School of ICT

KIT100 Programming Preparation

Tutorial 4

Aims

- to practice reading through code;
- to review good programming principles;
- to become familiar with using operators; and
- to be able to engage in decision-making with Python.

Background

1. Reading through code

> ➢ Working in pairs read through the following piece of code. Your tutor will ask the class to discuss what you think the code should do and if the script follows good programming principles.

```
'''
Minutes to days to years conversion
Author: David Herbert
Version 1.1
Date: March 2016
Purpose: To convert an entered int read as minutes to days
and years
'''

# Obtain input
minutes = int(input("Enter the number of minutes: "))

numberOfDays = minutes // (24 * 60)
numberOfYears = numberOfDays // 365
numberOfActualDays = numberOfDays % 365

# Display results
print("%s minutes is approximately %s years and %s days" \
    % (minutes, numberOfYears, numberOfActualDays))
```

- What are the variables for this program?
  What are the operators?
  What are the executable statements for the program?

2. Review good programming principles

The creator of Python often commented that code is read a lot more than it is written. Therefore, we need to make sure that when we write our script files they following good programming principles.

Line length: All lines should be limited to a maximum of 79 characters (IDLE shows what position your cursor is if you click somewhere on a line). If the line length is over 79 characters you will need to wrap the text. An example:

```
if (width == 0 and height == 0 and
color == 'red' and emphasis == 'strong' or
highlight > 100):
```

Blank lines: Extra blank lines may be used (sparingly) to separate groups of related functions. Blank lines may be omitted between a bunch of related one-liners (e.g. a set of dummy implementations). Use blank lines in functions, sparingly, to indicate logical sections.

Imports: Imports should be on separate lines and near the top of the source file.

String Quotes: In Python, single-quoted strings and double-quoted strings are the same. The style guide does not make a recommendation for this. Pick a rule and stick to it. When a string *contains* single or double quote characters, however, use the other one to avoid backslashes in the string. It improves readability.
For triple-quoted strings, always use double quote characters to be consistent with the docstring convention in PEP 257 .

Naming Conventions: Functions should be lower case with an underscore to improve readability, i.e: function_name. Variables should also be named lower case with an underscore when necessary. If current code has used lowerCaseCaps then Python guidelines will allow you to follow on from this.

This is a very handy resource for making sure your script code meets the Python styling requirements: https://www.python.org/dev/peps/pep-0008/

Let's try a few examples:

- Which is correct?

```
import os
import sys
```

or

```
import sys, os
```

- Which is the correct use of whitespace?

```
spam( ham[ 1 ], { eggs: 2 } )
```

or

```
spam(ham[1], {eggs: 2})
```

- Which is the correct use of whitespace around operators?

```
i = i + 1
submitted += 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

or

```
i=i+1
submitted +=1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)
```

Hint:
If operators with different priorities are used, consider adding whitespace around the operators with the lowest priority(ies).

3. Review good programming principles

a. Consider the following program fragment:

```
c = 'Y'
s = "whatever"
i = 34
b = False
```

- Could the following values be used as the condition for an *if* statement — if **yes** what would the result be, if **no**, why not?

Expression	Valid?	Result/Reason
`c=!'2'`		
`c<'y'`		
`s=="whatever"`		
`s==s`		
`input()`		
`b`		
`b==False`		
`b!=True`		
`b=False`		
`len(s)==i`		
`"cat" + "dog"`		

b. Identifying appropriate conditional statements:

- Assuming the user has been asked for their name and has entered this into a character string variable name, for each of the following scenarios, identify which conditional statement should be used:

i. Display their name prefaced with the word “Hello”.
☐ if ☐ if-else ☐ if-elif ☐ nested if ☐ no conditional required

ii. Display the phrase “Thank you sir” if the name variable begins “Mr ”.
☐ if ☐ if-else ☐ if-elif ☐ nested if ☐ no conditional required

iii. Display the phrase “an odd name…” if the length of the name variable is odd and “an even name…” if the length is not odd.
☐ if ☐ if-else ☐ if-elif ☐ nested if ☐ no conditional required

iv. Display the word “two” if there are two characters in the name variable, “three” if the name variable is three characters long, “four” if of length four, and so on, up to “nine”. Only one message should be displayed.
☐ if ☐ if-else ☐ if-elif ☐ nested if ☐ no conditional required

v. Display the message “Not a valid name” if the user’s name starts with a number.
☐ if ☐ if-else ☐ if-elif ☐ nested if ☐ no conditional required

c. Practice your decision-making statements:

- Creating a new script file and following good programming principles, implement the above scenarios in task 3b.