Coffee Time?

UNIVERSITY OF LIVERPOOL

Programming Construct Two: Selection

UNIVERSITY OF LIVERPOOL

Selection Statements

- Python, unlike some other programming languages, only has one types of selection statement, the “if-else” statement.
- Although Python support variations of this statement.

Problem Example 4: Triangles

UNIVERSITY OF LIVERPOOL

Triangles Introduction

- The anatomy of the Python "if-else" statement is as follows:

```
if (<BOOLEAN EXPRESSION>):
   <STATEMENTS X>
```

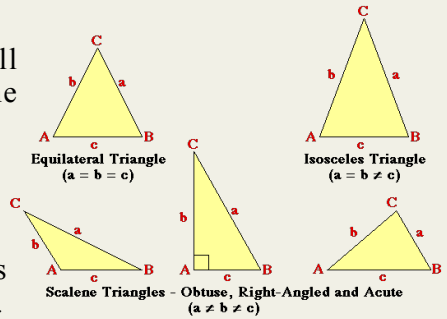
```
if (<BOOLEAN EXPRESSION>):
   <STATEMENTS X>
else :
   <STATEMENTS Y>
```

```
if (<BOOLEAN EXPRESSION>):
   <STATEMENTS X>
elif :
   <STATEMENTS Y>
else :
   <STATEMENTS Z>
```

Triangles Requirements

- Produce a Python program which, given three sides of a triangle, determines whether the triangle is either:

1. Equilateral (all sides the same length),
2. Isosceles (two sides the same length), or
3. Scalene (no sides the same length).



Triangles Source Code

- Load `PythonExampleProblems\Selection\Triangles\triangles.py` into the `IDLE` text editor

Triangles Comments (1)

- Lots of "if-else" statements!
- Test part of "if-else" can be a Boolean operator (<, >=, ==, <=, >) or a Boolean value (`True`, `False`).
- We can concatenate Boolean operators and values using logical operators (`and`, `or`, `not`).
- Note use of the `global` statement (only required when assigning values to "global variables").
- Test included to ensure user has defined a realisable triangle.

Run The Triangles Source Code

Test Cases
Designed to test every path through programme

Test Case			Expected result
Side A	Side B	Side C	
5	4	4	Isosceles
4	3	5	Scalene
4	5	4	Isosceles
3	4	5	Scalene
5	2	2	Error
4	4	4	Equilateral
4	4	2	Isosceles
4	2	4	Isosceles

Problem Example 5: Calculator

UNIVERSITY OF LIVERPOOL

Calculator Requirements

- Develop a calculator Python program that can resolve simple arithmetic expressions of the form:
  `<OPERAND> <OPERATOR> <OPERAND>`
  Where `<OPERAND>` is an integer of some kind and `<OPERATOR>` is one of the operators `+`, `-`, `*` or `/` (integer division).
- Thus given the expression `63*35` the program should calculate the value and display the result.
- **Remember to include a divide by zero test!**

Calculator Source Code

- Load `PythonExampleProblems \Selection\Calculator \calculator.py` into the `IDLE` text editor.

Calculator Comments

- Note `quit()` function.

Run The Calculator Source Code

Test Cases

Test Case			Expected result
operand1	operator	operand2	
3	+	2	5
3	-	2	1
3	*	2	6
3	/	0	Error
3	/	2	1
3	%	4	Error

- Try adding the % operator!

Data Structures

Data Structures

- The most commonly used Python data structures are:
  1. Lists
  2. Dictionaries
- (There are others, for example tuples)

Problem Example 6a: Distance Conversion Version 1 (Lists)

Distance Conversion Requirements

Design and create a piece of Python software that, when presented with a distance given in Metres converts it to a distance measure comprising Yards, Feet and Inches (1 Metre = 39.37 Inches, 12 inches = 1 foot, 3 feet = 1 yard). Output the result in whole Yards, Feet and Inches.

Distance Conversion Source Code Version 1

- Load `PythonExampleProblems\ListsAndDictionaries\DistanceConversion\distanceConversionVer1.py` into the `IDLE` text editor.

Distance Conversion Comments

- We declare an empty list using:

  `<LIST_NAME> = [].`

- We can add to a list using the append method:

  `append.<LIST_NAME>(<NEW_ITEM>).`

- We can access individual elements by "indexing" in to the list:

  `<LIST_NAME[<INDEX>].`

Run The Distance Conversion Source Code Version 1

Test Cases

Test Case	Expected result
0	[0,0,0]
1	[1,0,3]
4	[4,1,1]
50	[54,2,0]
100	[109,1,0]

Problem Example 6b: Distance Conversion Version 2 (Dictionaries)

UNIVERSITY OF LIVERPOOL

Distance Conversion Source Code Version 2

- Load `PythonExampleProblems/Selection/ListsAndDictionaries/DistanceConversion/distanceConversionVer2.py` into the `IDLE` text editor.

Distance Conversion Comments

- We declare an empty dictionary using:

```
<DICTIONARY_NAME> = {}
```

- We can add to a list using the append method:

```
<DICTIONARY_NAME>[<LABEL>] =
        <VALUE>
```

- We can access individual elements by "indexing" in to the list:

```
<DICTINARY_NAME[<LABEL>]
```

Run The Distance Conversion Source Code Version 2

Test Cases

Test Case	Expected result
0	[0,0,0]
1	[1,0,3]
4	[4,1,1]
50	[54,2,0]
100	[109,1,0]

Writing and Reading To and From Files

UNIVERSITY OF LIVERPOOL

Writing To Files

- We "open" a file using the statement:

```
<NAME> = open(<FILE_NAME>,<MODE>)
```

Frequently used modes are `'r'` (read only), `'w'` (write only) and `'a'` (append).

- Write to an opened file as follows:

```
<NAME>.write(<CONTENT>)
```

- And close the file at the end using:

```
<NAME>.close()
```

Writing to Files Example

- In a terminal window change directory to the directory `PythonExampleProblems\Temp` (in your directory structure).
- Open the python interpreter and type the following:

```
file = open('myfile.txt','w')
print file
file.write('I am really enjoying learning ')
file.write('about Python today\n')
file.close()
```

- Now open the file you have created in a text editor!

Reading From Files

- As before we "open" a file using the statement:

```
<NAME> = open(<FILE_NAME>,<MODE>)
```

- Read from an opened file as follows:

```
<CONTENT>=<NAME>.read()
```

- And close the file at the end as before:

```
<NAME>.close()
```

Writing to Files Example (1)

- Still in your `Temp` directory type the following in the Python interpreter:

```
file = open('myfile.txt','r')
text = file.read()
file.close()
print text
text.split()
```

Writing to Files Example (2)

- You should see something like:

```
>>> file = open('myfile.txt','r')
>>> text = file.read()
>>> print text
I am really enjoying learning
about Python today

>>> text.split()
['I', 'am', 'really', 'enjoying',
'learning', 'about', 'Python', 'today']
>>> file.close()
>>>
```

- Exit the Python interpreter

Problem Example 7: Landscape Gardening II, This Time With Dictionaries and File Output!

UNIVERSITY OF LIVERPOOL

Landscape Gardening II Introduction

- AQA GCSE Specimen Controlled Assessment example, Task 2:

"The Company has asked if it would be possible to save customer quotations so that these can be viewed at a later date. Create a section of the program that allows quotations to be saved ..."

Landscape Gardening II Source Code

- Load `PythonExampleProblems\FileHandling\LandscapeGardeningII\landsGardQuoteII.py` into the `IDLE` text editor.

Landscape Gardening II Comments (1)

- Data now stored as dictionaries.

```
MATERIAL_COST = {'Lawn' : 15.5,
'Patio' : 20.99, 'Water Feature' :
150.0}
```

```
quote = {'Lawn'  : {'Length' : 0,
'Width' : 0, 'Cost' : 0.0, 'Time' :
0.0}, 'Patio' : {'Length' : 0,
'Width' : 0, 'Cost' : 0.0, 'Time' :
0.0}, 'Water Feature' : {'Quantity' :
0, 'Cost' : 0.0, 'Time' : 0.0}}
```

Landscape Gardening II Comments (2)

- Note use of nested dictionaries.
- We access items in nested dictionaries as follows:

```
quote['Lawn']['Length']
```

- Now we have ability to write a quote to file.

Run The Landscape Gardening II Source Code

- In your `LandscapeGardeningII` directory check the file you have created!

Lunch Time?

UNIVERSITY OF LIVERPOOL