Programming and Scripting
Lab Topic 05-Functions

Introduction.

The first activity on this sheet is a quick quiz, answers are at the end of the sheet.
I would suggest that you create another folder in labs called week05-functions, to put the programs you write for the other activities.
You may push this to your GitHub account if you wish.

quiz:

```
# function yo takes one parameter
def yo(a):
    return a * 2
# here we are calling the function yo
# and are passing in an argument, 3
yo(3)
```

1. Look at the program above:
    a. What will be printed out when this program is run?
    b. What would you have to add to the program to get it to print out what the calling the function yo(3) returns?

Student management program

Last week I got you to make a complicated data structure called student which has two attributes, name and modules which has an array of modules and grades.
This week as an exercise I would like to create a program that allows a user to create new students and to view students. This will be a big program for you.
So, we should break it up into smaller bits (this is where functions come in handy)

2. Write a function that prints out a menu of commands we can perform, ie add, view and quit. The function should return what the user chose.
   Test the function. We don't need to worry about error handling yet

```
What would you like to do?
    (a) Add new student
    (v) View students
    (q) Quit
Type one letter (a/v/q):d
you chose d
```

Answer

```
def displayMenu():
    print("What would you like to do?")
    print("\t(a) Add new student")
    print("\t(v) View students")
    print("\t(q) Quit")
    choice = input("Type one letter (a/v/q):").strip()

    return choice
#test the function
choice = displayMenu()
print("you chose {}".format(choice))
```

3. We will now use that function. Create a program that keeps displaying the menu until the user picks q. if the user chooses a then call a function called doAdd() if the user chooses v then call a function called doView().
(I copied the function from the last function into this program)

```
What would you like to do?
    (a) Add new student
    (v) View students
    (q) Quit
Type one letter (a/v/q):a
in adding
What would you like to do?
    (a) Add new student
    (v) View students
    (q) Quit
Type one letter (a/v/q):q
(base) andrews-MacBook-Pro:topic06-functions andrewbeatty$
```

```
def displayMenu():
    print("What would you like to do?")
    print("\t(a) Add new student")
    print("\t(v) View students")
    print("\t(q) Quit")
    choice = input("Type one letter (a/v/q):").strip()
    return choice
def doAdd():
    # we fill this in later
    print("in adding")
def doView():
    # we fill this in later
    print("in viewing")

#main program
choice = displayMenu()
while(choice != 'q'):
    # we could do this with lamda functions
    # I am keeping this basic for the moment
    if choice == 'a':
        doAdd()
    elif choice == 'v':
        doView()
    elif choice !='q':
        print("\n\nplease select either a,v or q")
    choice=displayMenu()
```

Answer

4. We can now write the function doAdd(students). So we need to think what we want this to do.
    a. Read in the student's name (that is straightforward)
    b. Read in the module names and grades (this is a bit more complicated so let's put this in separate function and think about it by itself, see 5. below)
    c. Test this function, it creates a student dict, we can print that out.
    d. We should add the student dict to an array (pass this array in)
    e. Test this.

```
enter name :Andrew
enter name :Mary
[{'name': 'Andrew', 'modules': []}, {'name': 'Mary',
'modules': []}]
```

Note: I have put this in a separate file, I am not thinking about the menu at the moment, I just what to see if I can read in a student
Answer:

```
students= []
def readModules():
    return []

def doAdd(students):
    currentStudent = {}
    currentStudent["name"]=input("enter name :")
    currentStudent["modules"]= readModules()

    students.append(currentStudent)

#test

doAdd(students)

doAdd(students)
print (students)
```

5. We can now think about how to read in the modules. We want to keep reading modules until the user enters a module name of blank.

```
enter name :Andrew
    enter the first Module name (blank to quit) :Programming
        enter grade:45
    enter next module name (blank to quit) :History
        enter grade:90
    enter next module name (blank to quit) :
enter name :Mary
    enter the first Module name (blank to quit) :
[{'name': 'Andrew', 'modules': [{'name': 'Programming', 'grade': 45}, {'name': 'History',
'grade': 90}]}, {'name': 'Mary', 'modules': []}]
```

Doesn't the data structure look ugly, and hard to read. Maybe I should have done the function to view it before the function to add new ones!!

Answer

```
def readModules():
    modules = []
    moduleName = input("\tEnter the first Module name (blank to quit)
:").strip()

    while moduleName != "":
        module = {}
        module["name"]= moduleName
        # I am not doing error handling
        module["grade"]=int(input("\t\tEnter grade:"))
        modules.append(module)
        # now read the next module name
        moduleName = input("\tEnter next module name (blank to quit)
:").strip()

    return modules
```

The rest of the program is the same as in 4 above

6. We can now put these functions into the program we wrote in 3 and see if all works together. Put something into do view for the moment.

```
# the array we store everything in

def displayMenu():
    print("What would you like to do?")
    print("\t(a) Add new student")
    print("\t(v) View students")
    print("\t(q) Quit")
    choice = input("Type one letter (a/v/q):").strip()
    return choice

def doAdd(students):
    currentStudent = {}
    currentStudent["name"]=input("Enter name :")
    currentStudent["modules"]= readModules()
    students.append(currentStudent)

def readModules():
    modules = []
    moduleName = input("\tEnter the first Module name (blank to quit) :").strip()

    while moduleName != "":
        module = {}
        module["name"]= moduleName
        # I am not doing error handling
        module["grade"]=int(input("\t\tEnter grade:"))
        modules.append(module)
        # now read the next module name
        moduleName = input("\tEnter next module name (blank to quit) :").strip()

    return modules

def doView(students):
    # we fill this in later
    print(students)

#main program
students = []
choice = displayMenu()
while(choice != 'q'):
    # we could do this with lamda functions
    # I am keeping this basic for the moment
    if choice == 'a':
        doAdd(students)
    elif choice == 'v':
        doView(students)
    elif choice !='q':
        print("\n\nplease select either a,v or q")
    choice=displayMenu()
```

7. Ok I am going to be mean and say how would you do the doView() function?

   You can approach it the same way we did do add, just print out all the student names first and then make a function called displayModules() and call that after to print each name.

```
def displayModules(modules):
    print ("\tName    \tGrade")
    for module in modules:
        print("\t{}  \t{}".format(module["name"], module["grade"]))

def doView(students):
    for currentStudent in students:
        print(currentStudent["name"])
        displayModules(currentStudent["modules"]);
```

Conclusion

Ok we don't write menus like this anymore.
The purpose of this exercise is for you to see how to break down a program into little bits that you can handle.

Answers to quiz

a. Nothing! Nothing will appear when we run this, there is no print function
b. print(yo(3)
   or
   var = yo(3)
   print(var)

Something extra on the next page

EXTRA

Ok, this is beyond what you need to know yet, but I think it might be interesting. Variables can be of type function, and we can call those variables.

```
def fun1():
    print("this is fun1")
def fun2():
    print("this is fun2")

whichFun = fun1
whichFun()
whichFun= fun2
whichFun()
```

This will print

```
this is fun1
this is fun2
```

This means that lists, tuples and dicts can have variables of type function in them, so we could have a dict that stores the letter of the menu and the function associated with that letter. We could access that function by that letter. (NOTE: that this is not more efficient to run, just more efficient to write)

```
# a different way or dealing with the users choice

def displayMenu():
    print("What would you like to do?")
    print("\t(a) Add new student")
    print("\t(v) View students")
    print("\t(q) Quit")
    choice = input("Type one letter (a/v/q):").strip()
    return choice

def doAdd(students):
    print("do add")
def doView(students):
    print("do View")
def doNothing(dumby):
    pass

#the dict that maps a letter to function
choiceMap = {
    'a': doAdd,
    'v': doView,
    'q': doNothing # q is a valid choice

}
#main program
Students = []
choice = displayMenu()
while(choice != 'q'):
    if choice in choiceMap:
        # run the function
        choiceMap[choice]( students)
    else: # use did not enter something valid
        print("\n\nplease select either a,v or q")

    choice=displayMenu()
```