Python Modules

- A python module can be defined as a python program file which contains a python code including python functions, class, or variables. In other words, we can say that our python code file saved with the extension (.py) is treated as the module. We may have a runnable code inside the python module.
- Modules in Python provides us the flexibility to organize the code in a logical way.
- To use the functionality of one module into another, we must have to import the specific module.

Example

- we will create a module named as file.py which contains a function func that contains a code to print some message on the console.
- Let's create the module named as **file.py.**
- #displayMsg prints a message to the name being passed.
- **def** displayMsg(name)
- **print**("Hi "+name);
- Here, we need to include this module into our main module to call the method displayMsg() defined in the module named file.

Loading the module in our python code

- The import statement
- **import** module1,module2,..... module n
- Example
- **import** file;
- name = input("Enter the name?")
- file.displayMsg(name)
- **Output:**
- Enter the name?John Hi John

The from-import statement

- **from** < module-name> **import** <name 1>, <name 2>..,<name n>
- **calculation.py:**
- #place the code in the calculation.py
- **def** summation(a,b):
- **return** a+b
- **def** multiplication(a,b):
- **return** a*b;
- **def** divide(a,b):
- **return** a/b;
-

- **Main.py:**
- **from** calculation **import** summation
- #it will import only the summation() from calculat ion.py
- a = int(input("Enter the first number"))
- b = int(input("Enter the second number"))
- **print**("Sum = ",summation(a,b)) #we do not need to specify the module name while accessing sum mation()
- **Output:**
- Enter the first number10 Enter the second number20 Sum = 30

We can also import all the attributes from a module by using *.

- **from** <module> **import** *

Renaming a module

- **import** <module-name> as <specific-name>
- Example
- #the module calculation of previous example is imported in this example as cal.
- **import** calculation as cal;
- a = int(input("Enter a?"));
- b = int(input("Enter b?"));
- **print**("Sum = ",cal.summation(a,b))
- **Output:**
- Enter a?10
- Enter b?20
- Sum = 30
-

Using dir() function

- The dir() function returns a sorted list of names defined in the passed module. This list contains all the sub-modules, variables and functions defined in this module.
- Example
- **import** json
-
- List = dir(json)
-
- **print**(List)
- **Output:**
- ['JSONDecoder', 'JSONEncoder', '__all__', '__author__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__path__', '__spec__', '__version__', '_default_decoder', '_default_encoder', 'decoder', 'dump', 'dumps', 'encoder', 'load', 'loads', 'scanner']

The reload() function

- As we have already stated that, a module is loaded once regardless of the number of times it is imported into the python source file. However, if you want to reload the already imported module to re-execute the top-level code, python provides us the reload() function. The syntax to use the reload() function is given below.
- reload(<module-name>)
- for example, to reload the module calculation defined in the previous example, we must use the following line of code.
- reload(calculation)

Scope of variables

- In Python, variables are associated with two types of scopes. All the variables defined in a module contain the global scope unless or until it is defined within a function.
- All the variables defined inside a function contain a local scope that is limited to this function itself. We can not access a local variable globally.
- If two variables are defined with the same name with the two different scopes, i.e., local and global, then the priority will always be given to the local variable.
- Consider the following example.

- Example
- name = "john"
- **def** print_name(name):
- **print**("Hi",name) #prints the name that is local to t his function only.
- name = input("Enter the name?")
- print_name(name)
- **Output:**
- Hi David

Python packages

- The packages in python facilitate the developer with the application development environment by providing a hierarchical directory structure where a package contains sub-packages, modules, and sub-modules. The packages are used to categorize the application level code efficiently.
- Let's create a package named Employees in your home directory. Consider the following steps.
- 1. Create a directory with name Employees on path /**home**.
- 2. Create a python source file with name ITEmployees.py on the path /**home**/**Employees**.

- **ITEmployees.py**
- **def** getITNames():
- List = ["John", "David", "Nick", "Martin"]
- **return** List;
- 3. Similarly, create one more python file with name BPOEmployees.py and create a function getBPONames().
- 4. Now, the directory Employees which we have created in the first step contains two python modules. To make this directory a package, we need to include one more file here, that is __init__.py which contains the import statements of the modules defined in this directory.

- **__init__.py**
- **from** ITEmployees **import** getITNames
- **from** BPOEmployees **import** getBPONames
- 5. Now, the directory **Employees** has become the package containing two python modules. Here we must notice that we must have to create __init__.py inside a directory to convert this directory to a package.
- 6. To use the modules defined inside the package Employees, we must have to import this in our python source file. Let's create a simple python source file at our home directory (/home) which uses the modules defined in this package.

- **Test.py**
- **import** Employees
- **print**(Employees.getNames())
- **Output:**
- ['John', 'David', 'Nick', 'Martin']

Python Date and time

- In the real world applications, there are the scenarios where we need to work with the date and time. There are the examples in python where we have to schedule the script to run at some particular timings.
- In python, the date is not a data type, but we can work with the date objects by importing the module named with datetime, time, and calendar.
- In this section of the tutorial, we will discuss how to work with the date and time objects in python.

- Tick
- In python, the time instants are counted since 12 AM, 1st January 1970. The function time() of the module time returns the total number of ticks spent since 12 AM, 1st January 1970. A tick can be seen as the smallest unit to measure the time.

- Example
- **import** time;
-
- #prints the number of ticks spent since 12 AM, 1st January 1970
-
- **print**(time.time())
- **Output:**
- 1545124460.9151757

How to get the current time?

- The localtime() functions of the time module are used to get the current time tuple. Consider the following example.
- Example
- **import** time;
-
- #returns a time tuple
-
- **print**(time.localtime(time.time()))
- **Output:**
- time.struct_time(tm_year=2018, tm_mon=12, tm_mday=18, tm_hour=15, tm_min=1, tm_sec=32, tm_wday=1, tm_yday=352, tm_isdst=0)

Time tuple

The time is treated as the tuple of 9 numbers. Let's look at the members of the time tu

Index	Attribute	Values
0	Year	4 digit (for example 2018
1	Month	1 to 12
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 60
6	Day of weak	0 to 6
7	Day of year	1 to 366
8	Daylight savings	-1, 0, 1 , or -1

Getting formatted time

- The time can be formatted by using the asctime() function of time module. It returns the formatted time for the time tuple being passed.
- Example
- **import** time;
-
- #returns the formatted time
-
- **print**(time.asctime(time.localtime(time.time())))
- **Output:**
- Tue Dec 18 15:31:39 2018

Python sleep time

- The sleep() method of time module is used to stop the execution of the script for a given amount of time. The output will be delayed for the number of seconds given as float.

- Example
- **import** time
- **for** i **in** range(0,5):
- **print**(i)
- #Each element will be printed after 1 second time.sleep(1)
- **Output:**
- 0
- 1
- 2
- 3
- 4

The datetime Module

- The datetime module enables us to create the custom date objects, perform various operations on dates like the comparison, etc.
- To work with dates as date objects, we have to import datetime module into the python source code.
- Consider the following example to get the datetime object representation for the current time.

- Example
- **import** datetime;
-
- #returns the current datetime object
-
- **print**(datetime.datetime.now())
- **Output:**
- 2018-12-18 16:16:45.462778

Creating date objects

- We can create the date objects by passing the desired date in the datetime constructor for which the date objects are to be created.
- Consider the following example.

- Example
- **import** datetime;
-
- #returns the datetime object for the specified date
-
- **print**(datetime.datetime(2018,12,10))
- **Output:**
- 2018-12-10 00:00:00

- We can also specify the time along with the date to create the datetime object. Consider the following example. Example
- **import** datetime;
-
- #returns the datetime object for the specified ti me
-
- **print**(datetime.datetime(2018,12,10,14,15,10))

- **Output:**
- 2018-12-10 14:15:10

Comparison of two dates

- We can compare two dates by using the comparison operators like >, >=, <, and <=.
- Consider the following example.

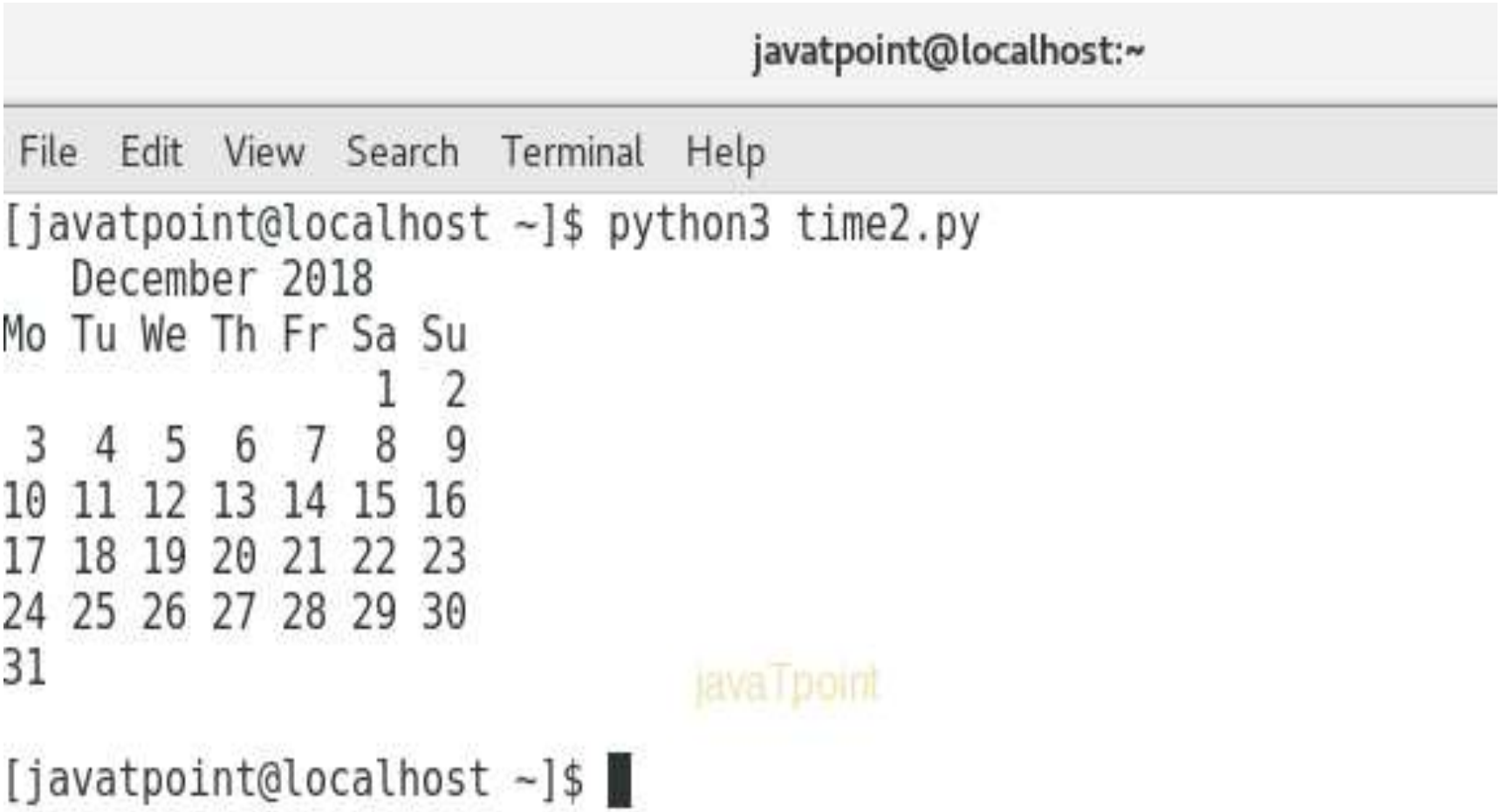
- Example
- **from** datetime **import** datetime as dt
- #Compares the time. If the time is in between 8AM and 4PM, then it prints working hours otherwise it prints fun hours
- **if** dt(dt.now().year,dt.now().month,dt.now().day,8)<dt.now()<dt(dt.now().year,dt.now().month,dt.now().day,16):
- **print**("Working hours....")
- **else**:
- **print**("fun hours")
- **Output:**
- fun hours

The calendar module

- Python provides a calendar object that contains various methods to work with the calendars.
- Consider the following example to print the Calendar of the last month of 2018.

- Example
- **import** calendar;
- cal = calendar.month(2018,12)
- #printing the calendar of December 2018
- **print**(cal)

- **Output:**


javatpoint@localhost:~
File Edit View Search Terminal Help
[javatpoint@localhost ~]$ python3 time2.py
December 2018
Mo Tu We Th Fr Sa Su
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
javaTpoint
[javatpoint@localhost ~]$

Printing the calendar of whole year

- The prcal() method of calendar module is used to print the calendar of the whole year. The year of which the calendar is to be printed must be passed into this method.
- Example
- **import** calendar
-
- #printing the calendar of the year 2019
- calendar.prcal(2019)

```
      January
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

```
     February
Mo Tu We Th Fr Sa Su
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28
```

```
       March
Mo Tu We Th Fr Sa Su
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

```
       April
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
```

```
        May
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

```
        June
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```



```
       July
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

```
      August
Mo Tu We Th Fr Sa Su
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

```
     September
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
```

```
      October
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

```
     November
Mo Tu We Th Fr Sa Su
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

```
     December
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```