



COS10022 – DATA SCIENCE PRINCIPLES

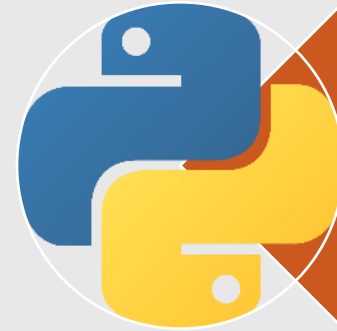
Dr Pei-Wei Tsai (Lecturer, Unit Convenor)
ptsai@swin.edu.au, EN508d

SWIN
BUR
NE

SWINBURNE
UNIVERSITY OF
TECHNOLOGY



Week 01



Topic:

- Basic Python Programming



Tool:

- Google Colab



BASIC OF PYTHON PROGRAMMING

Useful Online Resource

- CS Dojo
 - https://www.youtube.com/watch?v=Z1Yd7upQsXY&list=PLBZBJbE_rGRWeh5mIBhD-hhDwSEDxogDg
 - CS Dojo YouTube channel has a series of Python programming course for absolute beginners.
 - The key points of Python is introduced in the series in detail.





colab.research.google.com ▼

Google Colab

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your ...

You've visited this page many times. Last visit: 24/02/21

Introduction to Colab and Python Colaboratory

Welcome to this Colab where you will get a quick introduction to ...

Colaboratory, or "Colab" for short, is a product from Google ...

Google Drive

Uploading files from your local file system · Downloading files to ...

Colab Pro

Colab Pro · Get more from Colab · Faster GPUs · Longer runtimes ...

Tensorflow with GPU

In this notebook you will connect to a GPU, and then run some ...


Using Google Colab with GitHub

Colab can load public github notebooks directly, with no ...

[More results from google.com »](#)

Login to Google Colab

- Register a Google account if you do not have one.
- Download the .ipynb file from Canvas and upload the file to Google Colab.

 + Code + Text

Let's create the first comment in your program.



```
[ ] print(5)
```



```
[ ] print("I'm learning Python Programming.")
```

```
[ ] a = 1
    b = 6
    print(a)
    print(b)
    print(a,b)
```

```
[ ] a = b
    b = "It's new!"
    print(a,b)
```

Edit and Execute

- Add cells by clicking the “+ Code”.
- Click on the play icon to execute the cell.

Write Comments in Your Code

- Try the comment symbol to leave comments in your code.
 - The comment is commonly used in the programs for helping programmers to remember what is going on in the code.
 - A more import fact is that the comments help other programmers in the team to understand your code to boost up the efficiency of cooperation and team works.
- Tip: Try to write a sentence with the symbol # as the start and press “Shift” + “Enter” to execute the codes in the cell.

```
In [39]: # Let's create the first comment in your program.
```

Displaying the Result from the Code

- The basic function to use is *print()*
- It prints whatever in a given contain between the brackets.

- **Practice**

- Try to print a number 5.

```
In [7]: print(5)  
5
```

- **Practice**

- Try to print the sentence: I'm learning Python programming.

```
In [6]: print("I'm learning Python Programming.")  
I'm learning Python Programming.
```


Variable

A variable is a container, which can be used to temporary store data.

In Python, you don't need to declare the type of variable before using it.

A variable can be used to represent a single value, a vector, or a dictionary (array).

The value inside a variable can be changed anytime.

Set Variables and Print the Values

- Let's create 2 variables called *a* and *b* and assign values 1 and 6 to them, accordingly.
- Try to print *a* and *b* independently and then try to print them in the same line.

```
In [13]: a = 1
         b = 6
         print(a)
         print(b)
         print(a,b)
```

```
1
6
1 6
```

- We can reassign value to any of the variable.
 - Let's try to assign the value stored in *b* to *a* and change *b* to store a string "It's new!"
 - Print the final results of *a* and *b*.

```
In [14]: a = b
         b = "It's new!"
         print(a,b)
```

```
6 It's new!
```

Brainstorm Time!

Assume we have two variables, a and b , holding two strings “Alpha” and “Beta,” respectively.

What are you going to do if we want to make a swap to let variable a hold “Beta” and variable b hold “Alpha?”



Brainstorm Time!

```
In [22]: # Initial State
a = "Alpha"
b = 'Beta'
print("Before:")
print("a =", a, "\nb =", b)

# This is how you're going to do it.
temp = a
a = b
b = temp
print("\nAfter:")
print("a =", a, "\nb =", b)
```

Before:
a = Alpha
b = Beta

After:
a = Beta
b = Alpha



Mathematic Operators

- **Relationships:**
 - **>** Greater
 - **<** Smaller
 - **==** Equal to
 - **>=** Greater and equal to
 - **<=** Smaller and equal to
 - **!=** Not equal to

Flow Control: If-Else

In many situations, you'll need to control your program to do one thing if a certain criterion appears, otherwise, your program needs to achieve another operation when that criterion doesn't exist.

If-Else is a flow control description to help you change to execution structure of your code.

You need to “indent” the description under the criterion description to let Python know that this part of the code belongs to the criterion corresponding situation.

If-Else

- Expression:
 - *if criterion₁ relation criterion₂:*
descriptions₁
else:
descriptions₂
- Example:

```
In [24]: a = 2
         b = 3
         if a < b:
             print("a is less than b.")
         else:
             print("a is greater than or equal to b.")

a is less than b.
```

```
In [25]: a = 5
         b = 3
         if a < b:
             print("a is less than b.")
         else:
             print("a is greater than or equal to b.")

a is greater than or equal to b.
```

Brainstorm Time!

Assume you have 10 dollars on your hand.

A set of hamburger causes 12 dollars.

Write a program using If-Else to check whether you can afford to have a hamburger set.



Brainstorm Time!

```
In [27]: # Brainstorm 2

# initial values
x = 10
burger = 12

# main program
if x >= burger:
    print('You can afford to have a burger meal.')
else:
    print("The burger set is too expensive. You can't afford it.")

The burger set is too expensive. You can't afford it.
```



Brainstorm Time!

Assume you have 10 dollars on your hand.

A set of hamburger causes 12 dollars while a set of hotdog causes 8 dollars.

Write a program using If-Else to check what meal you can get.



Brainstorm Time!

```
In [32]: # Brainstorm 3

# initial values
x = 10
burger = 12
hotdog = 8

# main program
if x >= burger:
    if x >= hotdog:
        print("You can afford choose either a burger or a hotdog set as the meal.")
    else:
        print("You can only afford to have a burger set.")
else:
    if x >= hotdog:
        print("You can only afford to have a hotdog set.")
    else:
        print("You cannot afford either a burger or a hotdog set as the meal.")
```

You can only afford to have a hotdog set.



If-Elif-Else

- Expression:

- *if* *criterion*₁ *relation* *criterion*₂:
 descriptions₁
- *elif* *criterion*₃ *relation* *criterion*₄:
 descriptions₂
- *else*:
 descriptions₃

- Example:

```
In [38]: # The three cases example

# Set input variables
a = 2
b = 3

# Main program
if a < b:
    print("a is less than b.")
elif a == b:
    print("a is equal to b.")
else:
    print("a is greater than b.")

a is less than b.
```

Functions

Function can be treated as a collection of instructions.

Function can also be treated as a collection of codes.

The purpose of using function is to put the repeatable part of codes in an independent place and reuse in the future without rewrite them again.

Using functions can also help you to make your code more tidy.

Function

- `def function_name(input_vars):`
descriptions_in_the_function
- To use the function, simply call the function in the code.
- Example:

In [46]: # Functions

```
def my_first_function():  
    print("This line is inside the function")  
    print("This line is also inside the function")  
  
print("This line is outside of the function")
```

This line is outside of the function

In [48]: # Functions

```
def my_first_function():  
    print("This line is inside the function")  
    print("This line is also inside the function")  
  
print("This line is outside of the function")  
my_first_function()
```

This line is outside of the function
This line is inside the function
This line is also inside the function

Function

- Passing a variable into the function and use it.
- Example:

```
In [49]: # Passing variable(s) into functions

def function2(var1):
    print("The variable sent to the function is", var1)

print("This line is outside of the function")
myvar = 5
function2(myvar)
```

```
This line is outside of the function
The variable sent to the function is 5
```

Return Value from a Function

- *def function_name(input_vars):*
descriptions_in_the_function
return a_variable
- To use the function, simply call the function in the code and assign the function to a variable.

```
In [52]: # Return value from a function

def function3(var1):
    print("The variable sent to the function is", var1)
    return 3 * var1 + 2

print("This line is outside of the function")
myvar = 5
result = function3(myvar)
print(result)
```

```
This line is outside of the function
The variable sent to the function is 5
17
```

Passing and Returning Multiple Values from a Function

```
In [56]: # Passing and returning multiple values from a function
```

```
def function3(var1, var2):  
    return (3 * var1 + 2), (-1 * var2)  
  
print("This line is outside of the function")  
myvar = 5  
[result, res2] = function3(myvar, (myvar - 2))  
print(result, res2)
```

```
This line is outside of the function  
17 -3
```



Exercise 1 – Choosing Available Rooms

- Assume we have three class rooms called RoomA, Room B, and RoomC with capacities of 20, 50, and 120, respectively.
- Only 1 room can be used to contain all students.
- The student enrolment number is 48.
- Use *If-Else* and/or *If-Elif-Else* structure to recommend the available classroom.
- Use a function to handle the output of available recommendations.