30 Days Of Python: Day 12 – Modules

Modules

What is a Module

A module is a file containing a set of codes or a set of functions which can be included to an application. A module could be a file containing a single variable, a function or a big code base.

Creating a Module

To create a module we write our codes in a python script and we save it as a .py file. Create a file named mymodule.py inside your project folder. Let us write some code in this file.

```
# mymodule.py file
def generate_full_name(firstname, lastname):
    return firstname + ' ' + lastname
```

Create main.py file in your project directory and import the mymodule.py file.

Importing a Module

To import the file we use the *import* keyword and the name of the file only.

```
# main.py file
import mymodule
print(mymodule.generate_full_name('Suniksha', 'Patel'))
```

Import Functions from a Module

We can have many functions in a file and we can import all the functions differently.

```
# main.py file
from mymodule import generate_full_name, sum_two_nums, person, gravity
print(generate_full_name('Suniksha','Patel'))
print(sum_two_nums(1,9))
mass = 100;
weight = mass * gravity
print(weight)
print(person['firstname'])
```

Import Functions from a Module and Renaming

During importing we can rename the name of the module.

```
# main.py file
from mymodule import generate_full_name as fullname, sum_two_nums as total, person
as p, gravity as g
print(fullname('Suniksha','Patel'))
print(total(1, 9))
mass = 100;
weight = mass * g
print(weight)
print(p)
print(p['firstname'])
```

Import Built-in Modules

Like other programming languages we can also import modules by importing the file/function using the key word *import*. Let's import the common module we will use most of the time. Some of the common built-in modules: *math*, *datetime*, *os*,*sys*, *random*, *statistics*, *collections*, *json*,*re*

OS Module

Using python *os* module it is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating, changing current working directory, and removing a directory (folder), fetching its contents, changing and identifying the current directory.

```
# import the module
import os
```

```
# Creating a directory
os.mkdir('directory_name')
# Changing the current directory
os.chdir('path')
# Getting current working directory
os.getcwd()
# Removing directory
os.rmdir()
```

Sys Module

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment. Function sys.argv returns a list of command line arguments passed to a Python script. The item at index 0 in this list is always the name of the script, at index 1 is the argument passed from the command line.

Example of a script.py file:

```
import sys
#print(sys.argv[0], argv[1],sys.argv[2])  # this line would print out: filename
argument1 argument2
print('Welcome {}. Enjoy  {} challenge!'.format(sys.argv[1], sys.argv[2]))
```

Now to check how this script works I wrote in command line:

```
python script.py Suniksha 30DaysOfPython
```

The result:

```
Welcome Suniksha. Enjoy  30DayOfPython challenge!
```

Some useful sys commands:

```
# to exit sys
sys.exit()
# To know the largest integer variable it takes
sys.maxsize
# To know environment path
sys.path
# To know the version of python you are using
sys.version
```

Statistics Module

The statistics module provides functions for mathematical statistics of numeric data. The popular statistical functions which are defined in this module: *mean*, *median*, *mode*, *stdev* etc.

```
from statistics import * # importing all the statistics modules
ages = [20, 20, 4, 24, 25, 22, 26, 20, 23, 22, 26]
print(mean(ages))       # ~22.9
print(median(ages))     # 23
print(mode(ages))       # 20
print(stdev(ages))      # ~2.3
```

Math Module

Module containing many mathematical operations and constants.

```
import math
print(math.pi)           # 3.141592653589793, pi constant
print(math.sqrt(2))      # 1.4142135623730951, square root
print(math.pow(2, 3))    # 8.0, exponential function
print(math.floor(9.81))  # 9, rounding to the lowest
print(math.ceil(9.81))   # 10, rounding to the highest
print(math.log10(100))   # 2, logarithm with 10 as base
```

Now, we have imported the *math* module which contains lots of function which can help us to perform mathematical calculations. To check what functions the module has got, we can use *help(math)*, or *dir(math)*. This will display the available functions in the module. If we want to import only a specific function from the module we import it as follows:

```
from math import pi
print(pi)
```

It is also possible to import multiple functions at once

```
from math import pi, sqrt, pow, floor, ceil, log10
print(pi)                 # 3.141592653589793
print(sqrt(2))            # 1.4142135623730951
print(pow(2, 3))          # 8.0
print(floor(9.81))        # 9
print(ceil(9.81))         # 10
print(math.log10(100))    # 2
```

But if we want to import all the function in math module we can use * .

```
from math import *
print(pi)                  # 3.141592653589793, pi constant
print(sqrt(2))             # 1.4142135623730951, square root
print(pow(2, 3))           # 8.0, exponential
print(floor(9.81))         # 9, rounding to the lowest
print(ceil(9.81))          # 10, rounding to the highest
print(math.log10(100))     # 2
```

When we import we can also rename the name of the function.

```
from math import pi as  PI
print(PI) # 3.141592653589793
```

String Module

A string module is a useful module for many purposes. The example below shows some use of the string module.

```
import string
```

```
print(string.ascii_letters) # abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
print(string.digits)        # 0123456789
print(string.punctuation)   # !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Random Module

By now you are familiar with importing modules. Let us do one more import to get very familiar with it. Let us import *random* module which gives us a random number between 0 and 0.9999.... The *random* module has lots of functions but in this section we will only use *random* and *randint*.

```
from random import random, randint
print(random())   # it doesn't take any arguments; it returns a value between 0
and 0.9999
print(randint(5, 20)) # it returns a random integer number between [5, 20]
```

```
inclusive
```

🍪 You are going far. Keep going! You have just completed day 12 challenges and you are 12 steps a head in to your way to greatness. Now do some exercises for your brain and muscles.

💻 Exercises: Day 12

Exercises: Level 1

1. Writ a function which generates a six digit/character random_user_id.

2.
   ```
   print(random_user_id());
   '1ee33d'
   ```
3. Modify the previous task. Declare a function named user_id_gen_by_user. It doesn't take any parameters but it takes two inputs using input(). One of the inputs is the number of characters and the second input is the number of IDs which are supposed to be generated.

```
print(user_id_gen_by_user()) # user input: 5 5
#output:
#kcsy2
#SMFYb
#bWmeq
#ZXOYh
#2Rgxf

print(user_id_gen_by_user()) # 16 5
#1GCSgPLMaBAVQZ26
#YD7eFwNQKNs7qXaT
#ycArC5yrRupyG00S
#UbGxOFI7UXSWAyKN
#dIV0SSUTgAdKwStr
```

3. Write a function named rgb_color_gen. It will generate rgb colors (3 values ranging from 0 to 255 each).

```
print(rgb_color_gen())
# rgb(125,244,255) - the output should be in this form
```

Exercises: Level 2

1. Write a function list_of_hexa_colors which returns any number of hexadecimal colors in an array (six hexadecimal numbers written after #. Hexadecimal numeral system is made out of 16 symbols, 0-9 and first 6 letters of the alphabet, a-f. Check the task 6 for output examples).
2. Write a function list_of_rgb_colors which returns any number of RGB colors in an array.
3. Write a function generate_colors which can generate any number of hexa or rgb colors.

```
   generate_colors('hexa', 3) # ['#a3e12f','#03ed55','#eb3d2b']
   generate_colors('hexa', 1) # ['#b334ef']
   generate_colors('rgb', 3)  # ['rgb(5, 55, 175','rgb(50, 105, 100','rgb(15, 26, 
80']
   generate_colors('rgb', 1)  # ['rgb(33,79, 176)']
```

Exercises: Level 3

1. Call your function shuffle_list, it takes a list as a parameter and it returns a shuffled list
2. Write a function which returns an array of seven random numbers in a range of 0-9. All the numbers must be unique.

🎉 CONGRATULATIONS ! 🎉