BT furtHER Digital Intensive Programme

Module 02 - Python Fundamentals

Day 02

Recap day 01

Recap – Day 01:

- How to open spyder?
- How to write code in the python console?
- How to run .py code?
- What was the configuration setting for .py files?
- Zen of Python, how many you remembered?
- What does ** mean in python? And * ?
- What data types we learned on day 01?
- How to cast int to str?
- What to be aware of when writing python code?

Recap – Day 01: continued

- What is python good at?
- Difference between python 2 and 3?
- Cast int to float and vice versa?
- Rules for using print() function?
- What is OOP?
- Why OOP?

Module 2/ch02 folder

Open bash:

```
cd Desktop/module2/

mkdir ch02
```

why ch02 not ch2?

Day 02
Learning outcome

Learning Outcome - Curriculum Chapter 02:

- More on operations.
- What are variables?
  - Why have meaningful variable names?
  - Rules for naming variables
- More on strings
  - Simple string manipulation
  - String formatting
- Comments

More on operations

More on operations

Task 1. Use your Python file to print answers in your python console for the following maths problems

(Don't forget to put each statement on a new line! What will happen if you just copy and paste?):

```
5 - 6
8 * 9
6 / 2
5 / 2
5.0 / 2
5 % 2
2 * (10 + 3)
2 ** 4
```

More on operations

Task 1. tips/questions

- What name have you given your .py file?
- Which folder have you saved it in?
- How to print the answers that you chose to test?
- Can you try some variations of those operation? Be creative!
- It doesn't matter if you want to leave spaces between numbers and operators, just be consistent!

e.g. 2*(3-10) and 2 * (3 - 10) is the same

What are variables?

Introduce variables

Can anyone explain what variables are?

Can anyone explain what variables are?

E.g. A = 2 + 3

Name ▲	Type	Size	Value
A	int	1	5

E.g. A = 5 + 5

Name ▲	Type	Size	Value
A	int	1	10

Variables are used to store values, however the value can be changed!

Introduce variables- Advantages:

Variables are particularly helpful when you write code where you may need to adjust values afterwards.

When using variables you can change a variable once rather than change the value everywhere in your program.

Introduce variables- Advantages:

```
x = 50
A = 5*x + 4
B = 6*x + 3
C = 2/x - 4
```

A	int	1	254
B	int	1	303
C	float	1	-3.96
x	int	1	50

```
x = 2
A = 5*x + 4
B = 6*x + 3
C = 2/x - 4
```

Name ▲	Type	Size	
A	int	1	14
B	int	1	15
C	float	1	-3.0
x	int	1	2

Introduce variables- Advantages:

If you use a value frequently:

e.g. in a ATM scenario, both withdrawing cash and topping-up a mobile requires the balance value.

It is better to set the balance as a variable and do the calculation, so whenever the balance changes, you just need to update the number with the balance variable rather than go to each withdraw and top-up formula and update the number. (note do the calculate after change variables! -python does not use default global variable)

Introduce variables -Syntax:

We use “=” to assign a value to a variable (whose name appears to the left of =)

E.g. birds = 3

This is not an equal sign and we will come back to that.

Introduce variables:

Convert the code below in your .py file and print the x, y values. See if you get the same as below.

```
>>> x = 12.5
>>> x
12.5
>>> y = 3 * x + 2
>>> y
39.5
>>> x = x + 1.1
>>> x
13.6
```

Variable names?

Variable - Meaningful names:

"Many people can write machine readable code, not many can write human readable code!"

Variable - Meaningful names:

Zen of Python:

- Beautiful is better than ugly
- Explicit is better than implicit

‘balance’ or ‘birds’ is better than ‘x’, ‘y’

-- gives semantic meaning.

-- readable and understandable.

-- helps yourself and co-workers to understand your code!

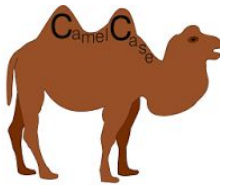
Variable - rules for naming:

Although it needs to be meaningful, but...

- It cannot be a python keyword: e.g. for, python, def, break, continue, etc
- It will change to a different colour if it is a keyword.

Variable names can contain letters, digits and underscores, but...

- It has to start with a letter, preferably a lowercase letter.
- No spaces please! But you can separate using underscores or camelcase: e.g. a_long_name or aLongName



Variable - can be assigned with any data type:

You can assign a variable with different data types:

age = 5

age = "almost three"

age = 29.5

age = ‘I really don’t know!’

If you do print(age), what is the result now? Get an error!!!

Variable - homework

Homework, read clean code chapter 2 meaningful names and summaries how you going to give names to variables, functions and also classes!

More on strings

Single and double quote

Be careful when you use single quote:

```
In [9]: age = 'I really don't know'
  File "<ipython-input-9-35f0cf3998ee>", line 1
    age = 'I really don't know'
                        ^
SyntaxError: invalid syntax
```

Or try: age = ‘I really don\’t know!’

Or using double quote: “I really don’t know!”

Single and double quote

Both single or double quotes are fine, but:

```
In [13]: age = "information age'
  File "<ipython-input-13-b43fb56c73c3>", line 1
    age = "information age'
                          ^
SyntaxError: EOL while scanning string literal
```

-- > Please be consistent!

STR?

```
In [11]: type(age)
Out[11]: str
```

You can use the **type()** function to test what is the data type inside a variable.

and ‘str’ means ‘string’ in python.

You also can use type() directly with data input, e.g

```
In [14]: type(3.5)
Out[14]: float
```

String operation

Task 3, try running the code below with an edited file.

```
print (‘hello’ + ‘world’)

print ("Joke " * 3)
print ("Chen" + 3)
print ("hello".upper())
print ("GOODBYE".lower())
print ("the lord of the rings".title())
```

Are the answers what you expected to see?

String operation

How about assigning variables as well?

S1 = ‘hello’ + ‘world’

S2 = "Joke " * 3

S3 = 5

What is the result of s1 + s2*10?

How about s1 + s2 + s3?

Learn to read error messages

How about s1 + s2 + s3?

```
In [24]: S1+S2+S3
Traceback (most recent call last):

  File "<ipython-input-24-7bf1e5a83d0e>", line 1, in <module>
    S1+S2+S3

TypeError: must be str, not int
```

What you have learned from the error message?

And how are you going to improve your code?

Str object methods

Have a look at your task 3 code file. What functions have you just used?

Str object methods

.upper() .lower() .title()

We often call a function a method as well.

As mentioned on day 01, the string data class has string object e.g. ‘hello’.

When using a method that can **only be used by a particular object**, we must connect that method with that object by prefixing it with a ‘**.**’

-- So if the object and method do not match, e.g. 5.lower() you will get an error.

String formatting

String formatting -intro

String formatting is a way of taking one or more variables (with same or different data types!) and putting them inside a string.

This is a very useful technique that is not only limited to Python (we will mention it again in SQL and web security).

String formatting - example

Let's use an example of a 5 year old who likes to paint. We need two variables to hold this information:

```
age = 5
like = "painting"
```

How do we print this information in a sentence that looks nice? There are a few different ways we can do this:

```
age_description = "My age is {} and I like {}.".format(age, like)
```

String formatting - example

What function/method you have just learned for string formatting?

.format(variable_0, variable_1, etc)

So we could also do it this way:

age_description = "My age is {0} and I like {1}.".format(age, like)

Try the code in your .py file and print(age_description) to see the result!

String formatting - python 2

You may have already noticed that with the format() method we use {} as a placeholder for the input variables.

However, in python 2, you may see %s, %d, %r which means string, digit, and repeat previous.

SQL (a database related coding language that we will learn later) also uses specific variable placeholders for specific data types too, as it is security related.

We don't cover this at the moment, but feel free to explore if it interests you.

Comments in python

Comments -

In Python, any part of a line that comes after a # is ignored (been commented out).

You can use it for:

- Giving titles/sections for groups of code.
- Explaining some complicated lines.
- Testing different versions of a method, by commenting out one and running the other, rather than having to delete and rewrite it each time.
    - greeting = "Hello World!"
    - #greeting.upper() *#don't delete this line yet*
      greeting.lower()

Comments - """ """

You may have also noticed, at the beginning of your .py file, spyder will auto generate some logs:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 31 16:50:22 2018

@author: wuchenhao
"""
```

→ for bash to notice this is a python file, but python will ignore it

→ """ file instruction or log date/author info """

Things inside """ """" is another type of comment for when you want to write more info, e.g. a to do list, how to solve error and other logs

Comments - homework

Homework, read clean code chapter 4 Comments and summaries how you going to write comments!

Homework and assessments

Home work

Please read through and practise all tasks in ch02.

Try the homework at the end of chapter 02.

Submit ch2_name.py to your git repo.

Preview module2curriculum: chapters 3-6.

Read clean code chapter 2 (meaningful names), and 4 (comments), and **summaries how you going to use them**!

Q / A?