University of Canberra

Faculty of Science and Technology

**Programming for Data Science G (11521)**

**Week 3 Tutorial**

**Functions**

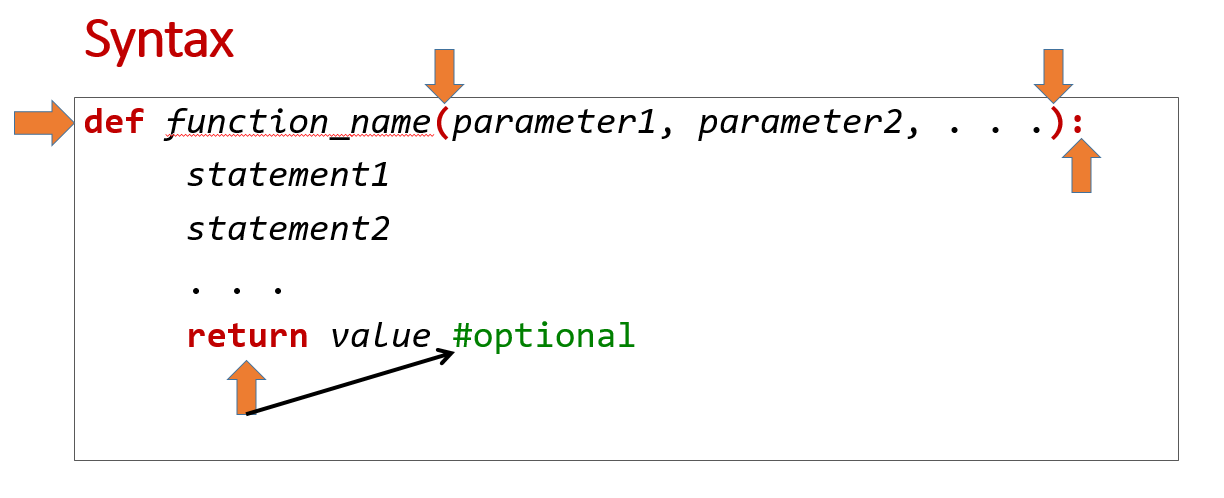
**Objectives**

* 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.

**Functions**



* **Example 1**: 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



If you call the function twice as below

#call function

welcome()

welcome()

You will have two messages output on the screen as follows



* **Example 2**: Function that has an input parameter and returns no value.

Comment out the code in Example 1 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



* **Example 3**: Function that inputs parameters and returns a value.

Comment out the code in Example 1 then enter the following to Week3Tutorial.py

#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



* **Example 4**: Function that calculates distance *d* between 2 points . In maths, the Euclidean distance is defined as

or

We use the second one to implement the function (the first one needs **sqrt** function from **math** library, and we will try it next week.

Comment out the code in Example 1 then enter the following to Week3Tutorial.py

#Example 4

#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



* **Example 5**: Same function as that in Example 4. However, the input parameters are 2 tuples (we will learn tuple later) which are . 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 p1 = (x1, y1) then p1[0] = x1 and p1[1] = y1

Comment out the code in Example 1 then enter the following to Week3Tutorial.py

#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



*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,

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

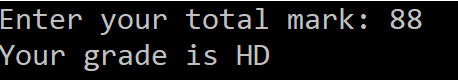
50 < total mark < 65: grade = P

65 < total mark < 75: grade = CR

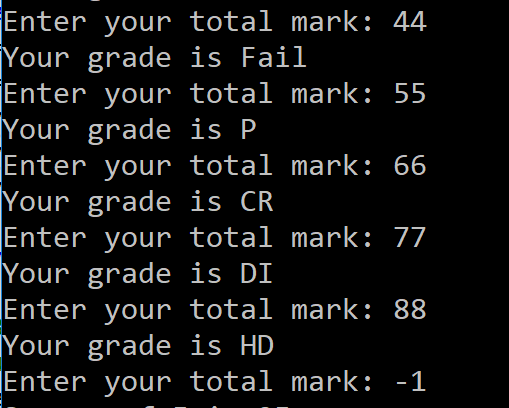
75 < total mark < 85: grade = DI

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,



* **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,



* **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)

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

max = find\_max(36, 52, 65)

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

**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

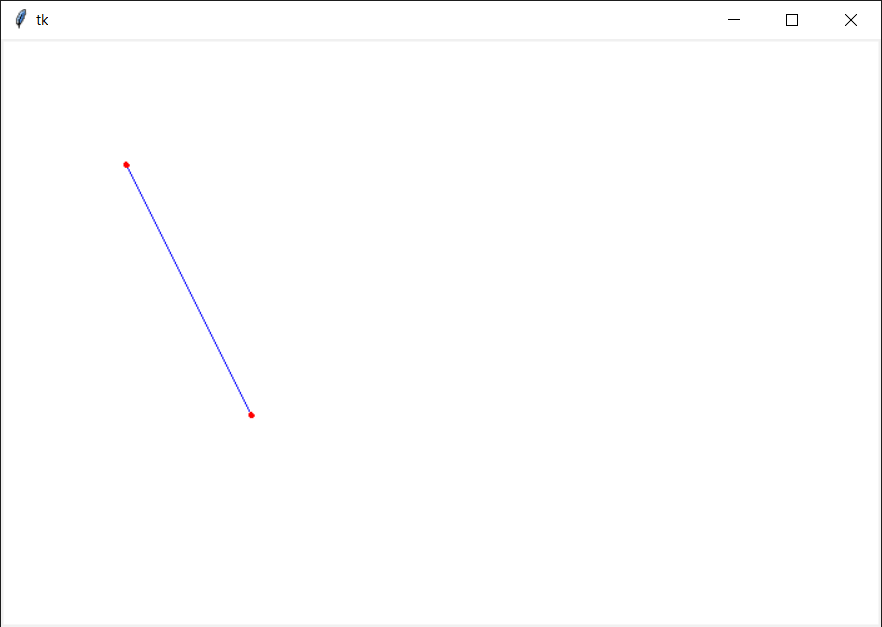
C.create\_line(x1, y1, x2, y2, fill = "blue")

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

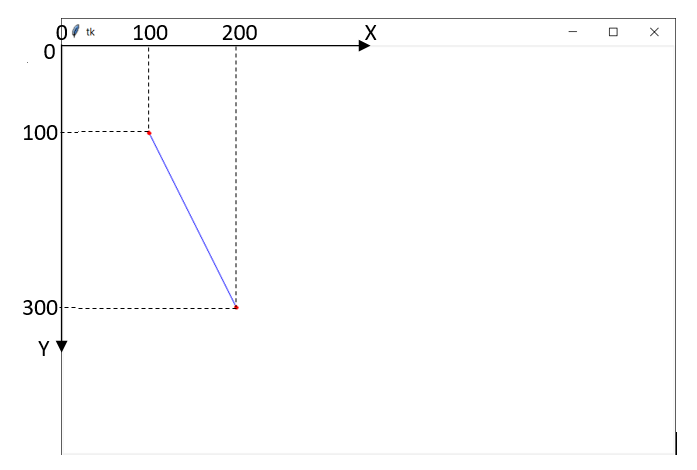
C.create\_oval(x1-2, y1-2, x1+2, y1+2, outline = "red", fill="red")

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 before you submit this project on Canvas.

**Total mark for assessment: 3%. Complete all examples and questions: 2%, and lab attendance: 1%. Submit after the due date: -0.5% and -0.5% for each day after. The due date is end of this tutorial session.**