.

→ Present the sequence of six PowerPoint slides (slides 40-45):

- *"Sets and arrays"*
- *"Sets and arrays: adding a second set" and*
- *"Intersection and Union operators".*

→ Note these slides contain animations – press the RETURN key to move to the next slide only after all the animations are complete.

→ In the example questionnaire Q1 and Q1_S both provide identical sets of answers, which are about power tools owned at the respondent's main home and second home respectively.

Union or '+'

→ `Union` merges the values contained in two different sets, such as the responses to two questions. You can use `union` or `+` interchangeably.

→ Explain that, for the value to be present in the union of two sets, it must be present in either of the sets being combined, which includes any values which are present in both of the sets.

Notes for participants

`Union` merges the values contained in two different sets, such as the responses to two questions. You can use **union** or **+** interchangeably.

For the value to be present in the union of two sets, it must be present in either of the sets being combined, which includes any values which are present in both of the sets.

Assuming Q1 contains 6 and 4, and Q1_S contains 6, 1 and 8:

```
Q1 Union Q1_S
```

The result will be `{6; 4; 1; 8}`

Intersection

→ `Intersection` retrieves the common values of two sets.

→ It compares the selected responses given to two questions, and discovers which responses were selected in both.

→ Explain that, for the value to be present in the intersection of two sets, it must be present in both of the sets being combined.

→ Add a routing that is true if the respondent has a hedge trimmer (response 3) at *both* their main and second homes:
`Q1 Intersection Q1_S Has 3`

→ Explain that the order of the returned items is determined by the leftmost set (the one before the `intersection` keyword). For example:

Notes for participants

`Intersection` retrieves the common values of two sets. It compares the selected responses given to two questions, and discovers which responses were selected in both.

For the value to be present in the intersection of two sets, it must be present in both of the sets being combined.

Assuming Q1 contains 6 and 4, and Q1_S contains 6, 1 and 8:

```
Q1 Intersection Q1_S
```

The result will be {6}

Note that the order of the returned items is determined by the leftmost set (the one before the intersection keyword).

Operations with sets

→ Show PowerPoint slides

- “Operations with Sets” and
- “Challenge”

→ Explain that you can intersect a variable with a constant set:

```
{1 to 6} intersection Q1
Q1 intersection {1 to 6}
```

→ Explain that you can start any expression with the empty set { } to ensure that the expression is evaluated as a set and returns a set, e.g.

```
Dim Q1combi = { } + Q1 + Q1_S
```

→ Ask the group what each of the three expression above will give if Q1 contains the set {6; 4} and Q1_S contains the set {1; 8; 6}.

→ Answers:

- The first will contain the values { 4; 6 } because the order is determined by the set on the left
- The second will contain the values { 6; 4 }
- In the third, Q1combi will contain the values { 6; 4; 1; 8 }

■ Notes for participants

You can intersect a variable with a constant set. Examples:

Example	Result
`{1 to 6} intersection Q1`	If Q1 contains the set {6; 3} then the result will be {3; 6} because the order is determined by the set on the left
`Q1 intersection {1 to 6}`	If Q1 contains the set {6; 3} then the result will be {6; 3} because the order is determined by the set on the left

You can start any expression with the empty set { } to ensure that the expression is evaluated as a set and returns a set, e.g.

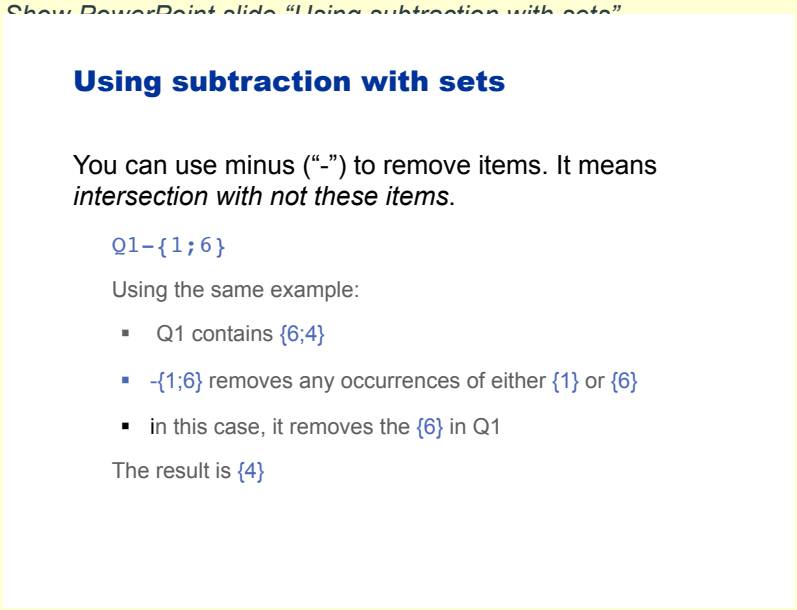
```
{} + Q3 + Q4 HasAll {5; 6; 7}
```

Returns true if all of the responses 5, 6 and 7 were selected at Q3 or Q4 (e.g. it would be true if 5 and 7 were selected at Q3 and 6 and 7 were selected at Q4).

Using subtraction with sets

You can use minus ("–") to remove items. It means *intersection with not these items*.

→ *Show PowerPoint slide "Using subtraction with sets".*



Size() function

The `size` function is particularly useful when working with sets and combinations of sets as it tells you how many values the set contains – i.e. the number of answers given.

→ *Show PowerPoint slide "Size() function".*

→ Explain that *size* retrieves the number of answers given in a subset.

For example, the following script returns the number of responses in Q1 and Q1_S combined (as a union of the two):

```
Size(Q1.Value + Q1_S.Value)
```

Note that when we use the Size keyword, we should also use Value, to ensure we retrieve the correct value.

The following script returns the number of responses selected by the respondent from the first five codes to Q1:

```
Size(Q1.Value Intersection {1 to 5})
```

The following script returns the number of responses selected by the respondent from codes 1, 3 and 4:

```
Size(Q1.Value Intersection {1;3;4})
```

→ Now demonstrate these examples by switching to the Power Tools example .QEX file *PowerTools_without_hh_loop_WITH_routings.qex*.

→ Show the first routing for *Cordless_tools_MH*. This returns the number of responses selected at *Cordless_tools_MH* and stores the number in *Number_Cordless_tools_MH*.

→ Explain that if you combine two questions with a union (as per routings Q1 and Q1_S), then duplicate responses will only be counted once (so in our example,

you will get a maximum of nine product types, even if some are selected in both Q1 and Q1_S). Also note that examples use "+" (which is the same as "Union").

→ Explain that when using the `Size` function, the `.Value` keyword is required.

■ Notes for participants

Note: If you combine two questions in this way with a union, then duplicate responses will only be counted once.

.SelectRandom(n) method

→ *Show PowerPoint slide ".SelectRandom(*n*) method".*

→ `SelectRandom` allows you to select a specific number of values at random from a set. For example, this could be the responses given to a multi-coded question.

→ Now demonstrate some of these examples by switching to the *PowerTools_without_hh_loop_WITH_routings.qex* file

→ Show the routing after *Total_tools_Own* (the routing sets a value to Q8loop). This routing selects up to three random product categories from the variable Total_tools_Own (which is a total of those selected at Q1 and Q1_S).

→ Talk through these other examples using the slide "*.SelectRandom(n) method examples*":

- {1 to 10}.SelectRandom()
  Returns one value randomly from 1 to 10 (such as 3).
- Dim i = {1 to 10}
  i.SelectRandom(2)
  Returns two values randomly from 1 to 10 (such as 3 and 7) and stores the values in a declarative (temporary) variable called "i".
- {5 to 10}.SelectRandom(3)
  Returns three values randomly from 5 to 10 (such as 5, 6 and 9).
- Q4.Value.SelectRandom(2)
  Returns two values randomly from the ones selected in Q4 (such as 2 and 4).
- (Q1 + Q4).SelectRandom()
  Returns one value randomly from the ones selected in Q1 and Q4 (such as 3).

→ Explain that the sequence of returned numbers reflects the order of values in the array. So `{1 to 10}.SelectRandom(3)` will return 3 values but the lower will always be the first one, the next highest will be always the second value, and so on, because the array is ordered from 1 to 10. If you want to shuffle these selected responses then you can use the `Shuffle()` method.

Notes for participants

`SelectRandom` allows you to select a specific number of values at random from a set. For example, this could be the responses given to a multi-coded question. The syntax is as follows:

```
.SelectRandom(n)
```

Example	Result
`{1 to 10}.SelectRandom()`	Returns one value randomly from 1 to 10 (such as 3).
`Dim i = {1 to 10}` `i.SelectRandom(2)`	Returns two values randomly from 1 to 10 (such as 3 and 7).
`{5 to 10}.SelectRandom(3)`	Returns three values randomly from 5 to 10 (such as 5, 6 and 9).
`Q4.Value.SelectRandom(2)`	Returns two values randomly from the ones selected in Q17 (such as 2 and 4).
`(Q1 + Q4).SelectRandom()`	Returns one value randomly from the ones selected in Q1 and Q4 (such as 3).

The sequence of returned numbers reflects the order of the values in the array. So `{1 to 10}.SelectRandom(3)` will return 3 values but the lower will always be the first one, the next highest will be always the second value, and so on, because the array is ordered from 1 to 10. If you want to shuffle these selected responses then you can use the `Shuffle()` method.

.Shuffle() method

→ Show PowerPoint slide ".Shuffle( ) method".

Use `.Shuffle` on an array to return the array in a randomised order

```
Q1.Value.Shuffle()
```

This returns all the values in a random order

`.Shuffle` can be used with `.SelectRandom` for a random selection in a random order

```
Q1.Value.SelectRandom(3).Shuffle()
```

This returns three randomly selected values in a random order.

So, if Q1_S contains {1; 2; 5; 7}, the `.SelectRandom(3)` method will select three items, such as {1; 5; 7} and shuffle will place these in a random order, e.g. {5; 7; 1}

```
{1 to 10}.SelectRandom(3).Shuffle()
```

It will therefore return three values randomly and shuffled (such as 7; 2; 5).

Count property

Use Count to return the number of items in an array.

→ Remaining in the *PowerTools_without_hh_loop_WITH_routings.qex* file, create a routing to demonstrate the Count property

→ Explain that the `Count` property can be used as an alternative to the `Size` function to obtain the number of answers to a question:

- For example, if Q1 has four responses, then `Q1.Value.Count` returns 4.

→ Explain that there is an alternative way to achieve the same effect. The following script returns *true* if Q1 has at least one response selected between 1 and 5:

```
Dim my_array = Q1 Intersection {1 to 5}

If my_array.Count > 0 Then
  Return true
Else
  Return False
EndIf
```

■ Notes for participants

Use *Count* to return the number of items in an array. It can be used as an alternative to the Size function to obtain the number of answers to a question.

For example, if Q1 has four responses, then

```
Q1.Value.Count
```

returns 4.

There is an alternative way to achieve the same effect. The following script returns *true* if the q1 has at least one response selected between 1 and 5:

```
Dim my_array = Q1 Intersection {1 to 5}

If my_array.Count > 0 Then
  Return true
Else
  Return False
EndIf
```

Recap

→ Show PowerPoint Slide "Summary"

→ Provide a one-minute summary of the topics covered and then prepare participants for the exercise.

In this session, we have explored:

- Using `Dim` (dimension) to declare new array variables

- Combining and performing logic on sets or array variables using `Union` and `Intersection`
- Subtracting items from a set with "–"
- Using the `Count` property to detect the number of items in an array
- Using `SelectRandom` to randomly pick items from an array
- Using `Shuffle` to randomise the order of a set

Practical exercises

→ Ask participants to complete these exercises.

Exercise 1: Creating question loops

Follow these steps:

1. Create a Brand Spontaneous awareness question with 10 brands and None.
   - Call it "*TopOfMind*".
   - For the list of brands, come up with own list. You might use Coca Cola, Dr Pepper, Evian, Minute Maid, Orangina, Perrier, Pepsi, Schweppes, Sprite, and/or one or more brands from your country.
   - Ensure you include a *None* category.
2. Create a Brand Spontaneous awareness variable as follows:
   - Call it "*Others"*.
   - Give it the same responses as *Top of Mind*, including the *None* category.
   - If *None* was selected in the *Top of Mind*, then do not show '*Others*' to the respondent, and set its value to *None*.
   - If one brand was selected in *Top of Mind*, then show the list of brands without the one selected in *Top of Mind*.
3. Create an "*Assisted*" Brand awareness variable with the same list of brands.
   - Call this question "*Assisted"*.
   - Do not show the brands selected in *Top of Mind* and *Others*.
   - If the respondent selected every brand in the combination of *Top of Mind* and *Others,* then do not ask *Assisted*.
4. Create a "*Favourite*" Brand variable with the same list of brands.
   - Call this question "*Favourite"*.
   - Only show the brands selected in *Top of Mind* + *Others* + *Assisted*.
   - Only ask this question if at least 2 brands were selected in *Top of Mind* + *Others* + *Assisted*.

- If only one brand was selected in *Top of Mind* + *Others* + *Assisted*, then do not show the *Favourite* variable and code it as the selected brand.

5. Create a "*Random Selection*" variable where you will select 3 brands at random from the ones known (*Top of Mind* + *Others* + *Assisted*).
6. Create a Loop to ask about the 3 brands selected.
   - Ask if the respondents will buy this product in the next month.
   - To do this, create a yes/no question which will be asked for each of the 3 brands.
   - The loop could be named "*Loop Buy*" and the question "*Buy*".
   - Do not ask the loop if none of the brands were selected.

Exercise 2: Using Dim

You will now be creating a script to check that the responses to three questions add up to 100.

Follow these steps:

1. Add the following chapter to your survey:

   Question type: *Chapter*

   Long caption: *When you are considering your financial future, how important are the short term (the next five years), medium term (five to ten years) or long term (more than ten years)? Please allocate 100 points between these three time periods, to indicate your priorities.*

2. Add the following three numeric questions:

   Question names: *Short_term, Medium_term, Long_term*

   Question type: *Numeric*

   Minimal value: *0*

   Maximal value: *100*

3. Add a further question:

   Question name: *Priority_total*

   Question type: *Numeric*

   Make the question invisible during data entry.

4. Add one or more routing conditions to add up the totals, using declared variable(s) to keep track of the running total. Store the running total in *Priority_total* (using the *Set value* routing action).
5. Add a new chapter, and have its long caption display *Priority_total*. Add a message to indicate how many further points they need to allocate. Add routing logic to ensure that the chapter is only displayed if *Priority_total* is less than 100.

Session 150-4 Advanced Loops

Outline

Topics presented

In this session, we will introduce you to:

- The loop properties of questions defined within a loop
- The loop built-in test values **IsLastIteration** and **CurrentIteration**
- The **For / Next** loop
- The **Break** directive

Learning outcomes

At the end of this session you will understand:

- How to perform tests and computations of the different elements of array variables within loops
- How to know, when executing a script, how many times a loop has executed
- How to take particular actions when you are on the last iteration of a loop
- How to create loops that will repeat a defined number of times – even if this number is calculated in real time
- How to create loops that will repeat until a particular defined state is reached

Tutorial

In this session, the tutor will be demonstrating the nested loop in the example QEX (PowerTools_with_hh_loop_WITH_routings.qex). Please pay close attention to the narrative you will find below when leading this session.

Questions residing in standard loops

Any question defined within a standard loop (such as a loop to create a question grid) has additional object properties pertaining to the loop.

→ Explain the following.

→ When the **start question** of a routing is set on a question which is inside a loop, the routing is executed on each of the iterations.

→ For example: we have a question inside a loop, called *Q2,* which asks how each type of power tool owned is powered. If the start question of the routing is on the loop, or on the question, then it is the same as having three routings: one on each type of power source (gas, corded or cordless).

→ In a routing condition where the start question is in a loop, the `Value` property of the question always references the question in the current iteration. If you want to access values in other loop items, you can use the `Iteration()` method.

→ Explain the household/second household principle that is used in the second QEX (*PowerTools_with_hh_loop_WITH_routings.qex*).

- This is achieved with a nested loop (or "loop within a loop").
- A paper copy of this QEX can be found at the back of the particpants' booklet.

→ In the *QEX*, show the routing for Q2 (as shown below). This displays a message when the first response (Gas) is displayed on the first loop iteration.

- Routing type: condition
- Start question: Q2
- Condition: Q2.iteration(1).Value = 1
- Action: Show message: "Gas" selected on first iteration.

→ Talk through these other examples:

- `Q2.Iteration(1).Value '=>will return the value for the first item.`
- `Q2.Iteration(1).Answers.Index '=>will return the value for the first item`

- Explain that the two solutions above will return the same result. However, `.Value` will return exactly and only the value, whereas `.Answers` can be used in combination with other useful keywords – such as `.Caption`, as we will see next.

- `Q2.Iteration(3).Answers.Caption '=>will return the caption for the third item.`

Notes for participants

Any question defined within a standard loop (such as a loop to create a question grid) has additional object properties pertaining to the loop.

When the **start question** of a routing is set on a question which is inside a loop, the routing is executed on each of the iterations.

For example: we have a question within a loop called Q2 which asks how each type of power tool owned is powered. If the start question of the routing is on the loop, or on the question, then it is the same as having three routings: one on each type of power source (gas, corded or cordless).

In a routing condition where the start question is in a loop, the *value* property of the question always references the question in the current iteration. If you want to access values in other loop items, you can use the **Iteration()** method.

Example:

If the first response (“Gas”) is displayed on the first loop iteration or Q2.

```
Q2.iteration(1).Value = 1
```

will return “Gas”.

Other examples:

Example	Result
`Q2.Iteration(1).Value`	Returns the value for the first item.
`Q2.Iteration(1).Answers.Index`	Returns the value for the first item.
`Q2.Iteration(3).Answers.Caption`	Returns the caption for the third item.

Loops within Loops

The principle is the same with a **loop of loops**.

→ In the QEX *PowerTools_with_hh_loop_WITH_routings.qex*, show the loop of loops based on the response given at S3.

- Q2 asks the respondent to select the type of energy used by each power tool they own. There are, therefore 27 responses (up to 3 fuel types for each of 9 power tools).

  - If a routing’s **start question** is the loop statement, or the question inside the second loop, it is the same as having 27 routings.

- If the **start question** is the first loop or on a question inside the first loop but outside the second loop, then it is equivalent to 9 routings.

→ In loops of loops, *Iteration* lets you indicate the loop shortcut.:

```
Q2.Iteration(Q2Loop:3,ResidenceLoop:2)
```

- This example would reference the third iteration of Q2Loop within the second household

→ This is equivalent to

```
Q2.Iteration(ResidenceLoop:3,Q2Loop:3)
```

→ **Note**: If the start question of the routing is outside the loop, the iteration method is mandatory.

■ Notes for participants

The principle is the same with a **loop of loops**. If we have a loop of loops with 27 responses (3 fuel types for each of 9 power tools), then if a routing's **start question** is the loop statement, or the question inside the second loop, it is the same as having 27 routings. If the **start question** is the first loop or on a question inside the first loop but outside the second loop, then it is equivalent to 9 routings.

In loops of loops, *Iteration* lets you indicate the loop shortcut.:

```
Q2_S.Iteration(Q2Loop:3,ResidenceLoop:2)
```

This is equivalent to

```
Q2_S.Iteration(ResidenceLoop:2,Q2Loop:3)
```

Note: If the start question of the routing is outside the loop, the iteration method is mandatory.

Object methods for loops

There are two object methods relevant to objects created inside loops which can be used in comparisons. These methods will only return a value when used on an object within a loop.

→ *Show PowerPoint slide "Object methods for loops"*

IsLastIteration and *CurrentIteration*

Referencing specific iterations of a loop.

→ Explain the following:

→ We can test whether we are in the latest iteration of the loop by using `IsLastIteration`:

- `Q2.IsLastIteration`
- `IsLastIteration` returns true or false.

→ If you need to refer to a specific iteration, then you can use the `CurrentIteration` property.

- In **routing mode**, show the routings for Q2loop, where responses are ignored, depending on the iteration of the loop.
- Explain that you can refer to `CurrentIteration` in the following ways:
  - `Q2loop.CurrentIteration = 9`
  - `Q2loop.CurrentIteration.EntryCode = "09"`

→ `CurrentIteration` returns a numeric value.

Notes for participants

There are two object methods relevant to objects created inside loops which can be used in comparisons. These methods will only return a value when used on an object within a loop.

IsLastIteration and *CurrentIteration*

You can test at what point in the execution of the loop your script has reached. `IsLastIteration` is true if the script has reached the final iteration of the loop.

```
Q2loop.IsLastIteration
```

If you need to refer to a specific iteration, then you can use the `CurrentIteration` property. This returns, as a numeric value, the current iteration number of the loop.

```
Q2Loop.CurrentIteration = 9
```

Is true if this is the ninth iteration of the loop.

An alternative way to achieve this is:

```
Q2loop.CurrentIteration.EntryCode = "09"
```

AllValues property

To obtain an array of all the answers to a question in a loop, we use the keyword `AllValues`.

→ Show the set of two PowerPoint slides "AllValues"

→ Explain the following examples from the slides:

Examples:

- If Q10 is a single question inside a loop three iterations, `Q10.AllValues` might return

  `{2; 1; 8}.`

- If Q11 is a multi-coded question, `Q11.AllValues` might return

  `{2; 5; 8; 1; 4; 8}`

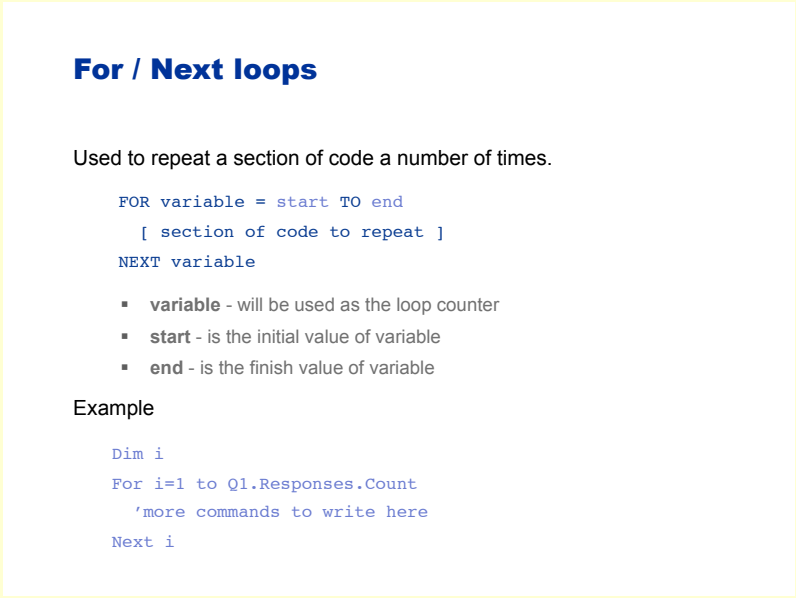
→ Return to the PowerTools_with_hh_loop_WITH_routings.qex and show the routing for Q1:

→ Explain that in this example it returns an array of all values selected at Q1 and stores it in Total_tools_own (with the Set Value action). This is used as the basis of the answer list for Q5loop (so the respondent is asked about tools owned).

For/Next Loops

A `FOR/NEXT` loop is an entirely different construct to a question loop. Used in a script, it allows you to create repeating, or "iterative" logic at any point within your script. `FOR/NEXT` loops do not create repeated questions in the way ordinary loops do.

→ *Show PowerPoint slide "For / Next loops"*



→ **Explain:**

- The control variable (loop counter) has a differing value each time through the loop.
- Remember that you need to declare the control variable with the `Dim` keyword.
- The 'start' and 'end' values specify what the control *variable's* values will be and how many times the loop is executed.

- Every time the 'next' line is reached, the value of *variable* is incremented by 1 (+1).
- When the control variable is outside the start to end values, the looping stops and program flow continues from the line after the next command.
- `FOR…NEXT` loops can be nested (remember to use a different variable for each loop).
- Open the QEX file and use the following example:
  - show the routing for the chapter `Setvalue`, which creates an array of cordless power tools owned by the respondent and stores the value (with the **set value** action) in the variable *Cordless_tools*.

Notes for participants

Syntax:

```
FOR control_variable = start TO end
      [section of code to repeat]
NEXT control_variable
```

control_variable - will be used as the loop counter

start - is the initial value of variable

end - is the finish value of variable

- The control variable (loop counter) has a differing value each time through the loop.
- The 'start' and 'end' values specify what the control *variable's* values will be and how many times the loop is executed.
- Every time the 'next' line is reached, the value of *variable* is incremented by 1 (+1).
- When the value in the control variable is outside the start to end values, the looping stops and program flow continues from the line after the next command.
- `FOR…NEXT` loops can be nested (remember to use a different variable for each loop).

Example

The code used is as follows:

```
Dim Arr_cordless = {}

Dim i

Dim j

  'Parent loop iteration

i = Residenceloop.CurrentIteration

For j=1 to Q2loop.Responses.Count
```

```
  If Q2.Iteration(Residenceloop : i, Q2loop : j)=3 Then
    Arr_cordless = Arr_cordless.Insert(j)
  EndIf
Next j
' return array of cordless tools
return Arr_cordless
```

→ Explain to the participants what the above script does, and see if they can work out what the `Insert` statement does in this script.

→ Explain that it is *not* good practice to manipulate the number of times that a loop might iterate by using a variable for the *end* value and then altering the value of that variable during the execution of the loop.

When looking at the above example, it is OK to use `Q2loop.Responses.Count` for the end value of the "j" loop, as that value will not change during the execution of the loop. However, if you used a variable (e.g. "x") for the end value of the loop and the value of "x" was changed inside the loop, this could result in a never-ending loop.

- This can give rise to an endless loop during an interview, and unpredictable results.
- You should always use either a system-defined value, a variable that will not change, or a constant, so that the outcome is predictable.
- Inside the loop, if you want to check the value in the control variable, you should put the expression in parentheses to avoid setting the value. For example:

  ```
  If (LoopCount = 1) then 'check whether this is the first iteration of the loop
  ```

  This ensures that you checking the value, rather than setting it.

→ If you are basing a loop on another part of your questionnaire – e.g. the number of responses to Q1 – this should not use an absolute number, but instead refer to the item by name (e.g. `Q1.Responses.Count`). This ensures that if the item being referenced is changed (e.g. more responses added), the script will still work.

Note that it is not good practice to manipulate the number of times that a loop might iterate by using a variable for the *end* value and then altering the value of that variable during the execution of the loop.

For example, if our loop begins `For LoopCount=1 to x`, then we should not change the value of `x` inside the loop. If we do:

- This can give rise to an endless loop during an interview, and unpredictable results.

- You should always use either a system-defined value, a variable that will not change, or a constant, so that the outcome is predictable (e.g. `For LoopCount = 1 to Q2loop.Responses.Count`).
- Inside the loop, if you want to check the value in the control variable, you should put the expression in parentheses to avoid changing the value. For example:

  ```
  If (LoopCount = 1) then 'check whether this is the
  first iteration of the loop
  ```

  This ensures that you checking the value of the control variable (in this case, *LoopCount*), rather than setting it.

Break command

You can use **break** to end a FOR/NEXT loop before the control variable reaches the end value.

→ *Show PowerPoint slide "Break command"*

Break command

Allows you to end a loop before the end value has been reached

```
FOR variable = start TO end
  [ commands]
   IF condition THEN
      BREAK
   ENDIF
NEXT variable
```

Note: Break should be written within a condition (If, Elseif or Else) otherwise the loop will always terminate during the first iteration

→ *Explain the following:*

→ Normally, a loop will repeat a set number of times, defined by the difference between the *start* and *end* values.

→ There are occasions where you might want the loop to end before the end value has been reached, and for this you can use the `break` keyword.

→ For example, suppose we want to check whether the respondent picked the first product category before the second one at Q1. Demonstrate the following:

- Create a new dummy variable in the Power Tools survey (*PowerTools_with_hh_loop_WITH_routings.qex)*, as follows:

- Shortcut: picked_first
- Type: closed question, single coded
- It should have one response: "true"

- Create a new routing instruction as follows:

  - start question: Q1
  - Condition: True
  - Action: set value
  - Target question: picked_first
  - Value if true: enter a script to check whether response 1 was selected before response 2, and if so, set the value in picked_first to true. For example, you could use the following script:

Notes for participants

Normally, a loop will repeat a set number of times, defined by the difference between the *start* and *end* values. However, there are occasions where you might want the loop to end before the end value has been reached, and for this you can use the `Break` keyword.

For example, suppose we want to check whether the respondent picked the first product category before the second one at Q1.

```
dim i
dim picked_a = False
for i = 1 to Q1.answers.count
  if Q1.answers[i] = 1 then
    picked_a = true
    break
  elseif Q1.answers[i] = 2 then
   break
 endif
next i

return picked_a
```

Recap

→ *Show PowerPoint Slide "Summary"*

→ Provide a one-minute summary of the topics covered and then prepare participants for the exercise.

In this session, we have looked at:

- Using loops to test and update elements of array variables
- Checking to see which iteration of a loop is currently being performed with the two built-in loop values `CurrentIteration` and `IsLastIteration`
- Writing Loops that repeat a defined number of times with `For … Next`
- Loops within loops
- Using `AllValues` to aggregate the values from loop questions
- Using the `Break` directive to exit a loop

Practical exercise

→ Ask the participants to complete this exercise.

→ For your reference, the file *session 150-4 model answer.qex* provides a model solution to this exercise.

Writing script loops

Follow these steps:

1. Ask the respondents about their hotel stays in the last year. Find out what proportion of their stays was for business, personal or other purposes. For each type of stay, ask them to allocate a percentage (0%-100%).

   You can achieve this by having a numeric question inside a loop.

   The sum of the percentages should equal 100. If it does not equal 100%, then show a warning message.

   Hint: you can use the property AllValues.Sum to check the value of the numeric question.

   Note: To show the current statement in the long caption, you can use ??ShortCutOfTheLoop??. Drag the loop from the left-hand panel into the Long Caption of the question to insert it automatically.

2. Create another loop with 5 brands and 2 questions attached.

   Would you consider buying this brand: [brand name]?
   Responses: Yes, No

Why would you not buy [brand]?
Only ask if no was selected in the first question (one screen per brand).

3. Outside the loop, create a summary multiple question including all the brands with a *yes* selected. Ask the respondent to select his or her first choice.

Appendix: Example questionnaire

POWER TOOLS SURVEY
(North America)

SCREENER SECTION

S1 (Screener). Do you have any power that you use tools at home? By power tools we mean tools like a lawn mowers, hedge trimmers, chain saws, pressure washers or power hand tools.

- Yes – *Continue*
- No - *Close*

S2 (Screener) Are these tools that you own personally, or are they tools used for business or trade?

- Only tools used for business or trade - *Close*
- Only tools owned personally - *Continue*
- A mixture of both - *Continue*

S3 (Screener) Do you also own a second residence for your own use, such as a vacation home?

- Yes - *Continue*
- No - *Continue*

MAIN QUESTIONNAIRE

Q1. Which of these power tools do you own?
IF yes at S3, use this text instead: **Which of these power tools do you own, and normally keep at your main home? For the moment, we will just focus on those those at your main home.]**

- Push Lawn mower (1)
- Riding Lawn mower (2)
- Hedge trimmer (3)
- Chain saw (4)
- String trimmer (5)
- Lawn edger (21)
- Pressure Washer (6)
- Hammer drill or power drill (7)
- Power sander/orbital sander (8)

Q2. What/And what/And finally, what kind of [power tool at Q1] is it?
*Use the text "**What"** the fist time the question is presented, "And what" for each subsequent time, and use **"And finally"** the last time Q2 is presented, provided if it has been presented more than twice.*

- Gas
- Electric (corded)
- Cordless /rechargeable

Note: the following power options are allowed at Q2

	Gas	Electric	Cordless/rechargeable
Push Lawn mower	Y	Y	Y
Riding Lawn mower	Y	n	Y
Hedge trimmer	Y	Y	Y
Chain saw	Y	Y	n
String trimmer	Y	Y	Y
Lawn edger	Y	Y	Y
Pressure Washer	Y	Y	n
Hammer drill or power drill	n	Y	Y
Power sander/orbital sander	n	Y	Y

If 2 or more cordless/rechargeable tools.
Q3. Are you aware that some power tool manufacturers offer interchangeable rechargeable batteries or power packs that you share between your power tools?

- Yes
- No
- Not sure

If Q3.yes
Q4. Do you share batteries between any of these tools that you have?
Only present those selected which are battery-powered:

- Hedge trimmer
- String trimmer
- Lawn edger
- Hammer drill or power drill
- Power sander/orbital sander

Q1_S. Now we would like to focus on any power tools you may normally keep at your second home. Which of these power tools do you own, and keep at your second home?
IF yes at S3, add: **For the moment, we are just interested in those at your main home.]**

- *same answer list as at Q1*

Q2_S. What kind of [power tool at Q1_S] is it?
*Use the text "**What"** the fist time the question is presented, "And what" for each subsequent time, and use **"And finally"** the last time Q2_S is presented, provided if it has been presented more than twice.*

- *same answer list as at Q2*

If 2 or more cordless/rechargeable tools. AND Q3 was not asked:

Q3_S. Are you aware that some power tool manufacturers offer interchangeable rechargeable batteries or power packs that you share between your power tools?

- *same answer list as at Q2*

If Q3.yes or Q3_S.yes
Q4_S. Do you share batteries between any of these tools that you have?
Only present those selected which are battery-powered:

- *same answer list as at Q2*

Q5. Which of these tools are you likely to replace in the next 12 months?
(Only present those selected at Q1 or Q1_S)

- Push Lawn mower
- Riding Lawn mower
- Hedge trimmer
- Chain saw
- String trimmer
- Lawn edger
- Pressure Washer
- Hammer drill or power drill
- Power sander/orbital sander

For each selected, if power source is currently not cordless, ask:
Q6. Would you consider purchasing a rechargeable/cordless [tool at Q5]?
(If more than one item selected "And similarly, …"

- Yes
- No
- Not sure

If Q6.no ask:
Q7. What are your reasons for not choosing a cordless or rechargeable [tool at Q5]

- *Open text response*

From the combined answers to Q1 and Q1_S, randomly select no more than 3 answers, and repeat Q8 for each answer selected.
Q8. What feature do you like the most about the [selected tool at Q1/Q1_S] you currently own?

- *Open text response*