

## Programming for Data Science G (11521 Online & On-campus)

### Week 3 Tutorial

### Control Flow and Functions

#### Objectives

- To practise conditional statements
- To practise function implementation
- To apply operators, expressions, and control flow to function implementation

#### Use Visual Studio or PyCharm or Spyder (in Anaconda) to Create a New Python Project

- Use your own computer or login to UC Student Virtual Desktop  
<https://frame.nutanix.com/university-of-canberra/ditm/uc-remote-access-student/launchpad/uc-virtual-desktop-student>
- Open Visual Studio or PyCharm or Spyder (in Anaconda) to create a new project and name it **Week3Tutorial**.
- Open **Week3Tutorial.py** file to practise Python functions in this file.

## 1. Conditional Statements

### The if statement

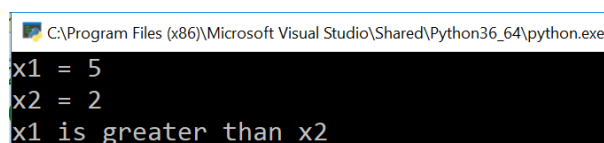
#### Syntax

```
if condition:
    command
elif condition:
    command
else:
    command
```

Examples of *condition*:  $a == b$ ,  $a != b$ ,  $a > b$ ,  $a >= b$ ,  $a < b$ ,  $a <= b$

- **Example 1:** Enter the following Python code to Week3Tutorial.py and run it.

```
x1 = 5
print(f'x1 = {x1}')
x2 = 2
print(f'x2 = {x2}')
if x1 == x2:
    print('x1 equals x2')
elif x1 < x2:
    print('x1 is less than x2')
else:
    print('x1 is greater than x2')
```



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
x1 = 5
x2 = 2
x1 is greater than x2
```

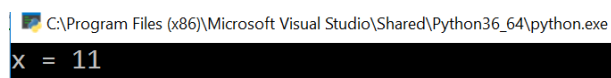
## The while statement

### Syntax

```
while condition: #condition is true
    command
else: #optional, condition is false
    command
```

- **Example 2:** Enter the following Python code to Week3Tutorial.py and run it.

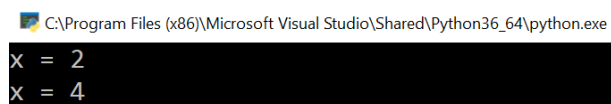
```
x = 5
while x <= 10:
    x += 1
else:
    print(f'x = {x}')
```



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
x = 11
```

- **Example 3:** Enter the following Python code to Week3Tutorial.py and run it.

```
x = 0
while x <= 3:
    x += 2
    print(f'x = {x}')
```



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
x = 2
x = 4
```

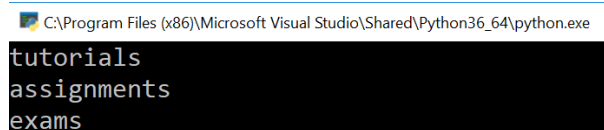
## The for..in statement

### Syntax

```
for variable in sequence:
    command
else: #optional
    command
```

- **Example 4:** Enter the following Python code to Week3Tutorial.py and run it.

```
assessments = ['tutorials', 'assignments', 'exams']
for ass in assessments:
    print(ass)
```



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
tutorials
assignments
exams
```

In this example, `assessments` is an array. You will learn more later

- **Example 5:** Enter the following Python code to Week3Tutorial.py and run it.

```
for c in 'data': #string data is a sequence of characters
    print(c)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
d
a
t
a
```

### The **break** statement

The **break** statement is to stop a loop even if the condition is true.

- **Example 6:** Enter the following Python code to Week3Tutorial.py and run it.

```
while True:
    x = int(input('Enter a number : '))
    if x == 0:
        break
    print('The number is', x)
print('Done')
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Enter a number : 3
The number is 3
Enter a number : 0
Done
```

In this example, the input number from `input('Enter a number : ')` is of type string, the `int()` function is to convert this string to integer.

### The **continue** statement

The **continue** statement is to stop the current iteration and continue with the next iteration.

- **Example 7:** Enter the following Python code to Week3Tutorial.py and run it.

```
while True:
    x = int(input('Enter a positive number : '))
    if x == 0:
        break
    elif x < 0:
        print('The number must be positive')
        continue
    print('The number is', x)
print('Done')
```

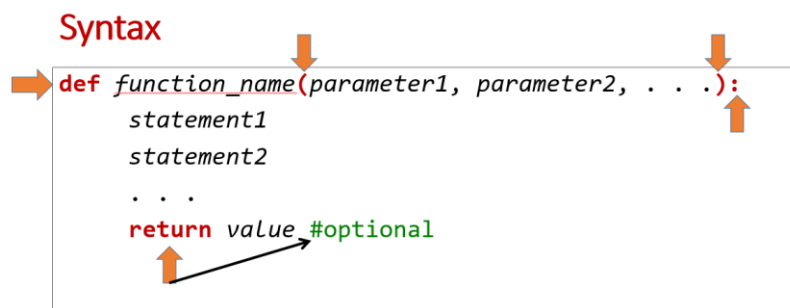
```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Enter a positive number : 8
The number is 8
Enter a positive number : -3
The number must be positive
Enter a positive number : 3
The number is 3
Enter a positive number : 0
Done
```

**Question:** Write a Python program that asks the user to enter a number then outputs the square of that number and asks if the user wants to continue. The program repeats the above process if the user enters 'y' or outputs 'Done' then stops if the user enters 'n'. For example,

```
Enter a number: 3
The square of 3 is 9
Continue? (y/n) y
Enter a number: 5
The square of 5 is 25
Continue? (y/n) n
Done
```

Please let your tutor see your completed tutorial before you submit this project on Canvas.

## 2. Functions



- **Example 8:** Function that has no input parameter and returns no value. Enter the following to Week3Tutorial.py

```
#Example 1
#define function
def welcome():
    print('Welcome to Python functions')
#end of function

#call function
welcome()
```

You can see from the above function implementation, the **welcome** function is to print out the welcome message. It does not require any input parameter and does not return anything.

The function call **welcome()** will call the welcome function to do its task (printing out the welcome message). If you do not call the function, the program will do nothing although the program has the welcome function in it.

Save then run the program. The output would be

```
Welcome to Python functions
```

If you call the function twice as below

```
#call function
welcome()
welcome()
```

You will have two messages output on the screen as follows

```
Welcome to Python functions
Welcome to Python functions
```

- **Example 9:** Function that has an input parameter and returns no value.  
Comment out the code in Example 8 then enter the following to Week3Tutorial.py

```
#Example 2
#define function
def square(x):
    print('Square of ' + str(x) + ' is ' + str(x*x))
#end of function

#call function
square(5)
```

This **square** function inputs  $x$  then prints out the square of  $x$ . The function call **square(5)** will assign 5 to  $x$  then call the square function to perform  $x*x$  then prints out its value.

Save then run the program. The output would be

```
Square of 5 is 25
```

- **Example 10:** Function that inputs parameters and returns a value.

```
#Example 3
#define function
def square(x):
    return x * x
#end of function

#call function
s = square(5)
print('Square of 5 is ' + str(s))
```

This **square** function inputs  $x$  then returns  $x*x$ . The function call **square(5)** will assign 5 to  $x$  then call the **square** function to perform  $x*x$  to have 25 then returns this value. The assignment (=) in **s = square(5)** will assign 25 (returned by **square(5)**) to  $s$ . The next statement will convert 25 in  $s$  to string using **str(s)** then call the **print** function to output the result.

Save then run the program. The output would be

```
Square of 5 is 25
```

- **Example 11:** Function that calculates distance  $d$  between 2 points  $(x_1, y_1)$  and  $(x_2, y_2)$ . In maths, the Euclidean distance is defined as

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad \text{or} \quad d = [(x_2 - x_1)^2 + (y_2 - y_1)^2]^{1/2}$$

We use the second one to implement the function (the first one needs **sqrt** function from **math** library).

```
#Example 11
#distance d = square root of (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1)
#define function
def distance(x1, y1, x2, y2):
    d = ((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1))**0.5
    return d
#end function

#call function
x1 = 3
y1 = 0
x2 = 0
y2 = 4
dist = distance(x1, y1, x2, y2)
print('Distance between (3, 0) and (0, 4) is ' + str(dist))
```

Save then run the program. The output would be

```
Distance between (3, 0) and (0, 4) is 5.0
```

- **Example 12:** Same function as that in Example 11. However, the input parameters are 2 tuples (we will learn tuple later) which are  $(x_1, y_1)$  and  $(x_2, y_2)$ . In the function implementation below:
  - The built-in function **len(c)** returns the number of items in **container c**.
  - The built-in function **range(m)** returns m integers ranging from 0 to m. For example, range(5) returns 5 integers which are 0, 1, 2, 3, and 4.
  - The for statement will assign each integer from range(len(p1)) to *i*. If range(len(p1)) returns 0 and 1, then *i* will be 0 then 1.
  - If  $p_1 = (x_1, y_1)$  then  $p_1[0] = x_1$  and  $p_1[1] = y_1$

```
#Example 5
#distance d = square root of (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1)
#Two points are two tuples p1 = (x1, y1) and p2 = (x2, y2)
#define function
def distance(p1, p2):
    d = 0
    for i in range(len(p1)):
        d += (p2[i] - p1[i]) * (p2[i] - p1[i])
    d = d**0.5
    return d
#end function
#call function
#Two points are two tuples (x, y)
p1 = (3, 0)
p2 = (0, 4)
dist = distance(p1, p2)
print('Distance between p1 and p2 is ' + str(dist))
```

Save then run the program. The output would be

```
Distance between p1 and p2 is 5.0
```

Why do we need the *for* statement and *tuple*? These will help us extend the data points from 2 dimensions to multiple dimensions without changing function implementation. We always see multi-dimensional data points in real-world data sets. For example, you can change p1 and p2 in the above program to the following

```
p1 = (3, 0, 0, 0)
p2 = (0, 4, 0, 0)
```

Save then run the program. There is **no error** found and the output is the same,

### 3. Please complete the questions below yourself. You can ask your tutor for help.

- **Question 1:** Write a function that inputs total mark and outputs grade with the following rules

```
total mark < 50: grade = Fail
50 ≤ total mark < 65: grade = P
65 ≤ total mark < 75: grade = CR
75 ≤ total mark < 85: grade = DI
total mark ≥ 85: grade = HD
```

- **Question 2:** Write a Python program that 1) asks the user to enter total mark, 2) calls the function in Question 1 with total mark as input, and 3) outputs the grade from the function to the user. For example,

```
Enter your total mark: 88
Your grade is HD
```

- **Question 3:** Modify the Python program in Question 2 to repeat the above three tasks until the user enters a negative number. Hint: use the while statement. For example,

```
Enter your total mark: 44
Your grade is Fail
Enter your total mark: 55
Your grade is P
Enter your total mark: 66
Your grade is CR
Enter your total mark: 77
Your grade is DI
Enter your total mark: 88
Your grade is HD
Enter your total mark: -1
```

- **Question 4:** Use **VarArgs parameters** seen in lecture, write a function that can find the maximum of a sequence of *positive* numbers as input arguments. For example,

```
#Question 4:
#define function
#write your function find_max here

#end of function
```

```
max = find_max(2, 4, 8, 3, 1, 33, 25, 65)
```

```
Maximum value is 65
```

```
print('Maximum value is ' + str(max))
```

```
max = find_max(36, 52, 65)
```

```
print('Maximum value is ' + str(max))
```

```
Maximum value is 65
```

#### 4. Drawings

- You will write a Python program that displays a canvas and draws lines and circles on this canvas. This tutorial is to prepare for Assignment 1.
- Add the following to your program to import the library for graphics

```
from tkinter import *  
top = Tk()
```

```
#add code in next steps below this line
```

```
C.pack()  
top.mainloop()
```

- To create a canvas for drawings, add the following code between `top = Tk()` and `C.pack()` as seen above:

```
C = Canvas(top, bg="white", height=700, width=700)
```

- To draw a line between two points (x1, y1) and (x2, y2), add the following code below the line for Canvas

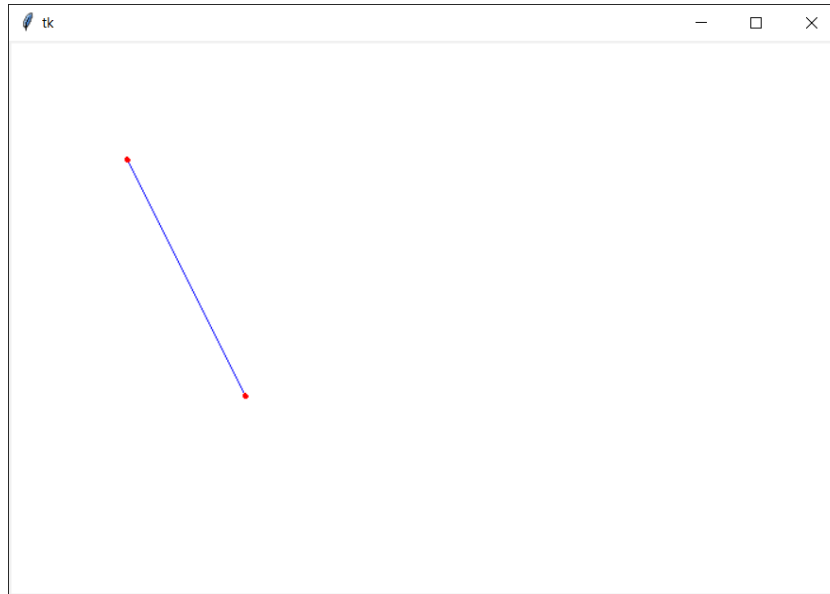
```
x1 = 100  
y1 = 100  
x2 = 200  
y2 = 300  
C.create_line(x1, y1, x2, y2, fill = "blue")
```

- To draw a circle at point (x1, y1) with radius 2, add the following

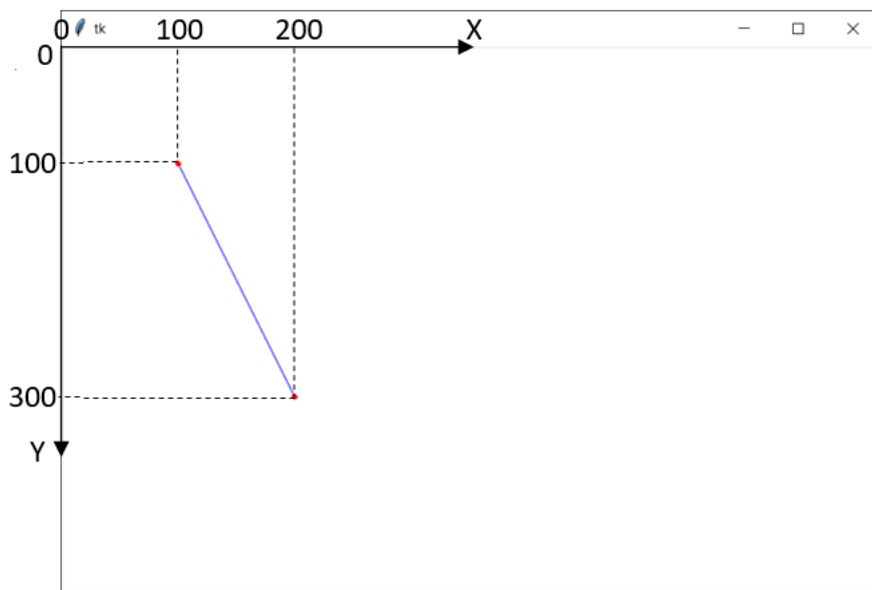
```
C.create_oval(x1-2, y1-2, x1+2, y1+2, outline = "red", fill="red")  
C.create_oval(x2-2, y2-2, x2+2, y2+2, outline = "red", fill="red")
```

- Save then run your program to see the output:





- The coordinates (X, Y) below explain how the program locates 2 points and draw the line between them.



Please let your tutor see your completed tutorial.