IT1303 - Programming

PRACTICAL

Funcation and Modules

Objectives

- Identify user-defined functions and modules
- Create user-defined functions and modules to solve programming problem

Funcations

- Create simple program “Practical12A”
- Run it.
- What is the output you expect? Why?

```
1   def iprint():
2       print("A")
3
4   def main():
5       print("B")
6       iprint()
7       print("C")
8
9   print("D")
10  main()
11  print("E")
12
```

```
D
B
A
C
E
```

These are functions.
Notice how do we create functions

Notice that the line print(“A”) belongs to the function def iprint():
The print(“B”), calling iprint() function and print(“C”) belongs to the function main.
Therefore when this program runs, it will skip over these statement and the next valid statement it reaches is print(“D”)
After that it will encounter main() what will lead it to the function def main(): where it will execute the codes under it….

Local / Global variable

```
1   def checkvalue():
2
3       print("In checkvalue, value=", value)
4
5   value = 1
6   checkvalue()
7   print("value=", value)
8
```

```
1   def checkvalue():
2       value = 20
3       print("In checkvalue, value=", value)
4
5   value = 1
6   checkvalue()
7   print("value=", value)
```

```
1   def checkvalue():
2       global value
3       value= 20
4       print("In checkvalue, value=",value)
5
6   value = 1
7   checkvalue()
8   print("value=",value)
```

- Create simple program “Practical12A”
- What is the output you expect? Why?

```
In checkvalue, value= 1
value= 1
```

- Now lets add this line into line 2 “value=20”
- Now what happens?

```
In checkvalue, value= 20
value= 1
```

- What must we do if we want to update the value to 20?

```
In checkvalue, value= 20
value= 20

Process finished with exit code 0
```

Local / Global variable

```
1   def checkvalue():
2       value= 20
3       print("In checkvalue, value=",value)
4       doublecheck()
5
6   def doublecheck():
7       print("In doublecheck, value=",value)
8       checkagain()
9
10  def checkagain():
11      print("In checkagain, value=",value)
12
13  value = 1
14  checkvalue()
15  print("value=",value)
```

- What is the output now?

```
In checkvalue, value= 20
In doublecheck, value= 1
In checkagain, value= 1
value= 1
```

Parameters and Arguments

```
def checkvalue(myvalue):
    value= 20
    print("In checkvalue, value=",value)
    print("In checkvalue, myvalue=", myvalue)

value = 1
checkvalue(value)
print("value=",value)
```

```
In checkvalue, value= 20
In checkvalue, myvalue= 1
value= 1
```

- Create simple program “Practical12B”
- What is the output you expect? Why?

```
1   def checkvalue(myvalue):
2       value= 20
3       print("In checkvalue, value=",value)
4       print("In checkvalue, myvalue=", myvalue)
5       return value
6
7   value = 1
8   value = checkvalue(value)
9   print("value=",value)
```

```
In checkvalue, value= 20
In checkvalue, myvalue= 1
value= 20
```

- How can we change the global variable named value without using the global label?

Exercise A

- Create simple program “myAddQty.py”

Write a function named “addQty” that will total up the numbers in a list provided as an argument.

It will take in a list as a parameter and return back the total of the numbers

The function will be called like this

```
testlist = [30,55,76]
sum = addQty(testlist)
print("sum="+str(sum))
```

- Outcome:

```
sum= 161
```

Exercise B

- Now create another file “getNumbers” to ask the user to enter integer until he enter ‘quit’
- When the user enter quit, the program will display the list of numbers entered
- Outcome:

```
Please enter quantity (Enter 'quit' to exit):32
Please enter quantity (Enter 'quit' to exit):35
Please enter quantity (Enter 'quit' to exit):55
Please enter quantity (Enter 'quit' to exit):41
Please enter quantity (Enter 'quit' to exit):quit
The quantities are: [32, 35, 55, 41]
```

Exercise C

- Now use the addQty function in myAddQty.py to sum up all the number in the list provided in getNumbers.py

- But first we got to remove these lines from myAddQty.py

```
testlist = [30,55,76]
sum = addQty(testlist)
print("sum="+str(sum))
```

- Next import the file

```
import addQty
```

- Finally call the function to sort the list and display the final sorted list

- Outcome:

```
Please enter quantity (Enter 'quit' to exit):50
Please enter quantity (Enter 'quit' to exit):32
Please enter quantity (Enter 'quit' to exit):16
Please enter quantity (Enter 'quit' to exit):40
Please enter quantity (Enter 'quit' to exit):quit
The quantities are: [50, 32, 16, 40]
The total quantity is 138
```

Exercise D

- Now create another module named myAvgqty.py.
- Create the function avgQty to find the average of the quantity provided in the list and return that average.
- Add this function to be called by getNumbers,py so that after it total the quantity, it will also calculate the average.
- Outcome:

```
Please enter quantity (Enter 'quit' to exit):52
Please enter quantity (Enter 'quit' to exit):12
Please enter quantity (Enter 'quit' to exit):30
Please enter quantity (Enter 'quit' to exit):40
Please enter quantity (Enter 'quit' to exit):quit
The quantities are: [52, 12, 30, 40]
The total quantity is 134
The average quantity is 33.5
```

Exercise E

- Now modify the code and add a new list named “itemlist” and it will store a list of items entered by the user.
- When the user quit, the program will display the item list corresponding to the respective quantity list.
- Outcome:

```
Please enter item (Enter 'quit' to end):apples
Please enter quantity:20
Please enter item (Enter 'quit' to end):oranges
Please enter quantity:30
Please enter item (Enter 'quit' to end):pineapples
Please enter quantity:10
Please enter item (Enter 'quit' to end):quit
Item: apples
Quantity: 20
Item: oranges
Quantity: 30
Item: pineapples
Quantity: 10
The total quantity is 60
The average quantity is 20.0
```