Programming Exercises 03: Data Types and Selection

Exercises 1 – Concept Revision in the Python Shell

Let's revise the concepts from this module by typing some simple statements into the **Python Shell**. Feel free to deviate from the workshop and experiment!

1. `type(-123)`
2. `type(3.14)`
3. `type('Example')`
4. `type(True)`

   The type() function tells you the data type of the value or variable that you pass it.

5. `int(3.14)`
6. `float(42)`
7. `str(123)`
8. `bool('False')`

   These functions are all used to convert data types. Remember, every language follows slightly different rules when it comes to such conversions – always test your code thoroughly!

9. `intVal = 10`
10. `strVal = '5'`
11. `intVal + strVal`
12. `intVal + int(strVal)`
13. `str(intVal) + strVal`

    These statements demonstrate combining an integer value and a numeric string – either by converting the string to an integer, or the integer to a string. The result is much different.

14. `8 // 5`
15. `8 % 5`

    You can use "//" if you want to do integer division, and "%" (known as the "modulo operator) to determine the remainder of a division. These can come in handy sometimes!

16. `'Fish' * 5`
17. `'10' * 5`

    While adding, subtracting or dividing a string doesn't make much sense, Python allows you to multiply strings – this can be useful, but also confusing if you have a numeric string.

    Try to determine the final True/False result of these boolean expresisons before typing them.

18. `False or False or not False and True`
19. `not(True and False) or False`

Exercises 2 – String Analyzing Program

Create a new file in the **Python Editor**. We will be creating a program that prompts the user to type something, and then uses if-then statements and "string methods" to analyse what they type.

String methods are built-in functions that *can only be used on* strings, allowing you to do things that either manipulate strings (e.g. changing to upper/lower case, removing whitespace, replacing parts of it, etc) or determine information about them (e.g. occurrences of a character, types of characters, length of string, etc). More information can be found in the string method documentation.

We will be focusing upon the string methods that determine information about the string – most of them begin with the word "is", e.g. `isalpha()`, `islower()` and `isnumeric()`. To use one of these, simply add it to the end of a string variable. e.g. If you have a string variable named `value` then `value.isalpha()` will return True or False depending on whether the string variable contains letters only.

Write a program that prompts the user to type something, and then goes through a series of if-then statements – each if-then statement will use a string method to check for something, and then print an appropriate message if the method returns True. Here is pseudocode of the program design:

Pseudocode

```
Prompt user to type something

If string meets certain criteria
    Print appropriate message

If string meets certain criteria
    Print appropriate message

...
```

Make your program check the string for at least four of the following:

Check	String Method
Is the string all upper case?	**`isupper()`**
Is the string all lower case?	**`islower()`**
Does the string contain letters only?	**`isalpha()`**
Does the string contain numbers only?	**`isdigit()`**
Does the string contain whitespace only?	**`isspace()`**
Does the string start with an "s"?	**`startswith()`**
Does the string end with a "t"	**`endswith()`**
Does the string contain at least two of the letter "e"?	**`count()`** (returns *number of matches*, not True or False)

Add another line to the end of your program, not related to the if-then statements, that prints the number of characters in the string (its length). To determine this, use the built-in `len()` function. Since this is not a string method, you give it the variable as a parameter, e.g. `len(value)`.

Exercises 3 – Enhancing a Conversion Program

Open one of your simple conversion programs from Workshop 1 in the **Python Editor**.
Modify it so that the program first prompts the user to choose whether they wish to convert from Metric to Imperial or from Imperial to Metric, e.g. kilometres to miles or miles to kilometres. Based on the user's response, prompt them for a value and perform the appropriate conversion as normal.

Here is pseudocode of the program design:

Pseudocode

```
Print options '1 for Metric to Imperial
              2 for Imperial to Metric'

Prompt user for choice

If choice == 1
    Prompt user for value in Metric
    Convert value to Imperial
    Print result

Else if choice == 2
    Prompt user for value in Imperial
    Convert value to Metric
    Print result

Else
    Print 'invalid choice' error message
```

Refer to Workshop 1 for the calculations needed to convert Metric/Imperial units.

Note that the "Prompt, Convert, Print" steps within this program design are essentially your original conversion program from Workshop 1.

For practise, try to draw a flowchart of this program design.

Exercises 4 – Enhancing the Grade Calculation Example

Create a new file in the Python Editor and copy the grade calculation code from the "Else-If Example" of Lecture 2 into it. Save the file and test the program to ensure that it works.

Now try to modify the program to incorporate the following criteria:

- ☐ After the prompt to enter a mark, a second prompt should ask the user whether or not they passed the exam, e.g. "Did you pass the exam? (y/n): "
- ☐ Assume that passing the exam is required in order to pass the unit. Hence, if the mark is greater than or equal to 50 but "n" was entered at the second prompt, the final grade should be "FI (Fail/Incomplete)"
- ☐ Otherwise, the code to calculate the grade works as per the example

You should only need to add a few new lines of code to the example to implement the new criteria.

It's very easy to make your code accommodate for users typing "N" instead of "n" to the second prompt by using the `lower()` function on the string containing their response to the prompt. It is up to you whether you convert their response to lower case when you obtain it, or just when you are comparing it to "n" in the boolean expression of an if-then statement.

Exercises 5 – Income Tax Program

The following table from the Australian Tax Office website describes the formula applied to calculate the amount of income tax that must be paid.

Taxable income	Tax on this income
0 – $18,200	Nil
$18,201 – $37,000	19c for each $1 over $18,200
$37,001 – $80,000	$3,572 plus 32.5c for each $1 over $37,000
$80,001 – $180,000	$17,547 plus 37c for each $1 over $80,000
$180,001 and over	$54,547 plus 45c for each $1 over $180,000

Based on this table, write a program that prompts the user to enter their income and then shows the amount of tax that they must pay. Before writing any code, use either (or both) pseudocode or a flowchart to design the program.