Python Menu Function and Module

Introduction to Programming
Comp07027

Lecture 9: *Python Menu Function and Module*

Dr Muhammad Aslam
Lecturer UWS Wuxi
Muhammad.Aslam@uws.ac.uk

Menu functions

In many programs users will be offered a choice of options.

A good way to do this is via a menu:

- list the choices available
- provide clear instructions
- provide an exit route

Example: Menu function

```python
def mainMenu():
    print("1. Do something good")
    print("2. Do something bad")
    print("3. Quit")
    selection=int(input("Enter the choice: "))
    if selection==1:
        good()
    elif selection==2:
        bad()
    elif selection==3:
        exit
    else:
        print("Invalid Choice. Enter 1-3")
        mainMenu()


mainMenu()
```

Output

```
1. Do something good
2. Do something bad
3. Quit
Enter the choice:
```

Example: Menu function

```
def mainMenu():
    print("1. Do something good")
    print("2. Do something bad")
    print("3. Quit")
    selection=int(input("Enter the choice: "))
    if selection==1:
        good()
    elif selection==2:
        bad()
    elif selection==3:
        exit
    else:
        print("Invalid Choice. Enter 1-3")
        mainMenu()


mainMenu()
```

Output

```
1. Do something good
2. Do something bad
3. Quit
Enter the choice:1
```

If we enter the choice 1 then we will get an error of
NameError: name 'good' is not defined

Example: Menu function with all functions defined

- We must define the all functions

```python
def mainMenu():
   print("1. Do something good")
   print("2. Do something bad")
   print("3. Quit")
   selection=int(input("Enter the choice: "))
   if selection==1:
      good()
   elif selection==2:
      bad()
   elif selection==3:
      exit
   else:
      print("Invalid Choice. Enter 1-3")
      mainMenu()


def good():
   print("Good")
def bad():
   print("Bad")

mainMenu()
```

Output

```
1. Do something good
2. Do something bad
3. Quit
Enter the choice: 1
Good

Process finished with exit code 0
```

- But this program will not let us to go back to the main menu

Menu function with all functions defined and return back to main function

- We must define the all functions

```
def mainMenu():
   print("1. Do something good")
   print("2. Do something bad")
   print("3. Quit")
   selection=int(input("Enter the choice: "))
   if selection==1:
      good()
   elif selection==2:
      bad()
   elif selection==3:
      exit
   else:
      print("Invalid Choice. Enter 1-3")
      mainMenu()

def good():
   print("Good")
   anykey=input("Enter any key to return to the main menu function ")
   mainMenu()

def bad():
   print("Bad")
   anykey = input("Enter any key to return to the main menu function ")
   mainMenu()

mainMenu()
```

Output

```
1. Do something good
2. Do something bad
3. Quit
Enter the choice: 1
Good
Enter any key to return to the main
menu function1
1. Do something good
2. Do something bad
3. Quit
Enter the choice: 2
Bad
Enter any key to return to the main
menu functiontrrtr
1. Do something good
2. Do something bad
3. Quit
Enter the choice:
```

Menu function with different choice

- If we put different input than integer value, then we will get an error in existing setting

```
def mainMenu():
   print("1. Do something good")
   print("2. Do something bad")
   print("3. Quit")
   selection=int(input("Enter the choice: "))
   if selection==1:
      good()
   elif selection==2:
      bad()
   elif selection==3:
      exit
   else:
      print("Invalid Choice. Enter 1-3")
      mainMenu()

def good():
   print("Good")
   anykey=input("Enter any key to return to the main menu function ")
   mainMenu()

def bad():
   print("Bad")
   anykey = input("Enter any key to return to the main menu function ")
   mainMenu()

mainMenu()
```

Output

```
1. Do something good
2. Do something bad
3. Quit
Enter the choice: dfd

ValueError: invalid literal for int() with
base 10: 'dfd'

Process finished with exit code 1
```

Menu function with different choice and error control

- By adding try and except error we can eliminate the error but that will also end our python program.

```
def mainMenu():
   print("1. Do something good")
   print("2. Do something bad")
   print("3. Quit")
   try:
      selection=int(input("Enter the choice: "))
      if selection==1:
         good()
      elif selection==2:
         bad()
      elif selection==3:
         exit
      else:
         print("Invalid Choice. Enter 1-3")
         mainMenu()
   except ValueError:
      print("Invalid Choice, enter 1-3")


def good():
   print("Good")
   anykey=input("Enter any key to return to the main menu function")
   mainMenu()

def bad():
   print("Bad")
   anykey = input("Enter any key to return to the main menu function")
   mainMenu()

mainMenu()
```

Output

```
1. Do something good
2. Do something bad
3. Quit
Enter the choice: reff
Invalid Choice, enter 1-3

Process finished with exit code 0
```

Menu function with different any type of input

- By introducing the while loop, we can Control the error and bring back the program back to main menu we must start infinite while loop.

```python
def mainMenu():
    print("1. Do something good")
    print("2. Do something bad")
    print("3. Quit")
    while True:
        try:
            selection=int(input("Enter the choice: "))
            if selection==1:
                good()
                break
            elif selection==2:
                bad()
                break
            elif selection==3:
                break

            else:
                print("Invalid Choice. Enter 1-3")
                mainMenu()
        except ValueError:
            print("Invalid Choice, enter 1-3")
    exit


def good():
    print("Good")
    anykey=input("Enter any key to return to the main menu function")
    mainMenu()

def bad():
    print("Bad")
    anykey = input("Enter any key to return to the main menu function")
    mainMenu()

mainMenu()
```

Output

```
1. Do something good
2. Do something bad
3. Quit
Enter the choice: RR
Invalid Choice, enter 1-3
Enter the choice:
```

Menu function another way of implementation

```
print("Welcome to NotFlix")
print("------------------")

#************ FUNCTIONS ************

def login():                    #STUB for Login - full code will be added later
    print("Stub for LOGIN")
    print()

def register():                 #STUB for Register - full code will be added later
    print("Stub for REGISTER")
    print()

def help():                     #STUB for Help - full code will be added later
    print("Stub for HELP")
    print()

def quit():                     #STUB for Quit - full code will be added later
    print("GOODBYE")

def invalid_entry():            #STUB for Invalid Entry - full code will be added later
    print("Invalid entry, please try again")
    print()

def menu():                     #Display menu, prompt for and accept keyboard choice
    print("Please select one of the following:")
    print()
    print("Type 1 if you want to Login")
    print("Type 2 if you want to Register")
    print("Type 3 if you want Help")
    print("Type 4 if you want to Quit")
    choice = input(">>")
    return choice
```

Menu function another way of implementation

- This is the main part of the program, everything before was definitions of functions.

- Program will repeat until option 4 is selected.
- Each other valid entry will call the appropriate function (at this point just a stub).
- Invalid entries will result in an error message and a chance to choose again.

```
menu_choice = menu()    # Call menu() and set                        # menu_choice to                     # returned value

while True:
    if menu_choice == "1":
        login()
    elif menu_choice == "2":
        register()
    elif menu_choice == "3":
        help()
    elif menu_choice == "4":
        quit()
        break
    else:
        invalid_entry()

    menu_choice = menu()
```

Develop a Menu function for a restaurant

-
  ```
  def mainMenu():
      print("1. Starter")
      print("2. Main Food")
      print("3. Dessert")
      print("4. Soft Drinks")
      print("5. Quit")

      while True:
          try:
              selection=int(input("Enter the choice: "))
              if selection==1:
                  starter()
                  break
              elif selection==2:
                  mainfood()
                  break
              elif selection==3:
                  dessert()
                  break
              elif selection==4:
                  softdrinks()
                  break
              elif selection==5:
                  break

              else:
                  print("Invalid Choice. Enter 1-5")
                  mainMenu()
          except ValueError:
              print("Invalid Choice, enter 1-5")
      exit
  ```

```
def starter():
    print("Chilli Potatoes: 10 RMB")
    print("Chilli Paneer: 20 RMB")
    print("Vegitable Gold Coins: 20 RMB")
    anykey=input("Enter any key to return to the main menu function")
    mainMenu()

def mainfood():
    print("Fried rice: 30 RMB")
    print("grlic fried: 30 RMB")
    print("vegetale fried rice: 30 RMB")
    print("mushroom rice fried rice: 30 RMB")
    anykey = input("Enter any key to return to the main menu function")
    mainMenu()

def dessert():
    print("Fried Banana: 30 RMB")
    print("Toffee apples: 30 RMB")
    print("Date wantons: 30 RMB")
    print("Ice cream: 30 RMB")
    anykey = input("Enter any key to return to the main menu function")
    mainMenu()

def softdrinks():
    print("cola: 30 RMB")
    print("sprite: 30 RMB")
    print("pepsi: 30 RMB")
    anykey = input("Enter any key to return to the main menu function")
    mainMenu()

mainMenu()
```

Python Module

- In Python, modules refer to the Python file, which contains Python code like Python statements, classes, functions, variables, etc.
- A file with Python code is defined with extension.py
- For example: In Test.py, where the test is the module name.
- In Python, large code is divided into small modules. The benefit of modules is, it provides a way to share reusable functions.
- Types of modules
- In Python, there are two types of modules.
  1. Built-in Modules
  2. User-defined Modules

Python modules types

- **Built-in modules**
- Built-in modules come with default Python installation.
- One of Python's most significant advantages is its rich library support that contains lots of built-in modules.
- Hence, it provides a lot of reusable code.
- Some commonly used Python built-in modules are datetime, os, math, sys, random, etc.
- **User-defined modules**
- The modules which the user defines or create are called a user-defined module.
- We can create our own module, which contains classes, functions, variables, etc., as per our requirements.

How to import modules?

- In Python, the import statement is used to import the whole module.
- Also, we can import specific classes and functions from a module.
- For example, import module name.
- When the interpreter finds an import statement, it imports the module presented in a search path.
- The module is loaded only once, even we import multiple times.
- To import modules in Python, we use the Python import keyword.
- With the help of the import keyword, both the built-in and user-defined modules are imported.

How to import modules?

-
  ```
  import math

  # use math module functions
  print(math.sqrt(5))
  # Output 2.23606797749979
  ```

- If we want to use more than one module, then we can import multiple modules. This is the simplest form of import a statement that we already use in the above example.

-
  ```
  import math, random

  print(math.factorial(5))
  print(random.randint(10, 20))
  ```

How to import specific class of modules?

- Import only specific classes or functions from a module
- To import specific classes or functions, we can use the form...import statement.
- It is an alternate way to import. By using this form, we can import individual attributes and methods directly into the program.
- In this form, we are not required to use the module name. See the following example. Syntax of from...import statement:

```
# import only factorial function from math module
from math import factorial

print(factorial(5))
```

- If we want to use the module with a different name, we can use from..import…as statement.

```
import random as rand

print(rand.randrange(10, 20, 2))
```

Import all names of a module

- If we need to import all functions and attributes of a specific module, then instead of writing all function names and attribute names, we can import all using an asterisk *.

-
  ```
  from math import *
  print(pow(4,2))
  print(factorial(5))

  print(pi*3)
  print(sqrt(100))
  ```

Create Module in Python

- In Python, to create a module, write Python code in the file, and save that file with the.py extension. Here our module is created.
- def my_func():

  print("Learn Python with PYnative")

Variables in Module

- In Python, the module contains Python code like classes, functions, methods, but it also has variables. A variable can list, tuple, dict, etc.
- Let's see this with an example: First, create a Python module with the name test_module.py and write the below code in that file.
- cities_list = ['Mumbai', 'Delhi', 'Bangalore', 'Karnataka', 'Hyderabad']

- Now, create a Python file with the name `test_file.py`, write the below code and import the above module `test_module.py` .

Create Module in Python

Now, create a Python file with the name `test_file.py`, write the below code and import the above module `test_module.py` in that file. See the following code.

```
import test_module

# access first city
city = test_module.cities_list[1]
print("Accessing 1st city:", city)

# Get all cities
cities = test_module.cities_list
print("Accessing All cities :", cities)
```

When we execute this `test_file.py`, the variable of `test_module.py` is accessible using the dot`(.)`operator.

Questions??