Scripting

LexisNexis

LexisNexis

Road Map

- Introduction to scripting
- Message functions
- Variables
- User interaction functions
- Using scripts in precedents
- Data retrieval functions
- Text insertion functions
- If … statements

Outcomes

- Understand the concept of scripting and how it builds functionality in Affinity
- Understand programming concepts such as functions, variables and syntax
- Build interactivity using user interaction functions
- Retrieve information from Affinity using scripting
- Place information into precedents using scripting
- Use if… statements to cater for all eventualities encountered when a script is run
- Apply scripting techniques to various Affinity applications
- Employ advanced scripting techniques where required

About the Activities

- There will be activities throughout this course
  - An opportunity for application and reinforcement
- These may be done in groups (max. 2-3 per group) if desired

LexisNexis

Road Map

- **Introduction to scripting** 
- Message functions
- Variables
- User interaction functions
- Using scripts in precedents
- Data retrieval functions
- Text insertion functions
- If … statements

LexisNexis

What does scripting do? (1)

- Allows you to build additional functionality within Affinity
- Highly customisable because you build it

What does scripting do? (2)

- With scripting, you can:
  - Make your life easier with far fewer documents needing to be maintained
  - Allow the user to make choices while a precedent is running
  - Include variable paragraphs
  - Perform calculations with data
  - Check that information has been entered
  - Check that information entered is appropriate
  - Cater for more eventualities – e.g. missing matter party contact details

What is a script?

- Instructions for computers
- These instructions need to cater for all possible eventualities

Meet Marvin.

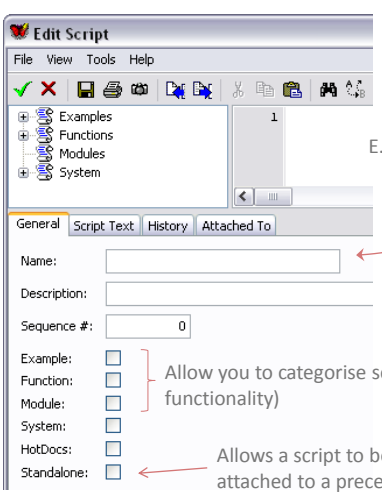
Write instructions for him to make a cup of tea.

LexisNexis

Don’t make assumptions

LexisNexis

Creating a Script



Provides the a description of the script’s purpose

E.g. “CONV_CALC_STAMP_DUTY”

Allow you to categorise scripts (does not affect functionality)

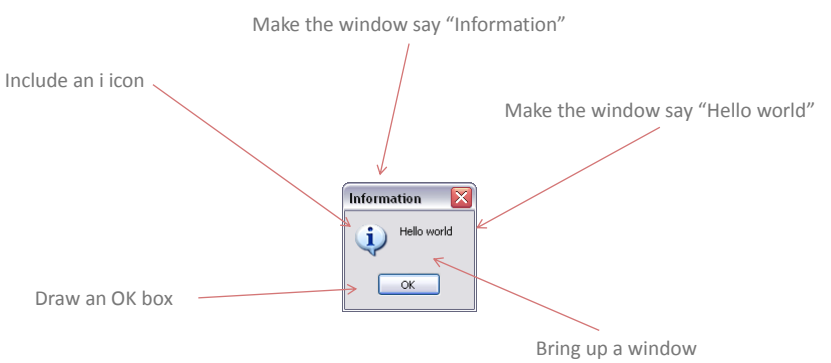
Allows a script to be run without being attached to a precedent or DataForm

PROCESS

1. Case Management → Scripts
2. New Script
3. 'General 'tab
   a) Name and description (call it TRAINING SCRIPT1)
   b) Options (Tick "standalone")
4. 'Script Text' tab

```
apMessage("Hello world!")
```

5. Save
6. Run



Imagine we had to get the computer to bring up a "Hello world box" manually. That's a lot of code.

LexisNexis

What is apMessage()?

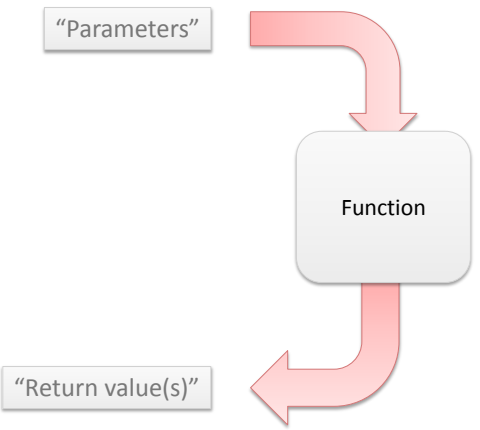
- apMessage() is an example of a **function**
- **Functions** are pre-packaged modules of code
- They are building blocks you can use to make writing code easier

LexisNexis

Functions

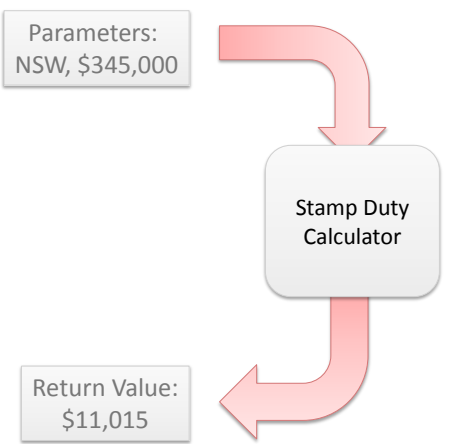
LexisNexis

Functions



LexisNexis

Functions



Affinity Functions

ap_____

Affinity comes with pre-defined functions to simplify common tasks

What they do can be determined from the user reference

Functions Available: 4 Categories

Functions that provide feedback to the user

Functions that prompt the user for information

Functions that obtain information from Affinity

Functions that insert text in documents

LexisNexis

Overview

Ask user what they want

Extract details from Affinity

Make decisions & Perform calculations

```
(if type="Doctor", then do A, B, C
X = Y + Z)
```

Place information into the document

LexisNexis

Python is a language

Give the ball to I

LexisNexis

Python is Case Sensitive

apMessage ≠ apmessage

LexisNexis

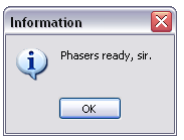
Road Map

- Introduction to scripting
- **Message functions** 
- Variables
- User interaction functions
- Using scripts in precedents
- Data retrieval functions
- Text insertion functions
- If … statements

LexisNexis

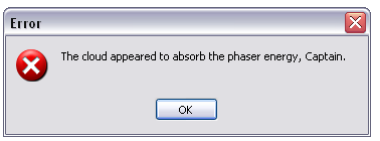
Message Functions

apMessage



apWarning

apError



LexisNexis

The Command Bar

Mini Quiz 1

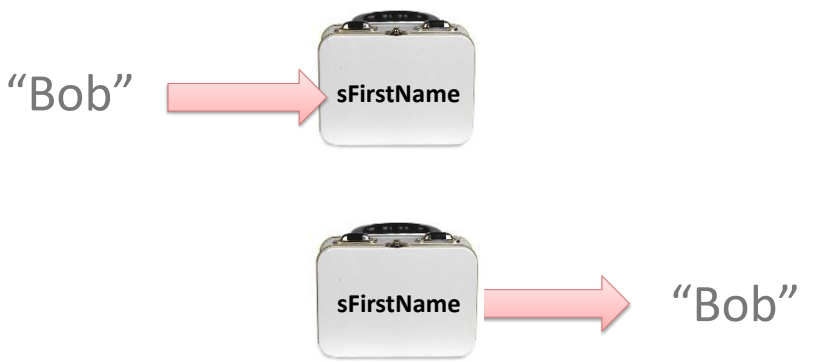
- To be provided as course progresses

Road Map

- Introduction to scripting
- Message functions
- **Variables**
- User interaction functions
- Using scripts in precedents
- Data retrieval functions
- Text insertion functions
- If … statements

LexisNexis

What are variables?



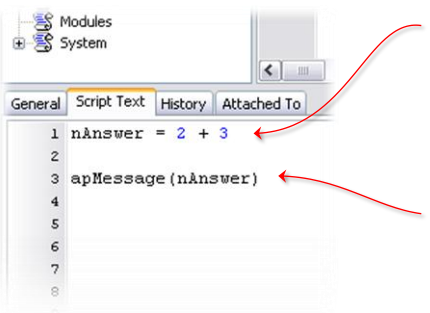
Variables are words we attach values to

LexisNexis

Why use variables?

- When you ask the user a question, you might wish to store their answer for the rest of the script
- When you make a calculation, you might wish to store the end result for later use
  - This allows you to re-use the result of previous calculations or functions
  - This simplifies your code, and reduces the chance of your code containing errors over time

How are variables used?



This line calculates 2 + 3 and attaches the result (5) to the word **nAnswer**

This line displays a message. The value of **nAnswer** is used as the message to display.

Variable Names Can't Contain Spaces

✗ Number of parties = apGetWordLinkField(...

✓ nNumberOfParties = apGetWordLinkField(...

LexisNexis

Storing information in variables

```
a = "red"
b = "green"
a = b
apMessage(a)
```


LexisNexis

Text vs. Variables

```
sGreeting = "Hello"
apMessage(sGreeting)
apMessage("sGreeting")
```

What do you expect to see? Run the above script in Affinity to check

LexisNexis

Programming Conventions

- To keep things tidy:
  - Name your variables properly
  - Always put comments in your code

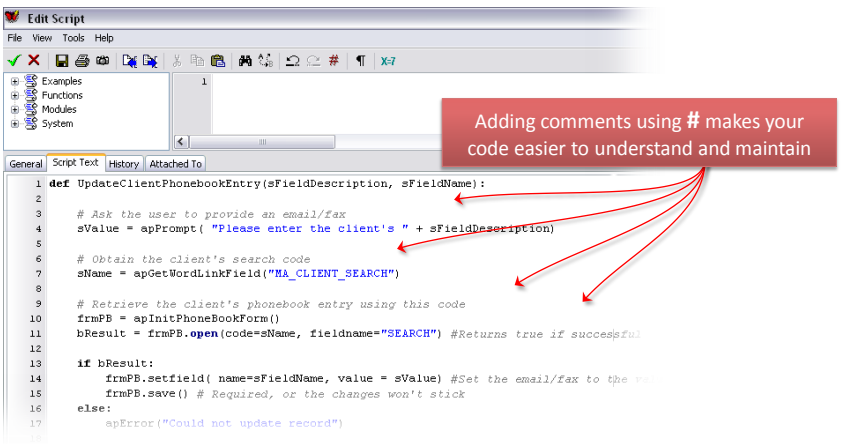
LexisNexis

Variable Naming

Type of information it holds	Variable should start with…	Example
String (i.e. “Text”)	s	sFirstName = “John”
Date	d	dSettlement = “1/1/2012”
Amount (i.e. “Currency”)	a	aSalary = 100000
Number	n	nPropertiesCount = 2
Boolean (i.e. “True/False”)	b	bDetailsPresent = True
List	l	lNames = [“John”, “Mary”, “Jo”]

Technically, this convention is called “Systems Hungarian” notation. Having a mix of lower and upper case is called “Camel Case”.

Adding Comments

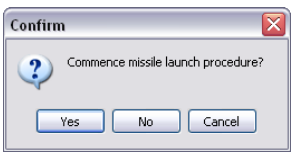


Road Map

- Introduction to scripting
- Message functions
- Variables
- **User interaction functions**
- Using scripts in precedents
- Data retrieval functions
- Text insertion functions
- If … statements

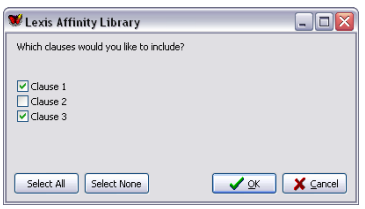
User Interaction Functions

apPrompt



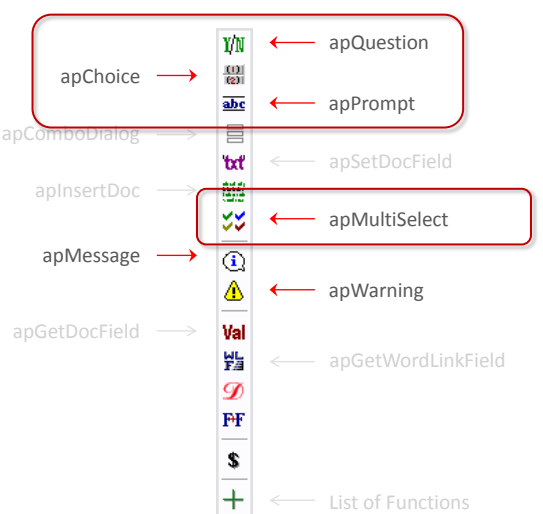
apQuestion

apChoice



apMultiSelect

The Command Bar



Mini Quiz 2

- To be provided as course progresses

Activity (S1)

- To be provided as course progresses

User Interaction Functions: Return Values (1)

Function	Requires User to Enter…	Return Values
apPrompt	Any text	Text
apQuestion	Yes/No	True/False
apChoice	Single selection	Number representing choice made
apMultiSelect	One or more boxes ticked	List of True/False values Position 0 is False if no options were selected

Testing it out

- Go back and edit your script from the previous activity
- After each prompt, use apMessage(…) to display the user's choice
  - E.g. sResponse = apPrompt(…)
    apMessage(sResponse)

Note: These values won't be shown to the end user normally!

LexisNexis

Which function to use for which type?

Type of information it holds	Variable should start with…	Functions that return this variable type
String (i.e. “Text”)	s	apPrompt
Date	d	apPrompt
Amount (i.e. “Currency”)	a	apPrompt
Number	n	apChoice
Boolean (i.e. “True/False”)	b	apQuestion
List	l	apMultiSelect, apComboDialog

LexisNexis

Road Map

- Introduction to scripting
- Message functions
- Variables
- User interaction functions
- **Using scripts in precedents**
- Data retrieval functions
- Text insertion functions
- If … statements

LexisNexis

Revision

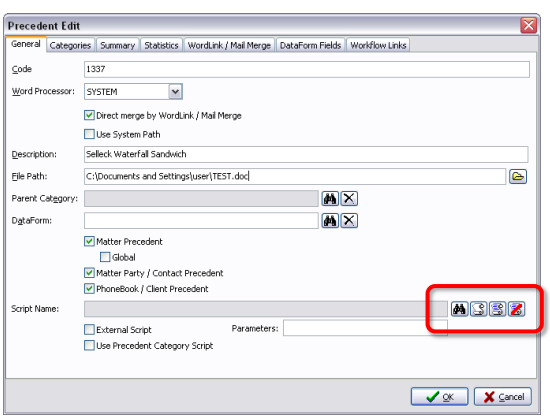
- Does everybody remember how to register precedents in Affinity?

LexisNexis

Attaching Scripts to Precedents

PROCESS

1. Case Management → Precedents
2. Right-click
3. Edit…
4. Attach Script
5. OK



LexisNexis

Road Map

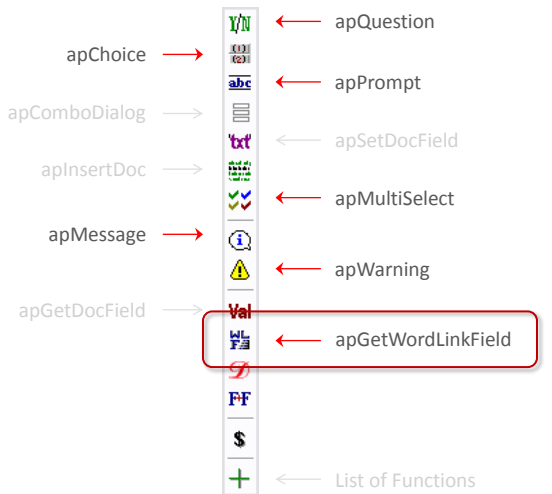
- Introduction to scripting
- Message functions
- Variables
- User interaction functions
- Using scripts in precedents
- **Data retrieval functions** 
- Text insertion functions
- If … statements

LexisNexis

Data Retrieval Functions

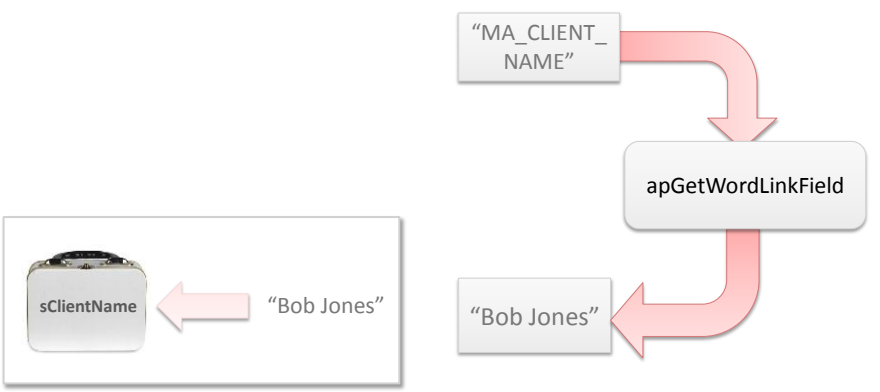
apGetWordLinkField
apGetPhoneBook
apGetMatter

The Command Bar



apGetWordLinkField: Getting Information

sClientName = apGetWordLinkField("MA_CLIENT_NAME")



Activity (S2)

- To be provided as course progresses

Road Map

- Introduction to scripting
- Message functions
- Variables
- User interaction functions
- Using scripts in precedents
- Data retrieval functions
- **Text insertion functions** 
- If … statements

Functions that insert text in documents

apSetDocField
apGetWordLinkField
apInsertDoc

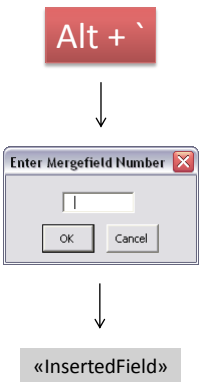
Text Insertion Functions: apSetDocField

apSetDocField
apGetWordLinkField
apInsertDoc

apSetDocField

The Command Bar

Inserting Fields in Word



Activity (S3)

- To be provided as course progresses

Text Insertion Functions: apGetWordLinkField

apSetDocField
apGetWordLinkField
apInsertDoc

apGetWordLinkField: Setting Information

1 It retrieves the information from Affinity

apGetWordLinkField(command="MA_CLIENT_NAME", fieldname="ClientName")

2 It places the information into the mergefield

«ClientName»

apGetWordLinkField: Setting Information

```
sClientName = apGetWordLinkField( command="MA_CLIENT_NAME")
apSetDocField( fieldname="ClientName", value= sClientName )
```

```
apGetWordLinkField( command="MA_CLIENT_NAME",  fieldname="ClientName")
```

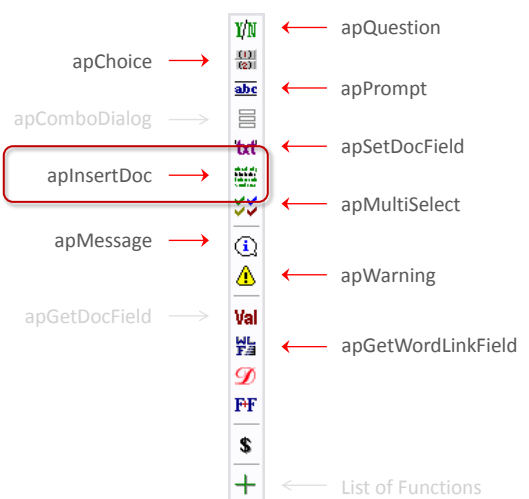

Activity (S4)

- To be provided as course progresses

Text Insertion Functions: apInsertDoc

apSetDocField
apGetWordLinkField
apInsertDoc

The Command Bar



INSERTDOC Command: Recap

INSERTDOC<Header.doc>

Dear Jim,

RE: Sale of Disneyland

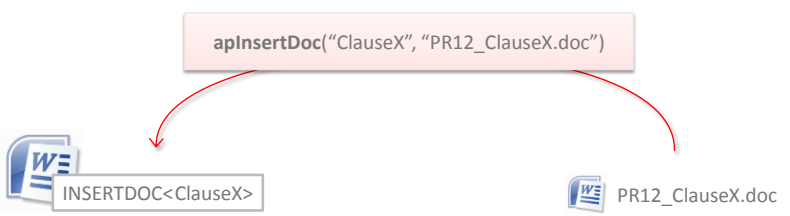
We write to confirm that Contracts for the sale of the above property were exchanged on 17/09/2013 and copies of the relevant pages of the Contract for Sale are enclosed for your information

Branch: SYDNEY
Our Ref: 012360:Damien Montague

29 March 2014

Header.doc

apInsertDoc



LexisNexis

Testing it out

- Delete everything in Training Letter from the sign-off line onwards
- Replace it with: INSERTDOC<YF>
- Add the below line to your TRAINING SCRIPT3 script
- Test your precedent out to make sure it still works

```
apInsertDoc("YF", r"%MATTER%\YF.doc")
```

Look for INSERTDOC<YF>

Treat the \ as just a \ ('r' stands for 'raw')

Look in the Matter Paragraphs base folder

LexisNexis

Functions that insert text in documents

Function	Inserts…
apSetDocField	A word or phrase you specify
apGetWordLinkField	Information directly from Affinity
apInsertDoc	Another document

Mini Quiz 3

- To be provided as course progresses

Road Map

- Introduction to scripting
- Message functions
- Variables
- User interaction functions
- Using scripts in precedents
- Data retrieval functions
- Text insertion functions
- **If … statements**

LexisNexis

If… Statements

- Remember scripts need to cater for all eventualities!
- E.g. Marvin needs to add milk only if the user wants it

LexisNexis

if… Statements

```
command one

if X == Y:
        do this
        and this
        and this

command two
```

```
if You == “happy”:
        stompyour(feet)

if YourLove > 0:
        letmeknow()

if client.name == “Vanilla Ice”:
        apError(“Stop!”)
        apMessage(“Collaborate”)
        apMessage(“Listen”)
```

Indentation is Important

```
bNeedLetter = apQuestion("Do you need letter 1242?")
if bNeedLetter == True:
        apMergeDoc("1242")
        apMessage("The letter has been prepared")

apSetDocField("Sample", "This is an example")
```

vs.

```
bNeedLetter = apQuestion("Do you need letter 1242?")
if bNeedLetter == True:
        apMergeDoc("1242")
apMessage("The letter has been prepared")

apSetDocField("Sample", "This is an example")
```


Comparison vs. Assignment

```
if a == b:
    apMessage("They're equal!")
```

If statements require **==**

```
a = 1
b = a + 1
apMessage(b)
```

Otherwise, use **=**

LexisNexis

Other Comparisons

Symbol	Meaning
==	Is equal to
!=	Is not equal to
>	Is greater than
>=	Is at least
<	Is less than
<=	Is no more than

LexisNexis

Activity (S5)

- To be provided as course progresses

Using apQuestion()

```
response = apQuestion("Would you like to include clause X?")
if response == True:
        <Actions to perform if they click yes>
else:
        <Actions to perform if they click no>
```


Using apQuestion()

```
response = apQuestion("Would you like to include clause X?")
if response == True:
        <Actions to perform if they click yes>
```

(If you don't want to do anything if they click no, don't include the second half)

Using apChoice()

```
choice = apChoice("Which letterhead would you like to include?",
                  "Conveyancing Letterhead",
                  "Commerical Letterhead",
                  "Family Letterhead")
if choice == 1:
    <Actions to perform if they choose the first option>
if choice == 2:
    <Actions to perform if they choose the second option>
if choice == 3:
    <Actions to perform if they choose the third option>
```

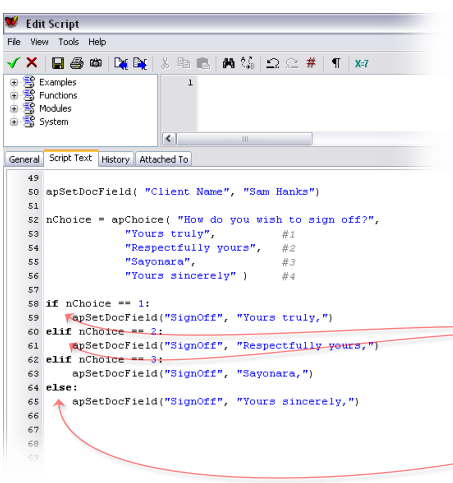

Using apMultiSelect()

```
selections = apMultiSelect("Which clauses would you like to include?",
                           "ClauseA",
                           "ClauseB",
                           "ClauseC")
if selections[0] == True:
    apMessage("You didn't make any selections")
if selections[1] == True:
    <Actions to perform if they choose the first option>
if selections[2] == True:
    <Actions to perform if they choose the second option>
if selections[3] == True:
    <Actions to perform if they choose the third option>
```

Ifs within ifs

```
response = apQuestion("Would you like to include a letterhead?")
if response == True:
        choice = apChoice("Which letterhead would you like to include?",
                          "Conveyancing Letterhead",
                          "Commercial Letterhead",
                          "Family Letterhead")
        if choice == 1:
                <Actions to perform if they choose the first option>
        if choice == 2:
                <Actions to perform if they choose the second option>
        if choice == 3:
                <Actions to perform if they choose the third option>
else:
        <Actions to perform if they click no>
```


If … Elif … Else




```
if a == 1:
   apMessage("It's one")
elif a == 2:
   apMessage("It's two")
else:
   apMessage("It's something else")
```

elif means "otherwise, if"

else means "otherwise" or "if all else fails"

LexisNexis

Activity (S6)

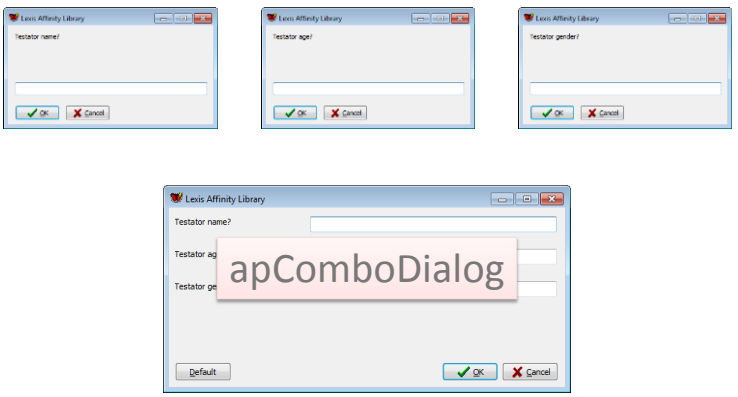
- To be provided as course progresses

LexisNexis

Application Menu

- apComboDialog & apShowDataForm
- Joining Text Fragments
- Scripting Letterheads
- Matter Party Functions
- Inserting Formatting into Documents
- Variable Paragraphs
- Plurals within Documents
- DataForm Scripting
- User-defined Functions
- Advanced Menu

Combining Prompts Together



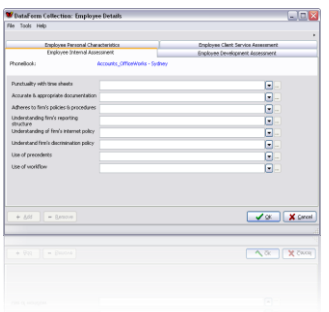
Using apComboDialog()

```
valueslist = apComboDialog( "Loan Amount:", "", "", "A", "$#,##0.00", "TEMP",
                            "Lower Interest Rate:", "", "", "N", "###0", "TEMP",
                            "Higher Interest Rate:", "", "", "N", "###0", "TEMP")

apSetDocField( fieldname="LoanAmountField", value=valueslist[0], type="A", format="$#,##0.00" )
apSetDocField( fieldname="LowerInterestField", value=valueslist[1], type="N", format="##.00%" )
apSetDocField( fieldname="HigherInterestField", value=valueslist[2], type="N", format="##.00%" )
```

Showing DataForms in Scripts

apShowDataForm()

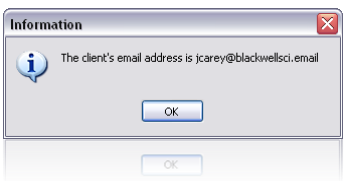


Which to Use?

apComboDialog	apShowDataForm
Data is not stored permanently • Can be tidier	Data is stored permanently • May save time if precedent needs to be re-merged
Can only use apPrompt-type questions	Can contain several different types of fields, e.g. date fields
Inflexible - if you want three names, you have to show three fields	Can be multi-instance, allowing to user to specify as many copies as desired

Joining Text Fragments (String Concatenation)

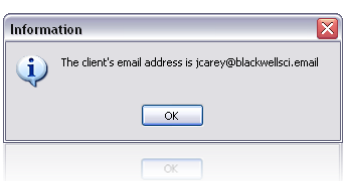
```
sEmail = apGetWordLinkField("MA_CLIENT_EMAIL")
apMessage("The client's email address is " + sEmail)
```



Inserting Variables Into Phrases

```
sEmail = apGetWordLinkField("MA_CLIENT_EMAIL")
apMessage("The client's email address is %s" % sEmail)
```

'String formatting operator'



Inserting Variables Into Phrases

```
nNumberBeverages = '99'
sBeverageType = 'beer'
sBeverageLocation = 'wall'
apMesssage('There were %s bottles of %s on the %s...' %  \
        (nNumberBeverages , sBeverageType, sBeverageLocation ))
```

Activity (S7)

- To be provided as course progresses

Scripting Letterheads

Precedent 1

MA_CLIENT_NAME
MA_CLIENT_MAILADDRESS
MA_CLIENTMAILSSP2

Client Letter.doc

Precedent 2

MA_MP(TYPE=Agent,FIELD=NAME)
MA_MP(TYPE=Agent,FIELD=MAILADDRESS)
MA_MP(TYPE=Agent,FIELD=MAILSSP2)

Agent Letter.doc

Precedent 3

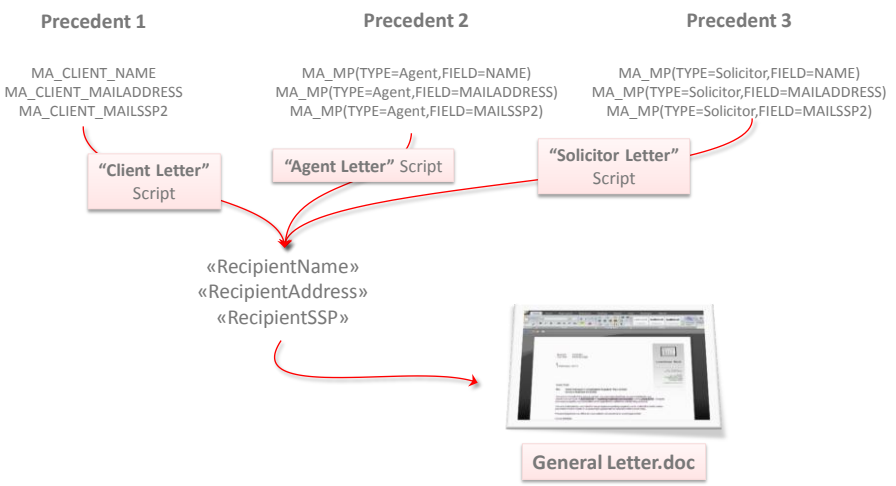
MA_MP(TYPE=Solicitor,FIELD=NAME)
MA_MP(TYPE=Solicitor,FIELD=MAILADDRESS)
MA_MP(TYPE=Solicitor,FIELD=MAILSSP2)

Solicitor Letter.doc

Activity (S8)

- To be provided as course progresses

Document Minimisation



Activity (S9)

- To be provided as course progresses

Matter Party Functions: apShowMatterParty

Testing it out

Redesign your GEN_AGENT_LETTER script to use the below.

Test it again

```
apShowMatterParty(type="Agent",selectgroupID="AGENT1")
apGetWordLinkField("MA_MP(CMD=?,id=AGENT1,Field=Name)", "RecipientName")
apGetWordLinkField("MA_MP(CMD=?,id=AGENT1,Field=MailAddress)", "RecipientAddress")
apGetWordLinkField("MA_MP(CMD=?,id=AGENT1,Field=MailSSP2)", "RecipientSSP")
apGetWordLinkField("MA_MP(CMD=?,id=AGENT1,Field=Salutation)", "Salutation")
```

Now test your precedent on matter 012339. What happens?

Additional Matter Party functions

What if the matter we're working with has no Agents on the contacts tab?

apGetMatterPartyCount(type="…", role="…")

- Returns the number of Matter Parties that match the type and/or role you specify, for the current matter.

apShowMatterPartyNew(type="…", role="…", selectgroupID="…")

- Create a new Matter Party for the current matter that has the type and/or role you specify.
- Optionally, label that Matter Party with the selectgroupID you specify.

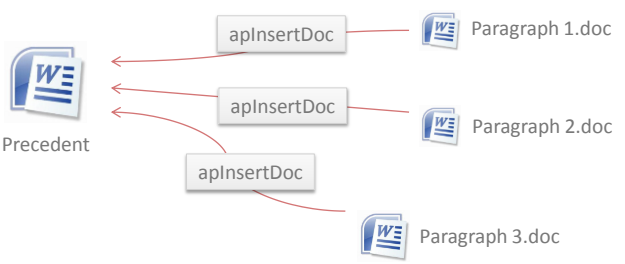
Activity (S10)

- To be provided as course progresses

Activity (S11)

- To be provided as course progresses

apInsertDoc: Inserting Variable Paragraphs



Testing it out

Our Ref: «MA_FILEID»«MA_AUTHOR»

Thursday, 10 February 2011

«Addressee»
«Address»
«SuburbStatePC»

We're going to create a script that inserts paragraphs into a precedent

We'll then develop it so that it allows the user to select which paragraphs to insert

Dear «Salutation»,

RE: «MA_LONGDESCR»

We advise that we act for the buyer «BuyerName» in this matter.

In order for us to advance this matter, please provide us with the following information:

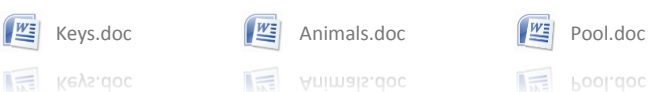
- Confirmation that the keys will be available for collection by the buyer at time of settlement.
- Confirmation that animals are allowed within the confines of the unit complex.
- Details advising what hours the swimming pool is accessible.

Yours sincerely,

«FIRM_COMPANY»

Testing it out: Creating the Paragraphs

1. Locate the paragraphs folder
   - This will be found in System Settings → Document
2. Create 3 paragraph documents in that folder
   - The content of these paragraphs is not really important (they don't have to be the same as the previous slide)
   - Be sure to end each with at least paragraph mark/carriage return



Testing it out: Creating the Precedent

1. Create a precedent with INSERTDOC commands for each paragraph
   - Start off by making a copy of a previous letter; call it 'Property Details'
   - Create all the INSERTDOC commands on the same line
2. Create a script (INSERT_PARAS) to process the INSERTDOCs
   - Start off by copying the GEN_AGENT_LETTER script if you have it
3. Test your new precedent

```
apInsertDoc("KEYS", r"%MATTER%\Keys.doc")
```

Look for INSERTDOC<KEYS>

Treat the \ as just a \ ('r' stands for 'raw')

Look in the Matter Paragraphs base folder

Activity (S12)

- To be provided as course progresses

LexisNexis

Inserting New Lines

```
sComment = apPrompt("Add an additional comment")
if sComment != "":
   apSetDocField( fieldname="AdditionalComment" value=sComment + "<newpara>")
else:
   apSetDocField( fieldname="AdditionalComment" value="")
```

Inserts a new paragraph

LexisNexis

Inserting Formatting

```
state = apChoice( "What state is the property in?",
                  "NSW",
                  "QLD",
                  "VIC" )

if state == 1:
        apSetDocField( fieldname="section", value="section 123 of the <i>Relevant Act (NSW) 2014</i>" )
if state == 2:
        apSetDocField( fieldname="section", value="section 45 of the <i>Relevant Act (QLD) 2015</i>" )
if state == 3:
        apSetDocField( fieldname="section", value="section 6A of the <i>Relevant Act (VIC) 2016</i>" )
```

Everything between and will be italicised

Inserting Formatting and New Lines

Insert Formatting

Insert Tab

New Line

New Paragraph

Break Time

Take a break!

LexisNexis

Plurals in Documents

"John Smith and Mary Wu agree that…"

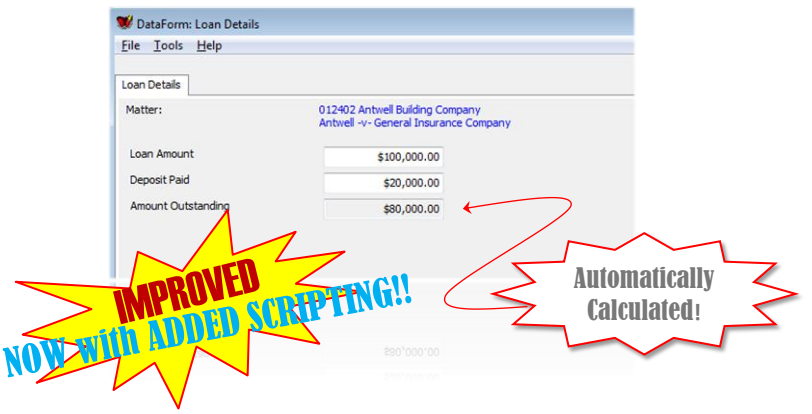
"Mary Wu agrees that…"

LexisNexis

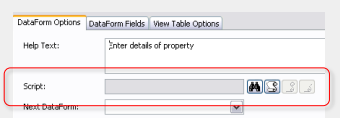
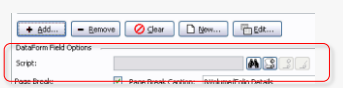
Activity (S13)

- To be provided as course progresses

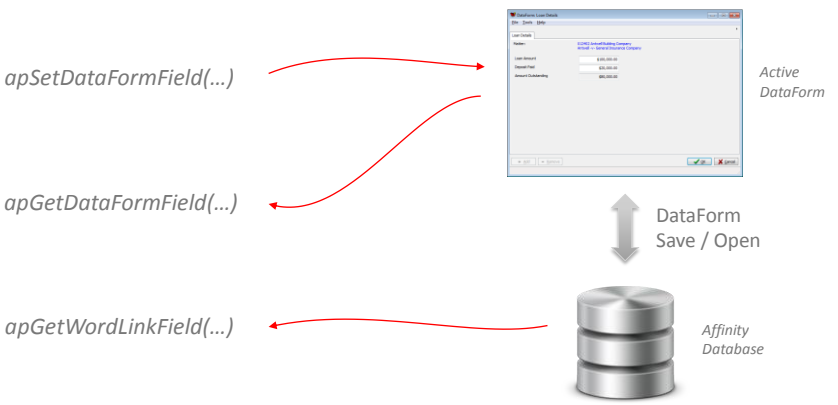
Scripting in DataForms



Adding Scripts to DataForms

Script Location	How to add a script here	When does the script run?
DataForm	Choose the "DataForm Options" tab 	When 'OK' is clicked
DataForm Field	1. Choose the "DataForm Fields" tab 2. Select the field 	*If added to a script type:* When the button is clicked *If added to any other type:* When the field information is updated

Three DataForm Scripting Functions (1)



Three DataForm Scripting Functions (2)

- apGetDataFormField(fieldname="…", type="A"/"S")
  - Gets the value of the field even if the user hasn't yet clicked save
  - The type parameter determines what kind of value is returned
- apGetWordLinkField("MA_DF(…)")
  - Gets the value of the field at the last time it was saved
- apSetDataFormField(fieldname="…", value="…")
  - Set the value of a field

Predefined DataForm functions

- def **DataFormOnOpen**():
  - Any code placed here will run when a DataForm is opened
- def **DataFormOnInstanceLoad**():
  - Any code place here will run when an instance is loaded in a multi-instance DataForm
- def **DataFormOnSave**():
  - Any code place here will run after the end user clicks 'OK', but before changes are saved.
- def **DataFormAfterSave**():
  - Any code place here will run after changes are saved

Predefined DataForm Field functions

- def **DataFormFieldOnEnter**(sender):
  - Any code here will run when the user clicks on a field
- def **DataFormFieldOnExit**(sender):
  - Any code here will run when the user moves away from a field (which includes when they leave the DataForm)
- def **DataFormFieldOnLookupChange**(sender):
  - Only runs when a lookup is applied to a text field

'Sender' is the name of the DataForm field that is being entered/exited

Making fields “read only”

‘Read only’ fields cannot be changed by users. Only scripts can change their values.

Testing it out

- Create a DataForm with two date fields:
  - Exchange Date
  - Settlement Date (make this read only)
- Attach a script that calculates the settlement date
  - Assume a 42-day settlement period

Activity (S14)

- To be provided as course progresses

Creating Your Own Functions

- Imagine you've just come back from a holiday
- Everyone asks you how it went
- You find yourself saying the same things over and over again

Creating Your Own Functions

Imagine if you could simply record what you have to say, and then hit "Play" everytime someone asks you.

Creating Your Own Functions

def **FillOutAddressBlock()**:

Script A

```
apGetWordLinkField("MA_CLIENT_NAME", "RecipientAddress")
apGetWordLinkField("MA_CLIENT_MAILADDRESS", "Address")
apGetWordLinkField("MA_CLIENT_MAILSSP2", "RecipientSSP")
apGetWordLinkField("MA_CLIENT_SALUTATION", "Salutation")

sysPurchRole = "Purchaser"
sysVendorRole = "Vendor"

sClient = int(apGetWordLinkField(
command="MA_MP(TYPE=MATTER CLIENT,ROLE=" +
sysPurchRole + ",CMD=COUNT)", fieldname="" ))
if sClient >=1:
        sClient = "Purchaser"
        apSetEPField("Field287","P")
        sysPurch1Type = "MATTER CLIENT"
        sysPurchType = "ASSOCIATED PARTY"
        sysPurch2Set = "1"
        sysPurch2Type = "OTHER PARTY"
        sysVendor1Type = "OTHER PARTY"
        sysVendor2Set = "2"
else:
        sClient = "Vendor"
        apSetEPField("Field287","V")
        sysVendor1Type = "MATTER CLIENT"
        sysVendorType = "ASSOCIATED PARTY"
```

Script B

```
apGetWordLinkField("MA_CLIENT_NAME", "RecipientAddress")
apGetWordLinkField("MA_CLIENT_MAILADDRESS", "Address")
apGetWordLinkField("MA_CLIENT_MAILSSP2", "RecipientSSP")
apGetWordLinkField("MA_CLIENT_SALUTATION", "Salutation")

#Gets Purchase Price
aPurchasePrice = apGetDataFormField(
"CON_PURCHASE_PRICE", "A" )

#Gets Deposit Paid
nDepPaid = apGetDataFormField(
fieldname="CON_DEPOSIT_PAID" )

#Calculate Balance Owing
nBalOwe = apStrToNumber(aPurchasePrice)-
apStrToNumber(nDepPaid)
apSetDataFormField("CON_BALANCE_OWING", nBalOwe)

#Automatically calc                    n purchase price.
aPurchasePrice = apGetDataFormField(
fieldname="CON_PURCHASE_PRICE", type="A" )
aGSTAmount = aPurchasePrice * 0.1
apSetDataFormField( fieldname="CON_GST_AMOUNT",
value=aGSTAmount )
```

Script C

```
apGetWordLinkField("MA_CLIENT_NAME", "RecipientAddress")
apGetWordLinkField("MA_CLIENT_MAILADDRESS", "Address")
apGetWordLinkField("MA_CLIENT_MAILSSP2", "RecipientSSP")
apGetWordLinkField("MA_CLIENT_SALUTATION", "Salutation")

#Gets the precedent category of the precendent being run
sSQLText = "SELECT PC.DESCR "
sSQLText = sSQLText + "FROM PREC P, PRECCATEGORY PC "
sSQLText = sSQLText + "WHERE P.NPREC = " + apRunTimeField(
'PRECID' ) + " AND P.NPRECCATEGORY = PC.NPRECCATEGORY "

dsPrec = apCreateDataSet( "PREC", sSQLText )

sPrec = dsPrec.FieldByName( "DESCR" ).AsString

del dsPrec

#If precendent is gene                    rt just the matter long
description in RE line
#Else also include the matter title
sType = apRunTimeField("KEYTYPE")

if sType =="PHONEBOOK":
    apInsertDoc( "RE", r"%PHONEBOOK%\RE.doc" )
    apSetDocField("sRE","//Please enter RE Line//")
```

LexisNexis

Creating your own functions

User defined functions contain sections of often-repeated code, to save having to recreate them each time

```
def FillOutAddress():
   apGetWordLinkField("MA_CLIENT_NAME", "RecipientName")
   apGetWordLinkField("MA_CLIENT_MAILADDRESS", “RecipientAddress")
   apGetWordLinkField("MA_CLIENT_MAILSSP2", “RecipientSSP")
   apGetWordLinkField("MA_CLIENT_SALUTATION", "Salutation")
```


LexisNexis

Listing All Functions

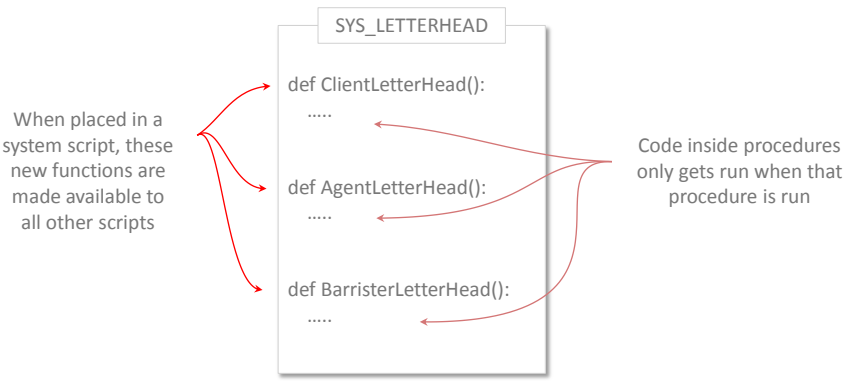
User-defined functions also appear

Ctrl + Space brings up a list of all functions

LexisNexis

System Scripts

- We normally group all of our user-defined functions together in System Scripts



LexisNexis

Activity (S15)

- To be provided as course progresses

LexisNexis

Advanced Menu

- 'While' Loops
- Boolean Operators
- Performing Calculations
- Lists and 'For' Loops
- Passing Variables Between Scripts
- Generator Expressions
- Data Types
- Showing Named Parameters
- Wrap Up

LexisNexis

While Loops

Think back for a second to when we programmed Marvin to add sugar to tea

LexisNexis

While Loops

Sugar needs to be stirred.
How long do you stir for?
How do you know when to stop stirring?

```
while bSugarDissolved==False:
   RotateSpoon()
```

LexisNexis

Using Boolean operators in if… statements

```
If bWantMilk:
   addMilk()

If not bInformationEntered():
   apShowMatterParty()
```

```
If bWantMilk==True:
   addMilk()

If bInformationEntered()==False:
   apShowMatterParty()
```

LexisNexis

Calculations in Python

Operation	Python Syntax	Example
Addition	+	4 + 5
Subtraction	-	9 - 2
Multiplication	*	2 * 3
Division	/	18 / 9
Exponent	**	3 ** 2

Also, what is 3 + 4 x 5?

LexisNexis

Lists

- Lists can be a useful way of storing information
- They represent a collection of items
- To refer to an item in the list, use […]. E.g.
  - lBradyBunch[0] = “Marcia Brady”
  - lBradyBunch[1] = “Carol Brady”
  - lBradyBunch[2] = “Greg Brady”

Note: Lists start at 0

[“Marcia Brady”, “Carol Brady”, “Greg Brady”, “Jan Brady”, “Peter Brady”, “Cindy Brady”, “Mike Brady”, “Bobby Brady”]

LexisNexis

Lists

- You can create a list of numbers using **range**(…), e.g.
  - range(5) = [0, 1, 2, 3, 4]
- You can easily count how many items are in a list using **len**(…)
  - len([“orange”, “green”, “blue”]) = 3

LexisNexis

‘For’ Loop

- A ‘for loop’ allows you to process all items in a list

```
lColours = [‘blue’, ‘red’, ‘green’, ‘yellow’]
for item in lColours:
        apMessage(item)
```

LexisNexis

Other looping techniques

```
lColours = [‘blue’, ‘red’, ‘green’, ‘yellow’]
for i in range(len(lColours)):
        sColour = lColours[i]
        apMessage(‘Colour %s is %s’ % (i, sColour))
```

```
nContacts = apGetWordLinkNumberField("MA_MP(CMD=COUNT)")
for i in range(nContacts):
        apGetWordLinkField("MA_MP(SET=%s, FIELD=NAME" % (i+1) )
```


LexisNexis

Generator Expression

```
lList = [function(i) for i in range(…)]
```

```
lNumbers = [x**2 for i in range(4)]
```

will return: [0, 1, 4, 9]

```
lLower = [“blue”, “green”, “red”]
lCaps = [upper(colour) for colour in lLower]
```

will return: [“BLUE”, “GREEN”, “RED”]

Generator Expressions in Affinity

Obtaining a list of matter parties for a matter:

```
nContacts = apGetWordLinkNumberField("MA_MP(CMD=COUNT)")
lContacts = [apGetWordLinkField("MA_MP(SET=%s, FIELD=NAME" % (i+1) )
for i in range(nContacts)]
chosen = apMultiSelect( “Select the relevant parties:”, *set(lContacts ))
```


Generator Expressions in Affinity

Obtaining a list of DataForm field values for each instance:

```
nInstances =
apGetWordLinkField(“MA_DF(CMD=COUNT,DATAFORM=COURT_DATES)”
lDates = [apGetWordLinkField(“MA_DF(DATAFORM=COURT_DATES,SET=%s,
FIELD=DATE”) % i for i in nInstances )]
dMostRecentDate = max(lDates)
```

Passing variables between scripts

- Normally scripts only recognise variables that were defined within that script
- By defining a variable within a system script (outside of a user defined function), it becomes recognised by all scripts
- You can use this to pass variables between two scripts

Data Types (1)

1 + 2 = 3

Integer Integer Integer

“ab” + “cd” = “abcd”

String String String

Data Types (2)

1 + “ab” = Error!

Integer String

1 + “2” = Error!

Integer String

Data Types Conversion

str(1) + “ab” = “1ab”

Integer → String String String

1 + int(“2”) = 3

Integer String → Integer Integer

LexisNexis

More “Type Error” Examples

“banana” - “orange”

“3” - “2”

“banana” - 2

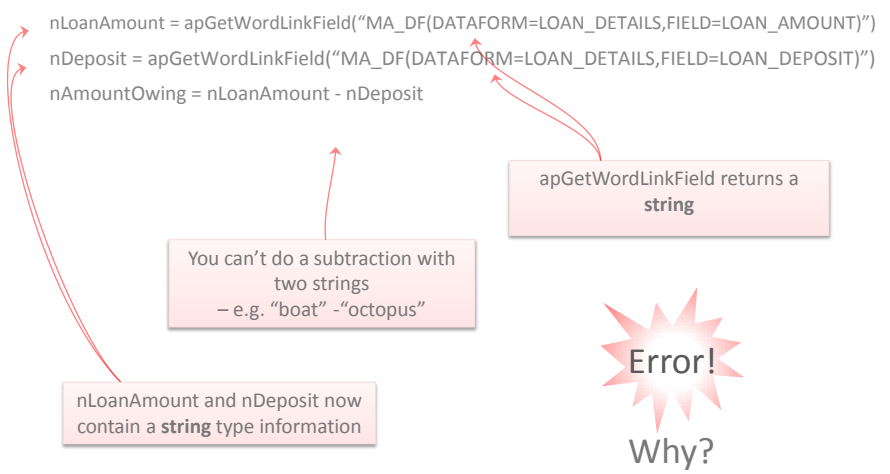
“banana” / “orange”

“banana” / 2

“apple” * 20

LexisNexis

Spot the Error



LexisNexis

The Solution

nLoanAmount = **int(**apGetWordLinkField("MA_DF(DATAFORM=LOAN_DETAILS,FIELD=LOAN_AMOUNT)"))

nDeposit = **int(**apGetWordLinkField("MA_DF(DATAFORM=LOAN_DETAILS,FIELD=LOAN_DEPOSIT)"))

nAmountOwing = nLoanAmount - nDeposit

OR

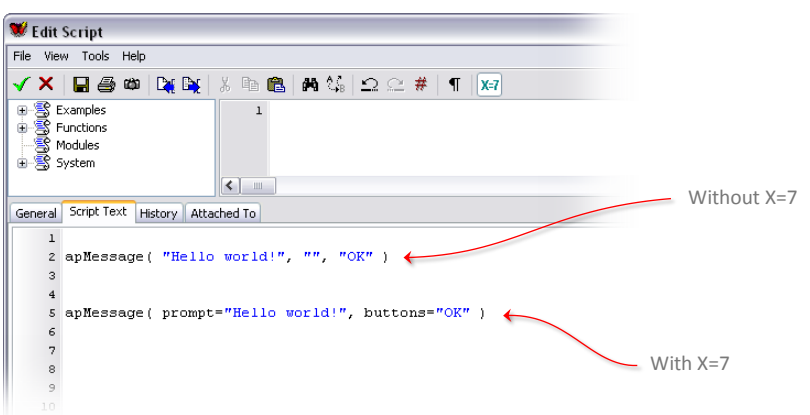
nLoanAmount = **apGetWordLinkNumberField**("MA_DF(DATAFORM=LOAN_DETAILS,FIELD=LOAN_AMOUNT)")

nDeposit = **apGetWordLinkNumberField**("MA_DF(DATAFORM=LOAN_DETAILS,FIELD=LOAN_DEPOSIT)")

nAmountOwing = nLoanAmount - nDeposit

LexisNexis

Showing parameters names



LexisNexis

Outcomes

- Understand the concept of scripting and how it builds functionality in Affinity
- Understand programming concepts such as functions, variables and syntax
- Build interactivity using user interaction functions
- Retrieve information from Affinity using scripting
- Place information into precedents using scripting
- Use if… statements to cater for all eventualities encountered when a script is run
- Apply scripting techniques to various Affinity applications
- Employ advanced scripting techniques where required