SIT 112 - Data Science Concepts

Lecturer: Truyen Tran | truyen.tran@deakin.edu.au
Assistant:

School of Information Technology,
Deakin University, VIC 3216, Australia.

Practical Session 1.2

The purpose of this session is twofold:

1. to give you the instruction to install Anaconda.
2. to revise key features of Python language and allow you the time to practice basic coding in python with IPython

At the end of this practical session, you should be able to:

- know how to install Anaconda and work with interactive python using IPython
- know the basic features of python, including arithmetic operations, fundamental data types and variables

Please note that:

- This unit is *not* about teaching you how to program in Python. You are responsible for developing your own programming skills. There are many excellent resources to learn python online as well as the recommended textbooks presented in the lecture. Developing skills in programming can take time, and let us remind ourselves:

  > "... even though we can speak and write English, it does not necessarily mean we can write a best-selling novel" -- Unknown.

- You are assumed to have a basic knowledge of programming.
- This session only recaps main features of python.

Part I: Anaconda and Installation

- 1-instructions.pdf gives you the step by step instruction on how to install Anaconda on your machine and an introduction to Python interpreter.

Part II: Review of Python Language

1. "Hello World!"

Instruction: Launch IPython by running the command `ipython` in your terminal and enter the following commands:

In [ ]:
```
print("Hello World!!")
print("Welcome to SIT 112 Data Science Concepts")
```

Note: the difference in the syntax. Python 3.x uses a slightly different syntax from 2.x version. This course will use python 3.6.

---

2. Commenting

Instruction: Type these lines below. What was displayed?

In [ ]:
```
# This is a comment.
# It will not do anything.
# Comments are to clarify code and are not interpreted by Python
print("Hello World!!") # this is a greeting
```

Exercise: Print several sentences and add a comment after each print command.

---

3. Variables and types

Variable names in Python can contain alphanumerical characters a-z, A-Z, 0-9 and some special characters such as _. Normal variable names must start with a letter. By convension, variable names start with a lower-case letter, and Class names start with a capital letter.

3.1 Assignment

The assignment operator in Python is `'='`. Python is a dynamically typed language, so you do not need to specify the type of the variable while creating it.

Assigning a value to a new variable creates the variable:

Instruction: Execute the lines below using IPython to see the output.

In [ ]:
```
# variable assignment

x = 10
y = "I am a string"

print("x = ", x)
print("y = {}".format(y))
```

Although not explicitly specified, a variable does have a type associated with it. The type is derived from the value that was assigned to it.

In [ ]:
```
type(y)
```

Exercise: Write a piece of code that declares a variable called `'my_var'` and assign the value 25 to it. Then print its value.

3.2 Re-assignment

We can update a variable by assigning a new value to that same variable name, which will change the value.

Instruction: Execute the lines below to see the output

In [ ]:
```
x = 10
x = 12
x = x + 2
print(x)
```

3.2 Fundamental types

In [ ]:
```
# integers
x = 1
type(x)
```

In [ ]:
```
# float
x = 1.0
type(x)
```

In [ ]:
```
# boolean
b1 = True
b2 = False

type(b1)
```

In [ ]:
```
# complex numbers: note the use of `j` to specify the imaginary part
x = 1.0 - 1.0j
type(x)
```

Exercise: Write a piece of code that declares a variable called `'my_float'` and assign the value 25 to it. Then print its value and type. What is its type? What do you need to do to make it a float instead of int?

---

4. Operators

4.1 Arithmetic operators

- addition: x + y
- subtraction: x - y
- multiplication: x*y
- division: x/y
- floor division: x//y (round the answer to the near integer)
- modulus: x%y (remainder of x divided by y)
- exponent: x**y (raise x to the power of y)

In [ ]:
```
x = 23
y = 4
print ('x + y = ', x+y)
print ('x - y = ', x-y)
print ('x * y = ', x*y)
print ('x / y = ', x/y)
```

Note: This is Python 3 code. The previous versions of Python behave differently - the division of one integer by another in Python 2 will yield an int, even if the expected result is not a whole number! This Python 2 behaviour is like integer division (//) in Python 3. We use Python 3 in this course.

In [ ]:
```
print('x // y = ', x//y)
print('x % y = ', x%y)
print('x ^ y = ', x**y)
```

Exercise: Try at least one example for each operator. Write a comment before the example to explain which operator you are entering.

4.2 Boolean operators

The boolean operators are spelled out as words `and`, `not`, `or`. Try the lines below:

In [ ]:
```
True and False
```

In [ ]:
```
not False
```

In [ ]:
```
True or False
```

4.3 Comparison operators

Comparison operators >, <, >= (greater or equal), <= (less or equal), == equality, and is identical

In [ ]:
```
2 > 1, 2 < 1
```

In [ ]:
```
2 > 2, 2 < 2
```

In [ ]:
```
2 >= 2, 2 <= 2
```