VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

USING PYTHON LIBRARIES

COLLECTION OF MODULES

Introduction

- As our program become larger and more complex the need to organize our code becomes greater. We have already learnt in ***Function chapter that large and complex program should be divided into functions*** that perform a specific task. ***As we write more and more functions in a program, we should consider organizing of functions by storing them in modules***
- A module is simply a file that contains Python code. When we break a program into modules, each modules should contain functions that perform related tasks.
- Commonly used modules that contains source code for generic needs are called ***Libraries***.

Introduction

- When we speak about working with libraries in Python, we are, in fact, working with modules that are created inside Library or Packages. Thus a Python program comprises three main components:
    - Library or Package
    - Module
    - Function/Sub-routine

Relationship between Module, Package and Library in Python

- □ A Module is a file containing Python definitions (docstrings) , functions, variables, classes and statements
- □ Python package is simply a directory of Python module(s)
- □ Library is a collection of various packages. Conceptually there is no difference between package and Python library. In Python a library is used to loosely describe a collection of core or main modules

Commonly used Python libraries

STANDARD LIBRARY	
math module	Provides mathematical functions
cmath module	Provides function for complex numbers
random module	For generating random numbers
Statistics module	Functions for statistical operation
Urllib	Provides URL handling functions so that you can access websites from within your program.
NumPy library	This library provides some advance math functionalities along with tools to create and manipulate numeric arrays
SciPy library	Another useful library that offers algorithmic and mathematical tools for scientific calculation
Tkinter library	Provides traditional user interface toolkit and helps you to create user friendly GUI interface for different types of applications.
Matplotlib library	Provides functions and tools to produce quality output in variety of formats such as plot, charts, graph etc,

What is module?

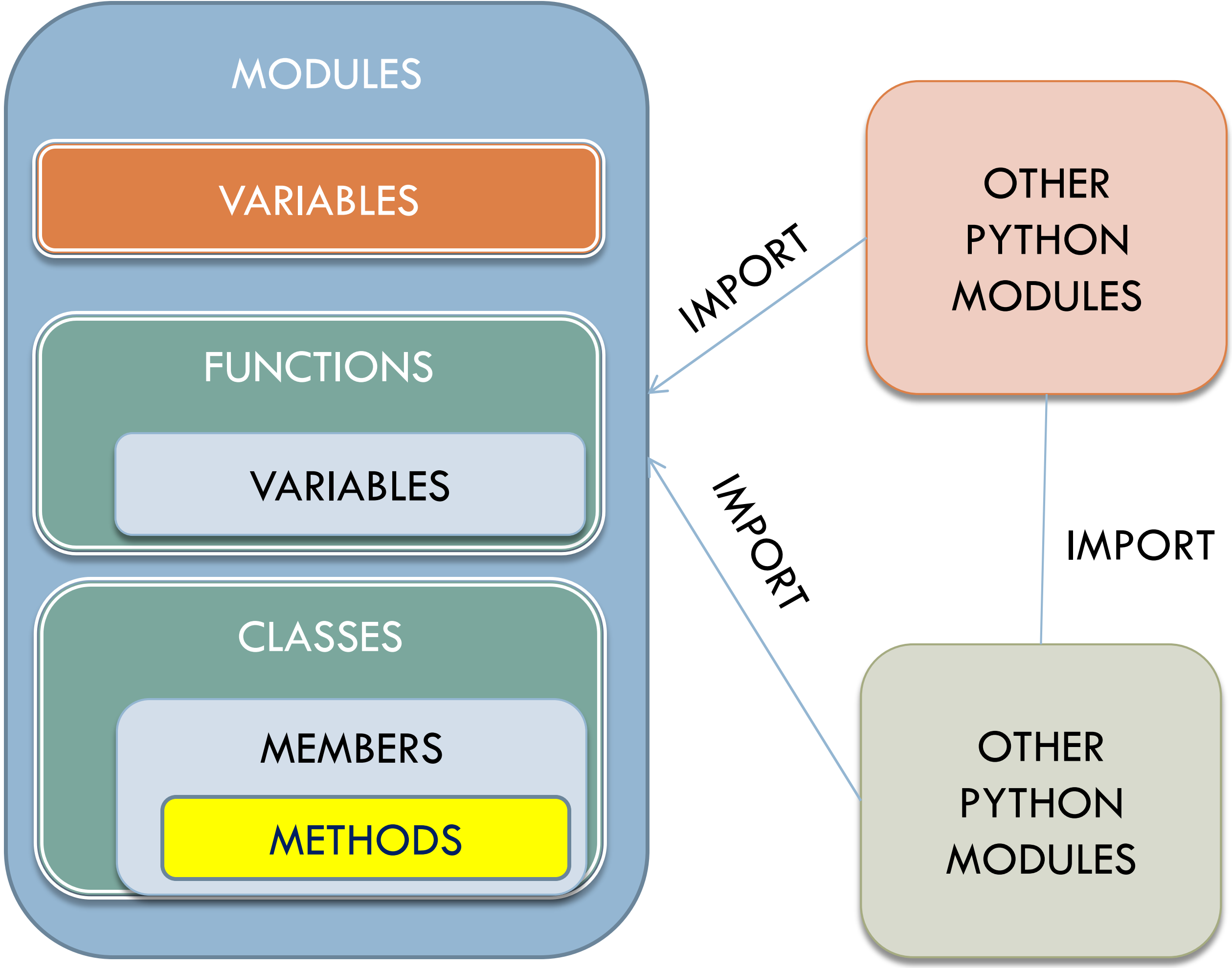
- Act of partitioning a program into individual components(modules) is called modularity. A module is a separate unit in itself.
  - It reduces its complexity to some degree
  - It creates numbers of well-defined, documented boundaries within program.
  - Its contents can be reused in other program, without having to rewrite or recreate them.

Structure of Python module

- A python module is simply a normal python file(.py) and contains functions, constants and other elements.
- Python module may contains following objects:

docstring	Triple quoted comments. Useful for documentation purpose
Variables and constants	For storing values
Classes	To create blueprint of any object
Objects	Object is an instance of class. It represent class in real world
Statements	Instruction
Functions	Group of statements

Composition/Structure of python module



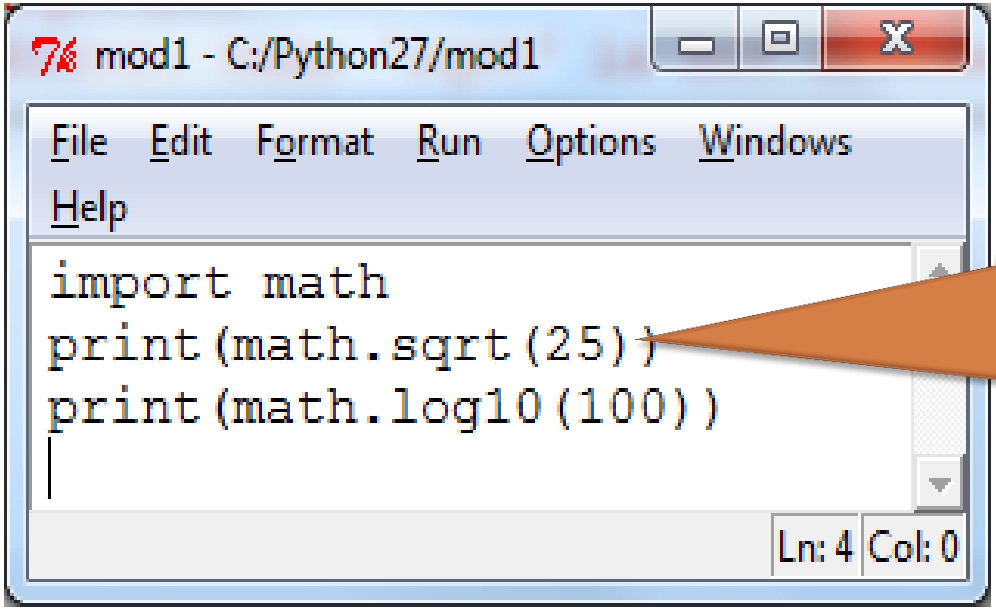
Importing Python modules

- To import entire module
  - **import <module name>**
  - **Example:** **import math**
- To import specific function/object from module:
  - **from <module_name> import <function_name>**
  - **Example:** **from math import sqrt**
- **import *** : can be used to import all names from module into current calling module

Accessing function/constant of imported module

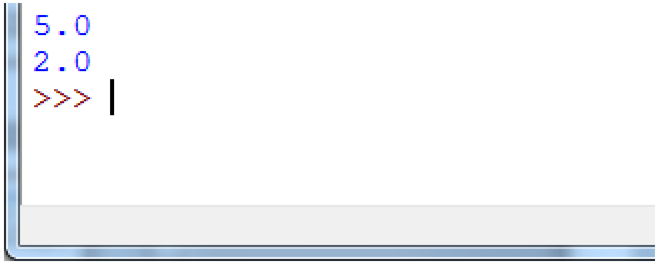
- To use function/constant/variable of imported module we have to specify module name and function name separated by dot(.). This format is known as dot notation.
  - **<module_name>.<function_name>**
  - **Example: print(math.sqrt(25))**

Example : import module_name



```
import math
print(math.sqrt(25))
print(math.log10(100))
```

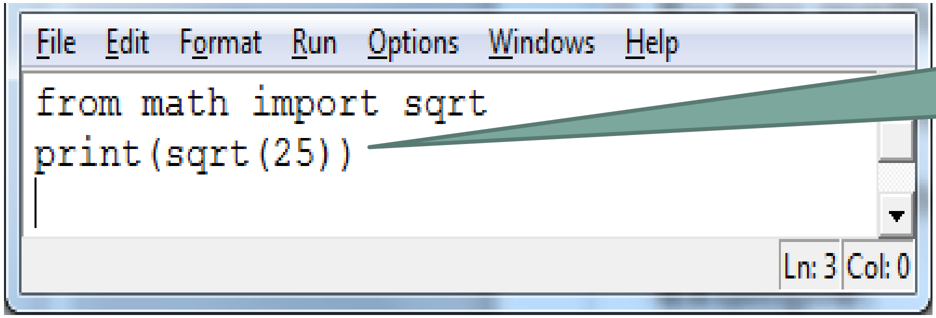
Here we can see that after **import math**, we have to qualify the name of function with name of package using dot(.) notation



```
5.0
2.0
>>> 
```

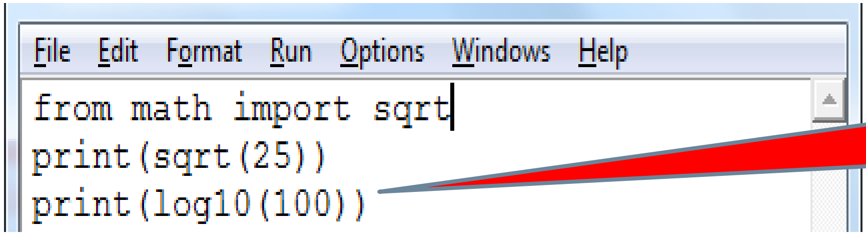
Example: from module import function

- By this method only particular method will be added to our current program. We need not to qualify name of method with name of module. Or example:



```
from math import sqrt
print(sqrt(25))
```

Here function sqrt() is directly written

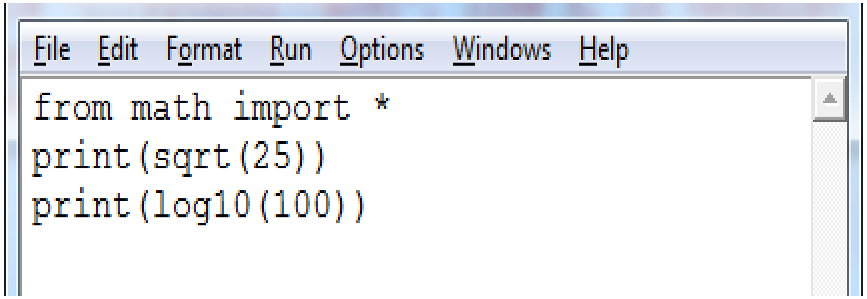


```
from math import sqrt
print(sqrt(25))
print(log10(100))
```

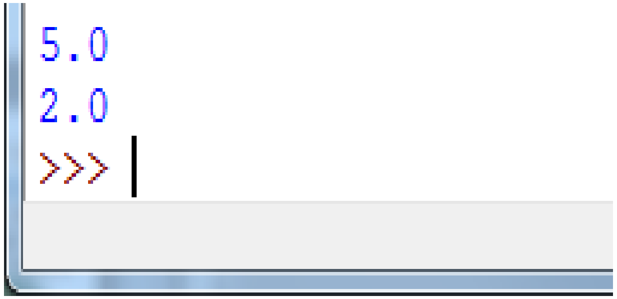
This line will not be executed and gives an error

Example: from module import *

- It is similar to importing the entire package as “import package” but by this method qualifying each function with module name is not required.



```
from math import *
print(sqrt(25))
print(log10(100))
```

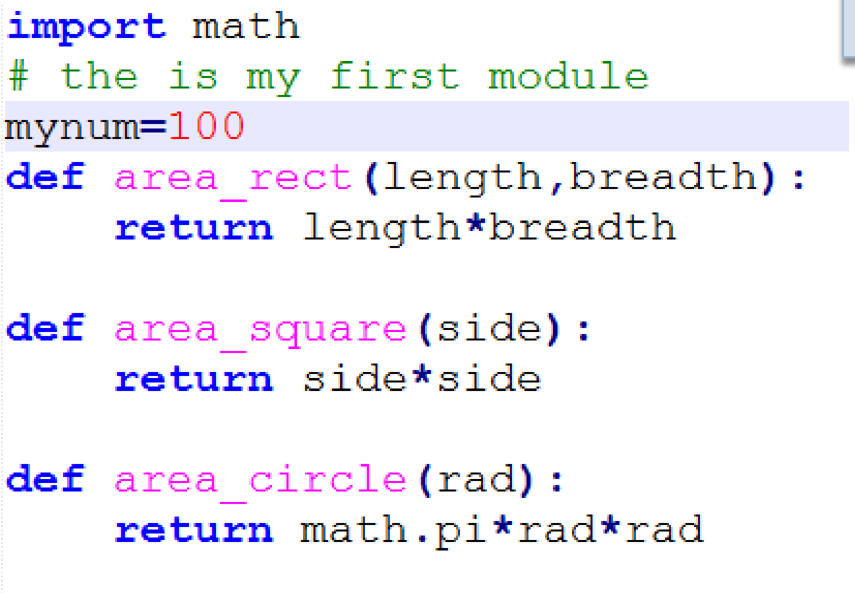


```
5.0
2.0
>>> 
```

We can also import multiple elements of module as :
from math import sqrt, log10

Creating our own Module

- □ Create new python file(.py) and type the following code as:



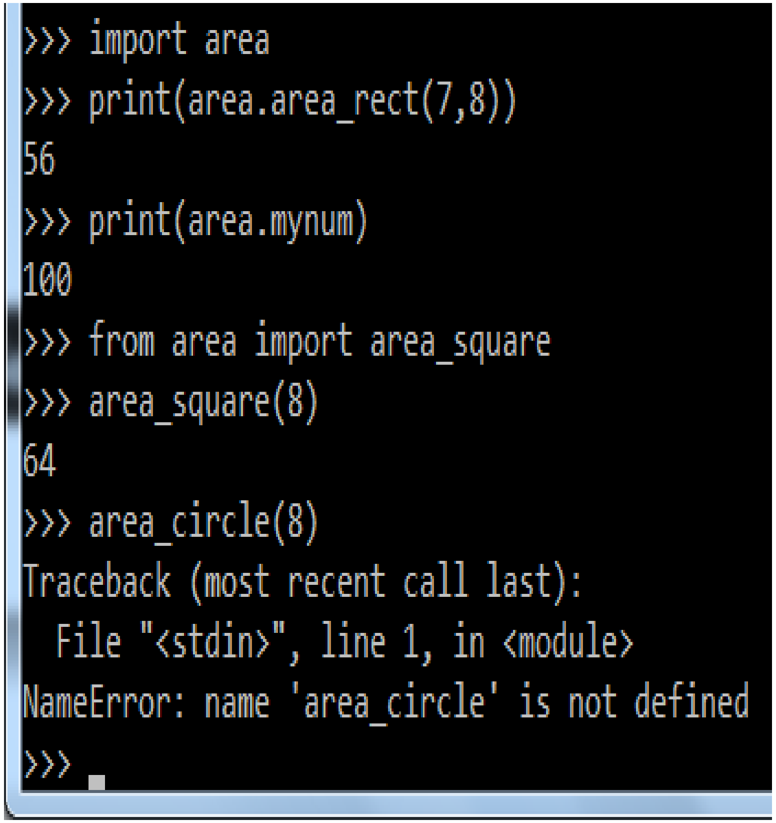
```
import math
# the is my first module
mynum=100
def area_rect(length,breadth):
    return length*breadth

def area_square(side):
    return side*side

def area_circle(rad):
    return math.pi*rad*rad
```

Save this file are "area.py"

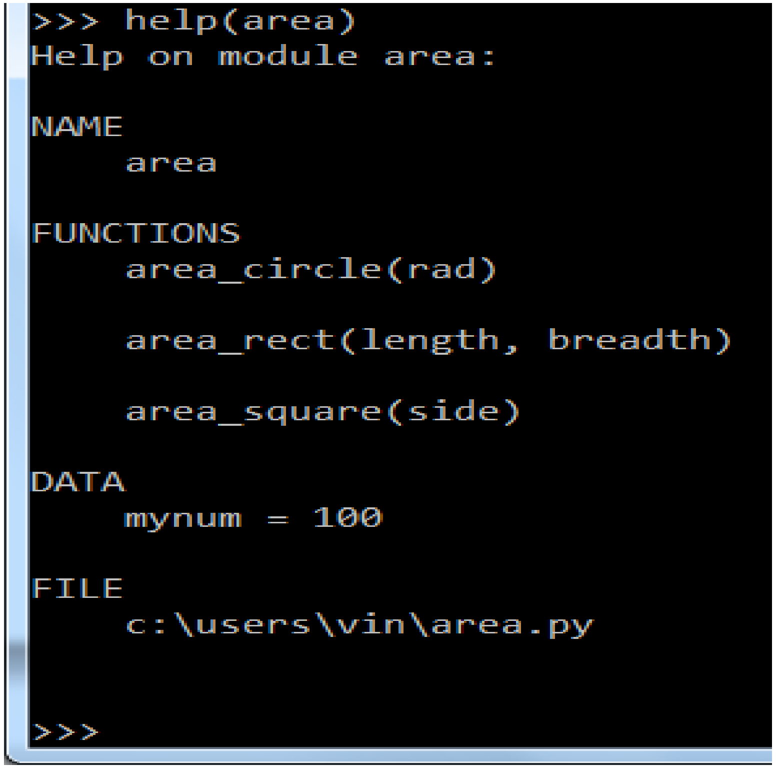
Execute the following code to import and use your own module



```
>>> import area
>>> print(area.area_rect(7,8))
56
>>> print(area.mynum)
100
>>> from area import area_square
>>> area_square(8)
64
>>> area_circle(8)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'area_circle' is not defined
>>>
```

help() function

- Is used to get detailed information about any module like : name of module, functions inside module, variables inside module and name of file etc.



```
>>> help(area)
Help on module area:

NAME
    area

FUNCTIONS
    area_circle(rad)

    area_rect(length, breadth)

    area_square(side)

DATA
    mynum = 100

FILE
    c:\users\vin\area.py


>>>
```

Namespace

- Is a space that holds a bunch of names. Consider an example:
  - In a CCA competition of vidyalaya, there are students from different classes having similar names, say there are three POOJA GUPTA, one from class X, one from XI and one from XII
  - As long as they are in their class there is no confusion, since in X there is only one POOJA GUPTA, and same with XI and XII
  - But problem arises when the students from X, XI, XII are sitting together, now calling just POOJA GUPTA would create confusion-which class's POOJA GUPTA. So one need to qualify the name as class X's POOJA GUPTA, or XI's or XII's and so on.

Namespace

- From the previous example, we can say that class X has its own namespace where there no two names as POOJA GUPTA; same holds for XI and XII
- In Python terms, namespace can be thought of as a named environment holding logical group of related objects.
- For every python module(.py), Python creates a namespace having its name similar to that of module's name. That is, namespace of module AREA is also AREA.
- **When 2 namespace comes together, to resolve any kind of object name dispute, Python asks you to qualify the name of object as <modulename>.<objectname>**

Processing of **import <module>**

- The code of import module is interpreted and executed
- Defined functions and variables in the module are now available to program in **new namespace created by the name of module**
- **For example, if the imported module is area, now you want to call the function area_circle(), it would be called as area.area_circle()**

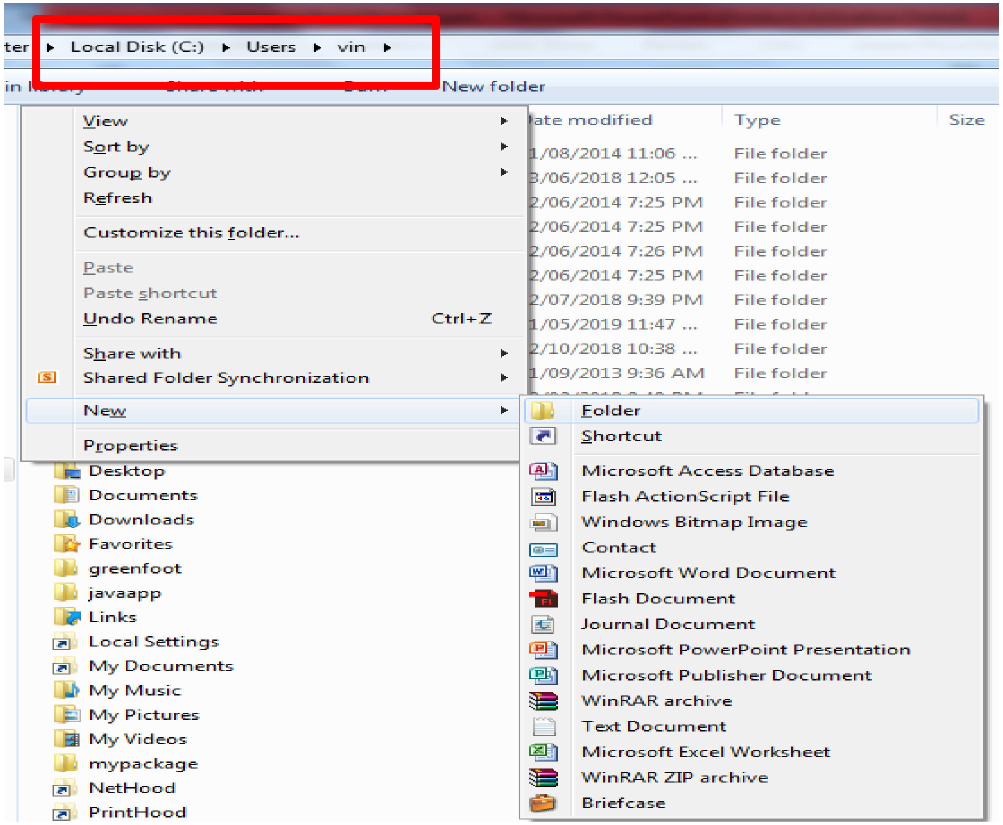
Processing of from module import object

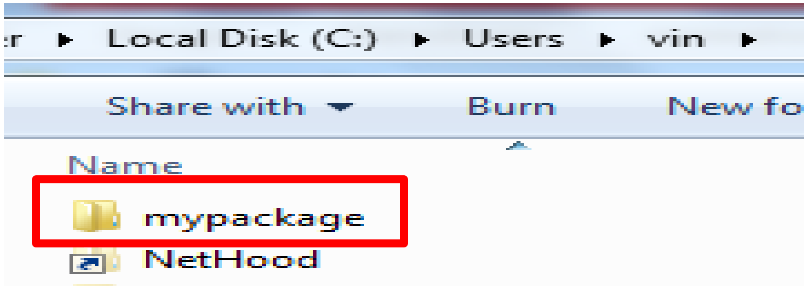
- When we issue from module import object command:
  - The code of imported module is interpreted and executed
  - Only the asked function and variables from module are now available in the current namespace i.e. **no new namespace is created that's why we can call object of imported module without qualifying the module name**
  - **For example:**

    ```
    from math import sqrt
    print(sqrt(25))
    ```
  - *However if the same function name is available in current namespace then local function will hide the imported module's function*
  - *Same will be apply for* **from math import *** *method*

Creating Package

- Step 1
  - Create a new folder which you want to act as package. The name of folder will be the name of your package





IN THE C:\USERS\VIN

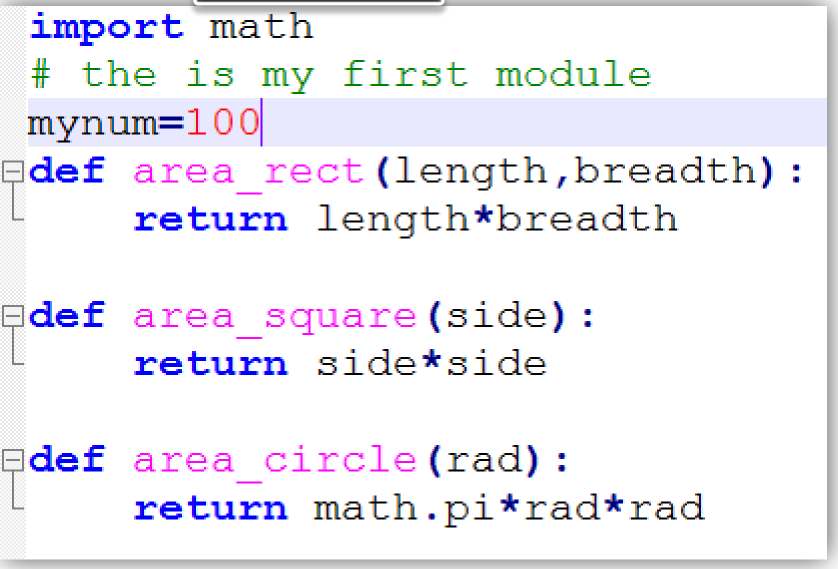
A new Folder “mypackage” is created.

Note: you can create folder in any desired location

Creating Package

- Step 2: Create modules (.py) and save it in "mypackage" folder

area.py

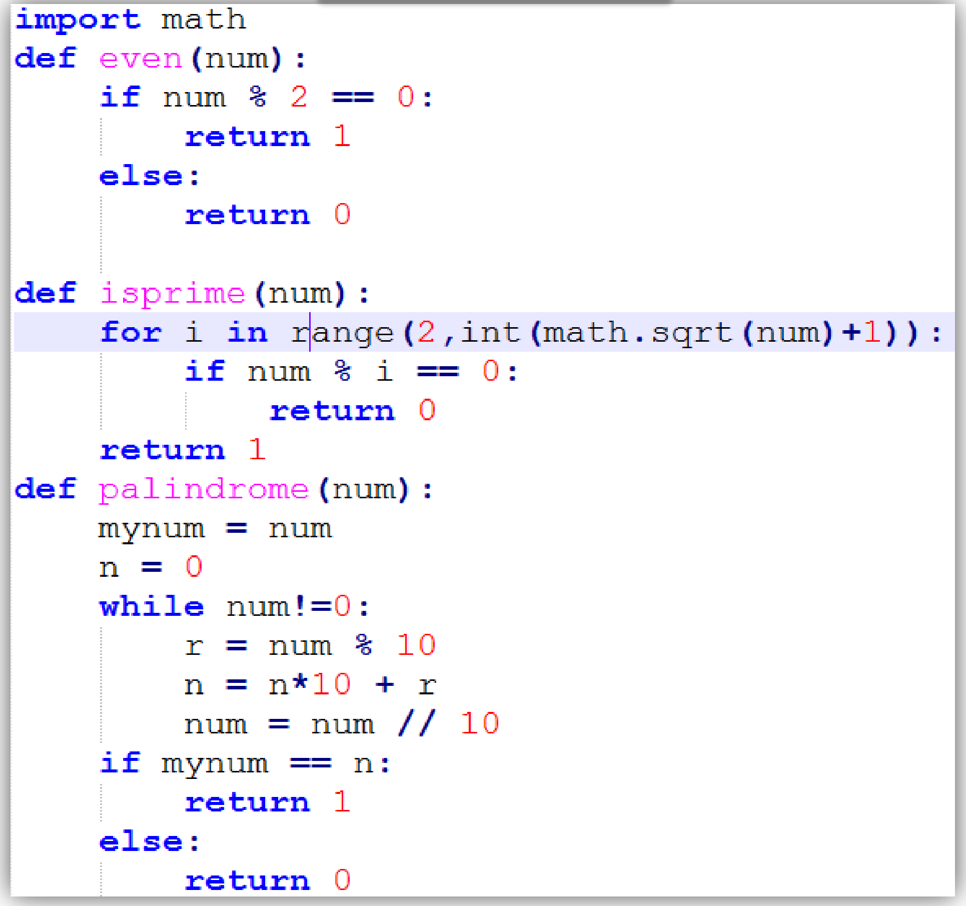


```
import math
# the is my first module
mynum=100
def area_rect(length,breadth):
    return length*breadth

def area_square(side):
    return side*side

def area_circle(rad):
    return math.pi*rad*rad
```

numcheck.py

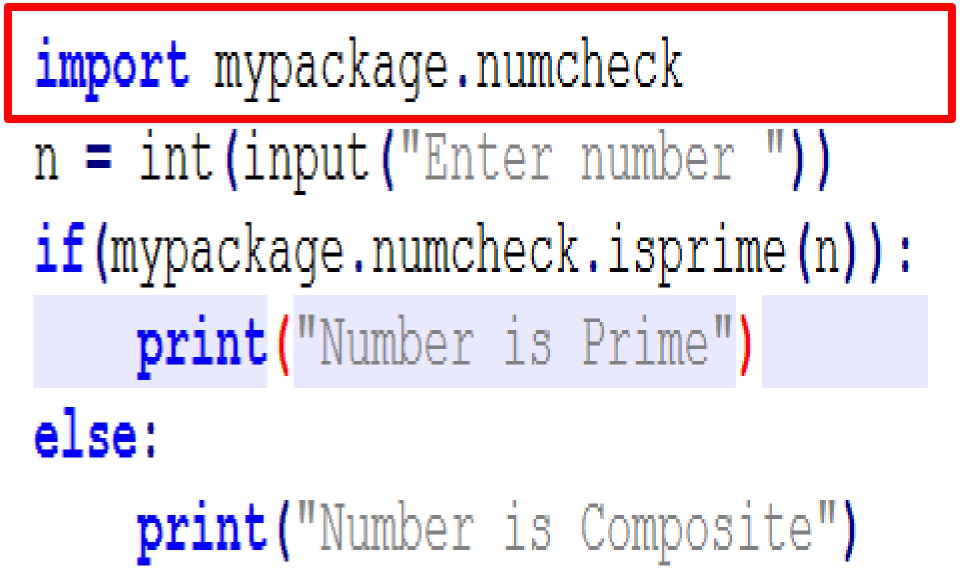


```
import math
def even(num):
    if num % 2 == 0:
        return 1
    else:
        return 0

def isprime(num):
    for i in range(2,int(math.sqrt(num)+1)):
        if num % i == 0:
            return 0
    return 1
def palindrome(num):
    mynum = num
    n = 0
    while num!=0:
        r = num % 10
        n = n*10 + r
        num = num // 10
    if mynum == n:
        return 1
    else:
        return 0
```

Creating Package

- Step 2: importing package and modules in python program



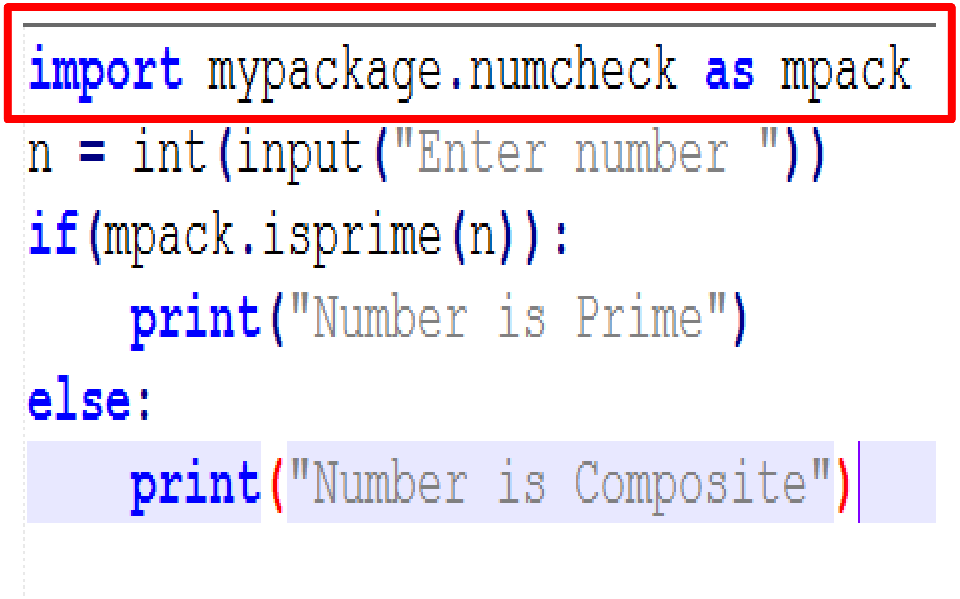
```
import mypackage.numcheck
n = int(input("Enter number "))
if(mypackage.numcheck.isprime(n)):
    print("Number is Prime")
else:
    print("Number is Composite")
```

Save this file by "anyname.py" outside the package folder

RUN THE PROGRAM

Creating Alias of Package/module

- Alias is the another name for imported package/module. It can be used to shorten the package/module name



```python
import mypackage.numcheck as mpack
n = int(input("Enter number "))
if(mpack.isprime(n)):
    print("Number is Prime")
else:
    print("Number is Composite")
```

Save this file by “anyname.py” outside the package folder

RUN THE PROGRAM

__init__.py file

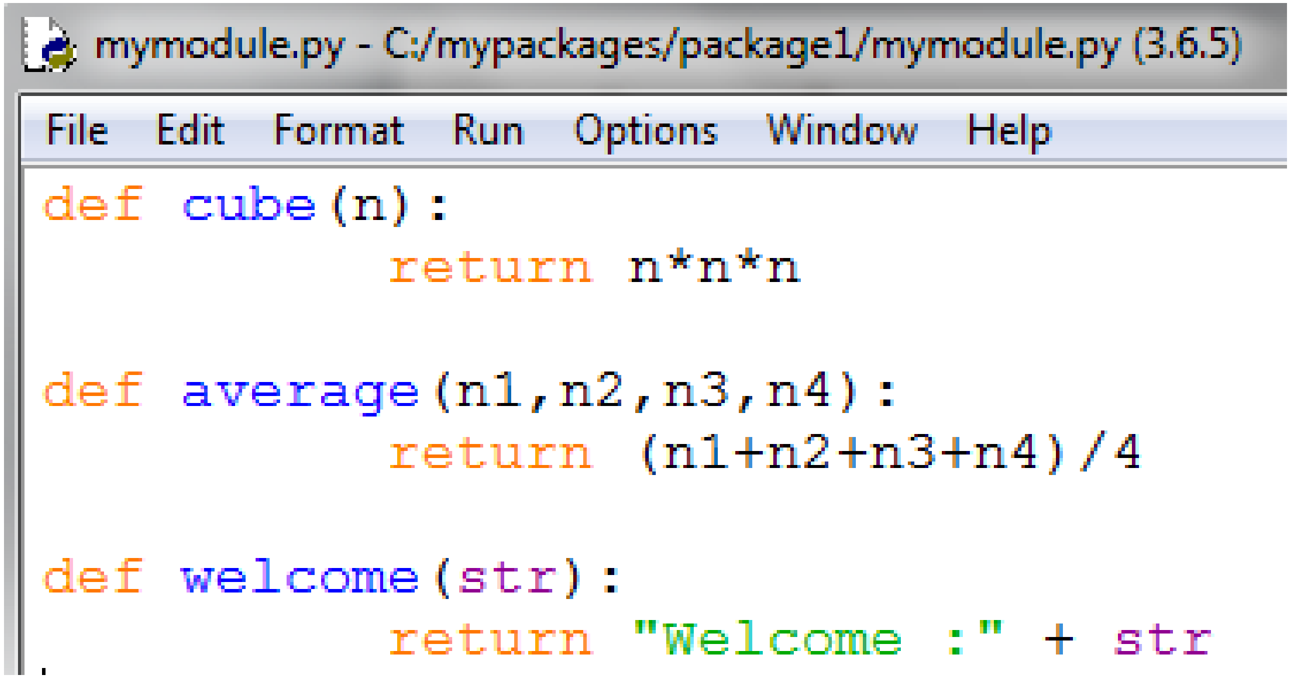
- **__init__.py** (double underscore is prefixed and suffixed) : This file is required to make Python treat directories containing the file as packages.
- **__init__.py** can be just empty file, but it can also execute initialization code for the package.
- **Note: Python 2 requires __init__.py to be inside a folder in order for the folder to be considered a package and made importable but in Python 3.3 and above, it support implicit namespace packages, all folders are packages regardless of the presence of a __init__.py file**
- **So from Python 3.3 it is optional to create __init__.py file**

Example - creating and using package and module and __init__.py

- Create a folder to act as a package
- For. e.g. In C: (C Drive) a folder “mypackages” is created
- Create __init__.py file (do not write any thing in it) in this folder “mypackages”
- Now create another folder “package1” inside “mypackages”
- Create __init__.py file inside “package1” also
- Create a module(.py) for e.g. “mymodule.py” file in “package1” to have some functions in it.

Another example of creating and using package and module and __init__.py

- mymodule.py



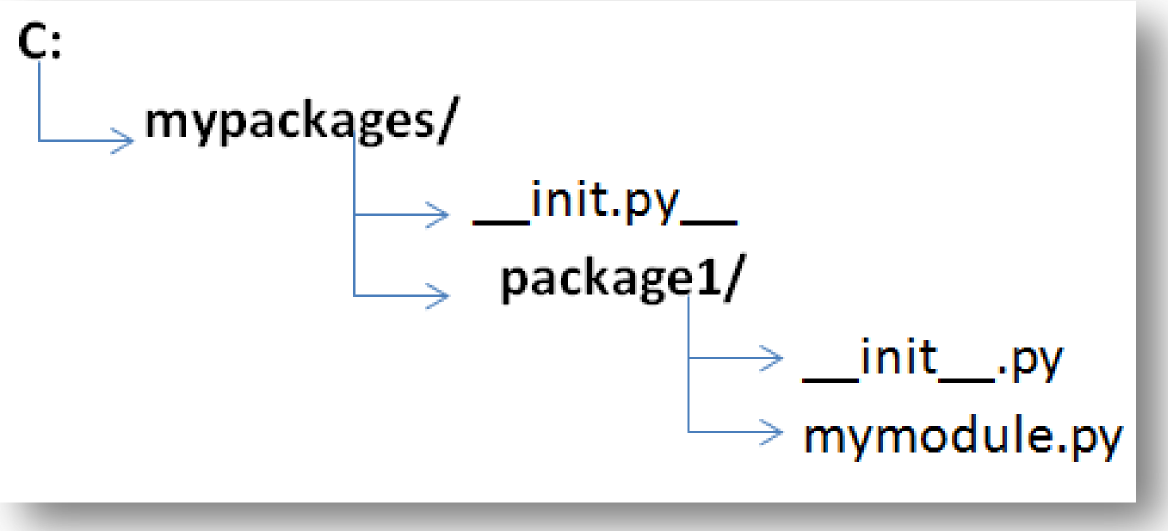
```
def cube(n):
        return n*n*n

def average(n1,n2,n3,n4):
        return (n1+n2+n3+n4)/4

def welcome(str):
        return "Welcome :" + str
```

Another example of creating and using package and module and __init__.py

- □ Now our entire contents will be like this:



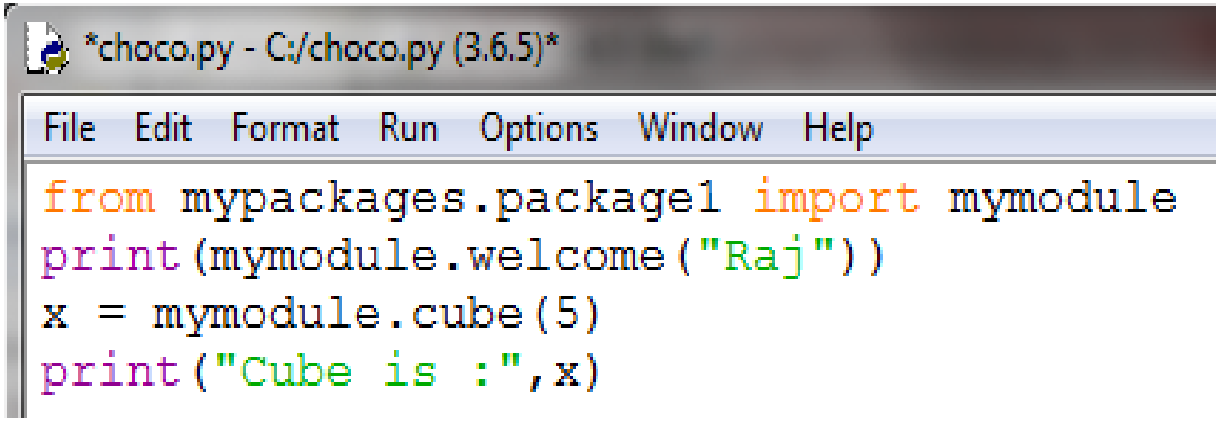
```
C:
 └──> mypackages/
          ├──> __init.py__
          └──> package1/
                   ├──> __init__.py
                   └──> mymodule.py
```

Here, **mypackages** and **package1** are folders

- □ Now in C: (C Drive), create a file for e.g. “choco.py” in which we will import module “mymodule”

Another example of creating and using package and module and __init__.py

- To import the packages, we can either use **Absolute addressing** or **Relative Addressing.** (already discussed in data file handling chapter)
- Absolute means following the complete address whereas Relative means with relation to current folder by using Single dot(.)

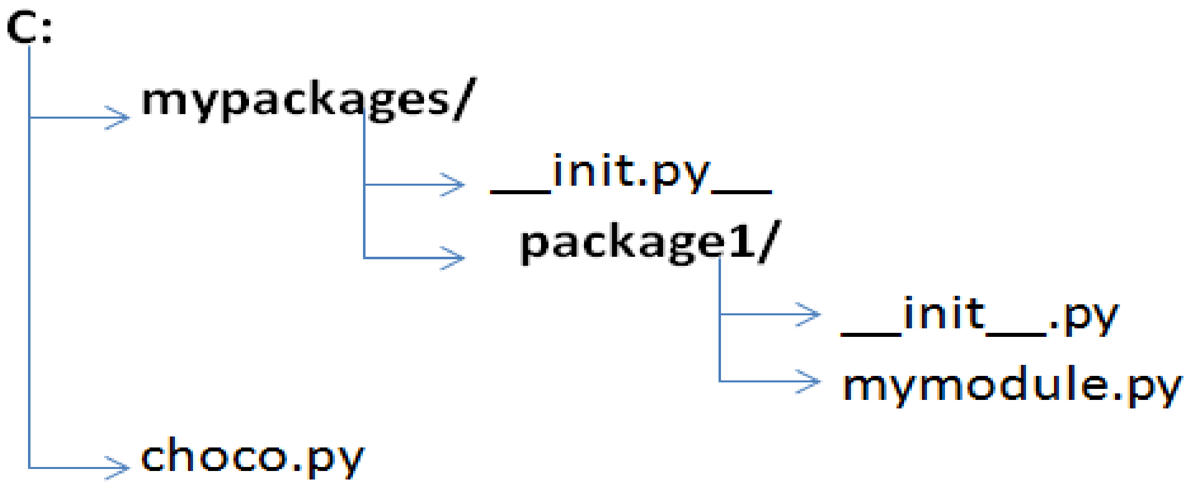

choco.py - C:/choco.py (3.6.5)
File Edit Format Run Options Window Help

```
from mypackages.package1 import mymodule
print(mymodule.welcome("Raj"))
x = mymodule.cube(5)
print("Cube is :",x)
```

Absolute path of package

Another example of creating and using package and module and __init__.py

- Now our entire content structure will be as:



```
C:
 |
 |---> mypackages/
 |           |
 |           |---> __init.py__
 |           |---> package1/
 |                        |
 |                        |---> __init__.py
 |                        |---> mymodule.py
 |
 |---> choco.py
```

- Run the choco.py file and you will get the output.

TIP : to use package from any location

- At this time, the package “mypackages” will be accessible from its location only i.e. the file which wants to import this package must be in same folder/drive where “mypackages” is.
- To enable “mypackages” to be used from any location, Copy this mypackages to Python’s **site-packages** folder inside **Lib** folder of Python’s installation folder. (*Try with copying to Lib folder also*)
- Path of site-packages is :

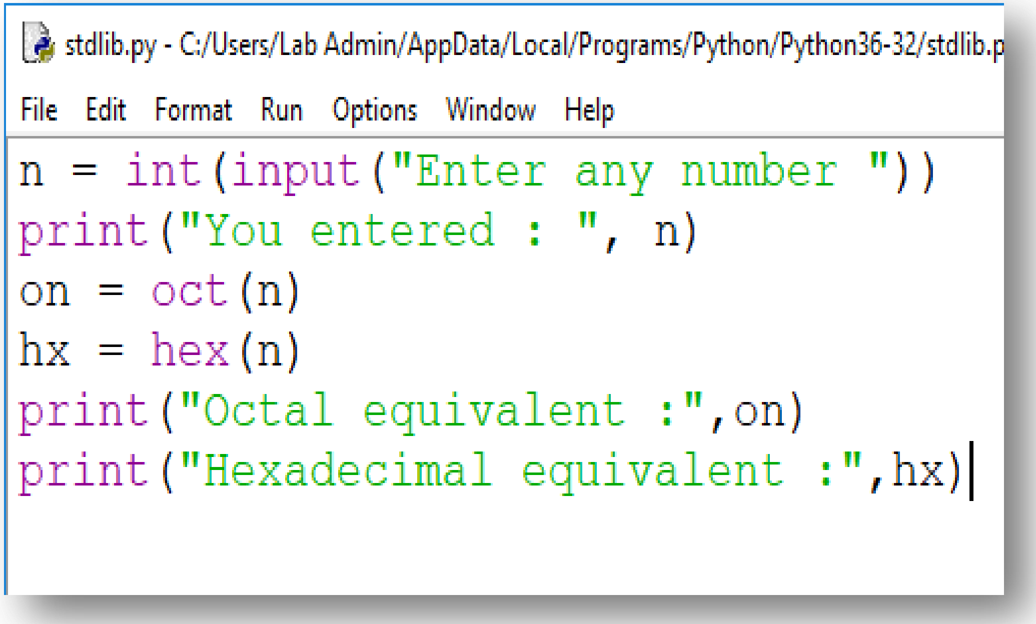
  C:\Users\vin\AppData\Local\Programs\Python\Python 36-32\Lib\site-packages
- After this you can import this package “mypackages” in any python file.

Using Python's Built-in Function

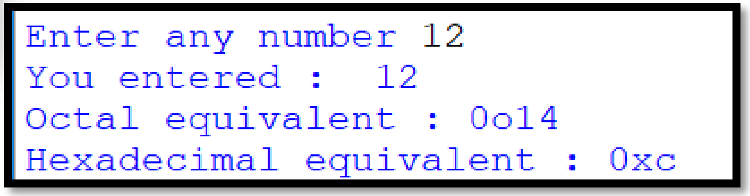
- Python's standard library is very extensive that offers many built-in functions that **we can use without having to import any library.**
- Using Python's Built-in functions
  - Function_name()

Mathematical and String functions

- oct(int) : return octal string for given number by prefixing “0o”
- hex(int) : return octal string for given number by prefixing “0x”


stdlib.py - C:/Users/Lab Admin/AppData/Local/Programs/Python/Python36-32/stdlib.p
File Edit Format Run Options Window Help

```
n = int(input("Enter any number "))
print("You entered : ", n)
on = oct(n)
hx = hex(n)
print("Octal equivalent :",on)
print("Hexadecimal equivalent :",hx)
```

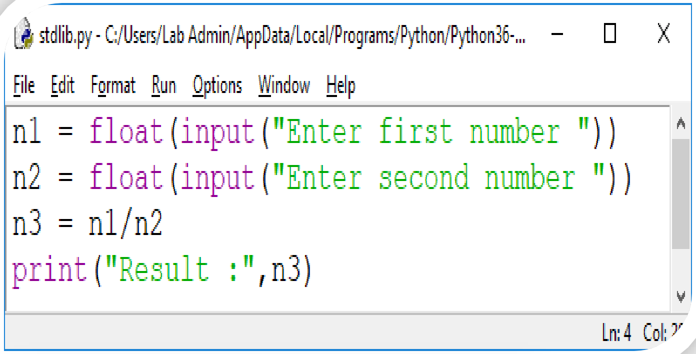


```
Enter any number 12
You entered :  12
Octal equivalent : 0o14
Hexadecimal equivalent : 0xc
```

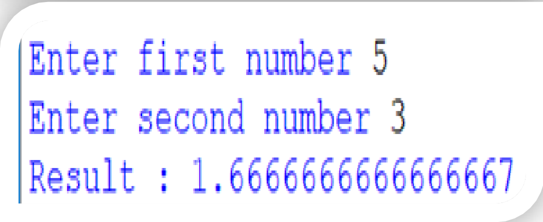
Mathematical and String functions

- int(number) : function convert the fractional number to integer
- int(string) : convert the given string to integer
- round(number,[nDIGIT]) : return number rounded to nDIGIT after decimal points. If nDIGIT is not given, it returns nearest integer to its input.
- Examples: (next slide)

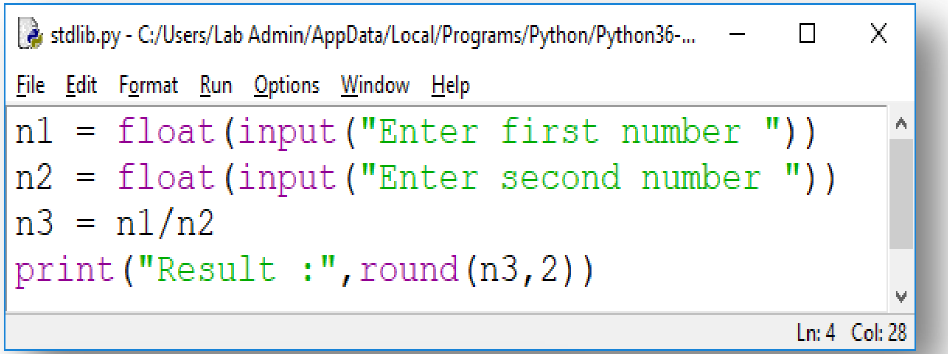
Mathematical and String functions


stdlib.py - C:/Users/Lab Admin/AppData/Local/Programs/Python/Python36-...
File Edit Format Run Options Window Help
Ln: 4 Col: 2

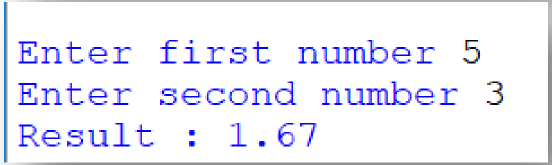
```
n1 = float(input("Enter first number "))
n2 = float(input("Enter second number "))
n3 = n1/n2
print("Result :",n3)
```



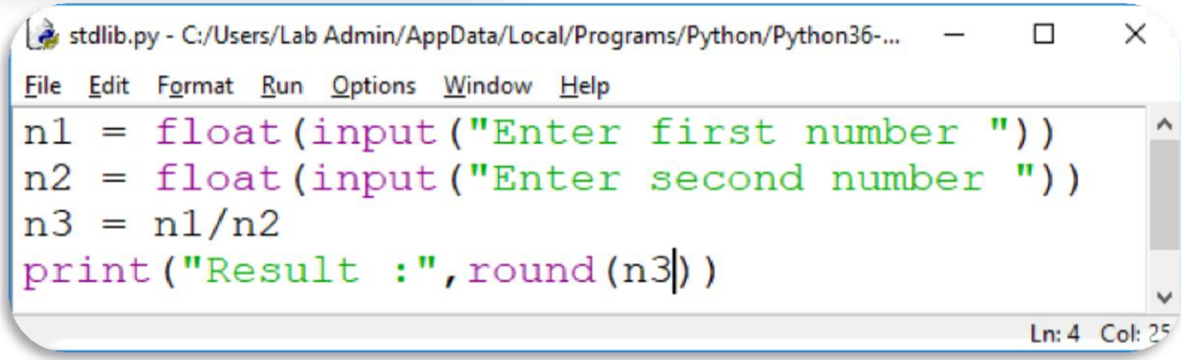
```
Enter first number 5
Enter second number 3
Result : 1.6666666666666667
```


stdlib.py - C:/Users/Lab Admin/AppData/Local/Programs/Python/Python36-...
File Edit Format Run Options Window Help
Ln: 4 Col: 28

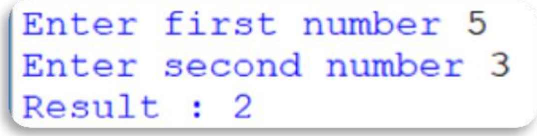
```
n1 = float(input("Enter first number "))
n2 = float(input("Enter second number "))
n3 = n1/n2
print("Result :",round(n3,2))
```



```
Enter first number 5
Enter second number 3
Result : 1.67
```


stdlib.py - C:/Users/Lab Admin/AppData/Local/Programs/Python/Python36-...
File Edit Format Run Options Window Help
Ln: 4 Col: 25

```
n1 = float(input("Enter first number "))
n2 = float(input("Enter second number "))
n3 = n1/n2
print("Result :",round(n3))
```

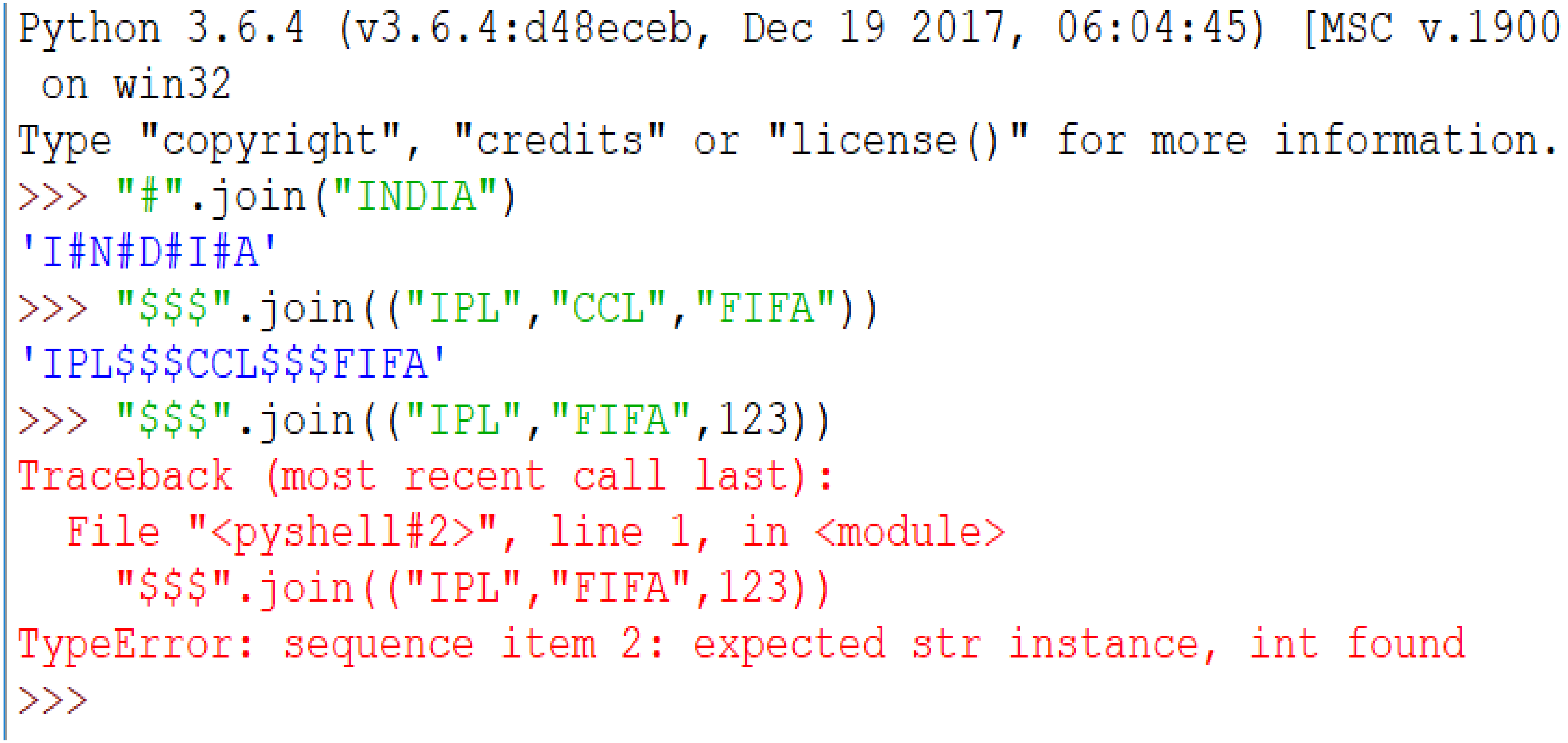


```
Enter first number 5
Enter second number 3
Result : 2
```

Other String function

- We have already used many string function in class XI, here are few new functions
  - <string>.join() : if the string based iterator is a string then the <string> is inserted after every character of the string.
    - If the string based iterator is a list or tuple of strings then, the given string/character is joined after each member of the list of tuple. **BUT the tuple or list must have all members as string otherwise Python will raise an error**
- **Examples (next slide)**

Other String function

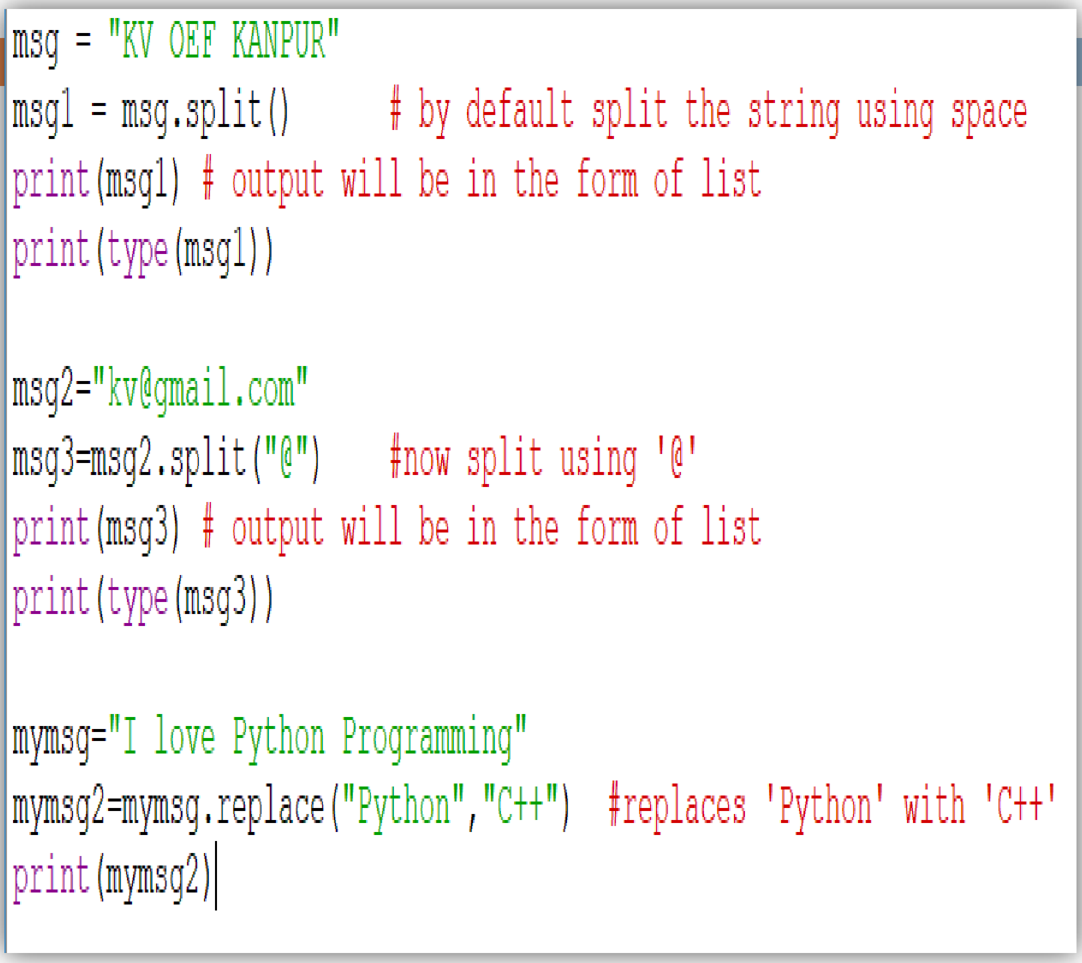


```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> "#".join("INDIA")
'I#N#D#I#A'
>>> "$$$".join(("IPL","CCL","FIFA"))
'IPL$$$CCL$$$FIFA'
>>> "$$$".join(("IPL","FIFA",123))
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    "$$$".join(("IPL","FIFA",123))
TypeError: sequence item 2: expected str instance, int found
>>>
```

Other String function

- We have already used many string function in class XI, here are few new functions
  - <string>.split() : allow to divide string in multiple parts and store it as a LIST. If you do not provide delimeter then by default string will be split using space otherwise using given character.
  - <str>.replace() : **allows you to replace any part of string with another string.**
  - **Example (NEXT SLIDE)**

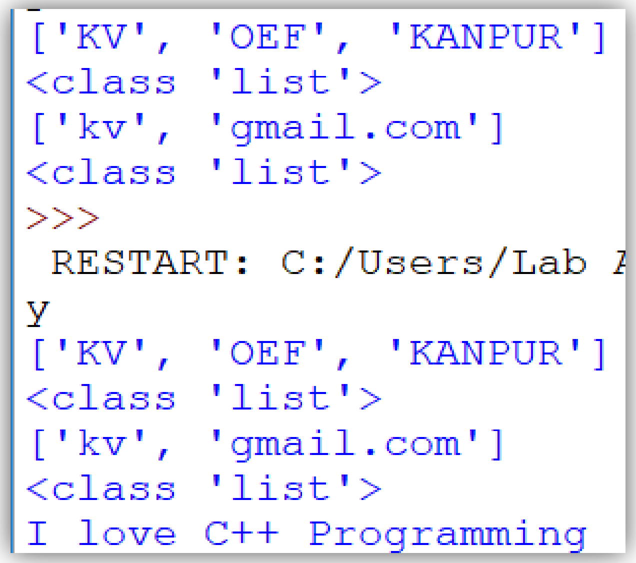
Example (split() and replace())



```
msg = "KV OEF KANPUR"
msg1 = msg.split()      # by default split the string using space
print(msg1) # output will be in the form of list
print(type(msg1))

msg2="kv@gmail.com"
msg3=msg2.split("@")    #now split using '@'
print(msg3) # output will be in the form of list
print(type(msg3))

mymsg="I love Python Programming"
mymsg2=mymsg.replace("Python","C++")  #replaces 'Python' with 'C++'
print(mymsg2)
```



```
['KV', 'OEF', 'KANPUR']
<class 'list'>
['kv', 'gmail.com']
<class 'list'>
>>>
 RESTART: C:/Users/Lab A
y
['KV', 'OEF', 'KANPUR']
<class 'list'>
['kv', 'gmail.com']
<class 'list'>
I love C++ Programming
```

Using **URLLIB** and **WEBBROWSER** modules

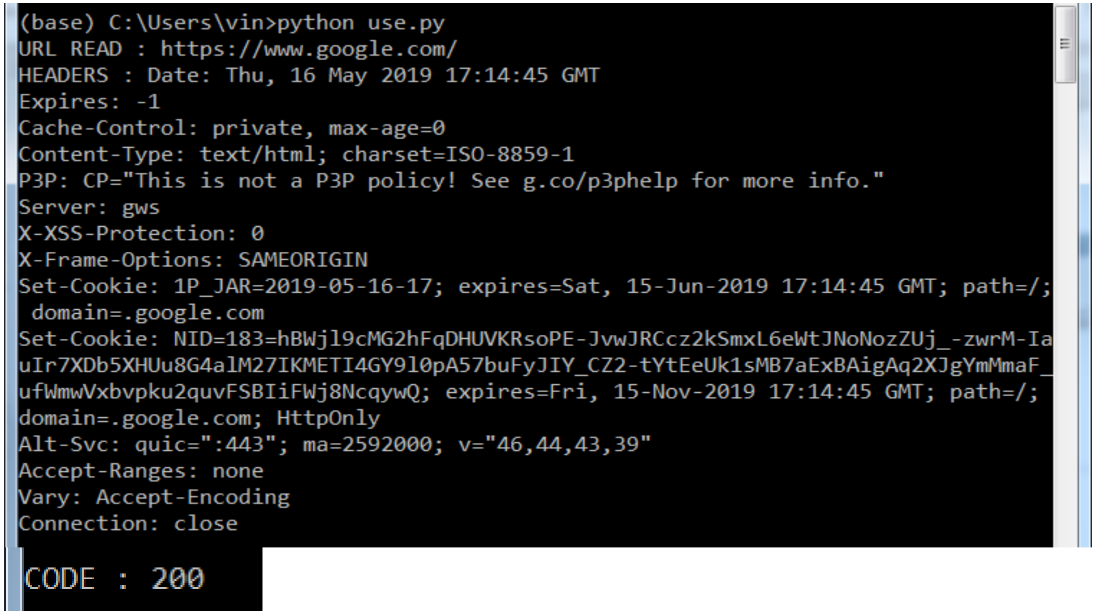
- □ Python provides **urllib** module to send http request and receive the result from within your program. To use **urllib** we have to first import **urllib** module.
- □ **urllib** module is a collection of sub-module like request, error, parse etc. following functions of **urllib** we can use: **(next slide)**

Functions of URLLIB

FUNCTION NAME	PURPOSE
urllib.request.urlopen(<url>)	Opens a website or network object denoted by URL for reading and return file like object using which other functions are often used
urlopen_object.read()	Return HTML or the source code of given url
urlopen_object..getcode()	Returns HTTP status code where 200 means ‘all okay’ 404 means url not found etc. 301, 302 means some kind of redirections happened
urlopen_object.headers	Stores metadata about the open URL
urlopen_object.info()	Returns same information as by headers
urlopen_object.geturl()	Return URL string

Example:

```
import urllib.request
x=urllib.request.urlopen('https://www.google.com/')
print("URL READ :",x.geturl())
print("HEADERS :",x.headers)
print("INFO :",x.info())
print("CODE :",x.getcode())
```



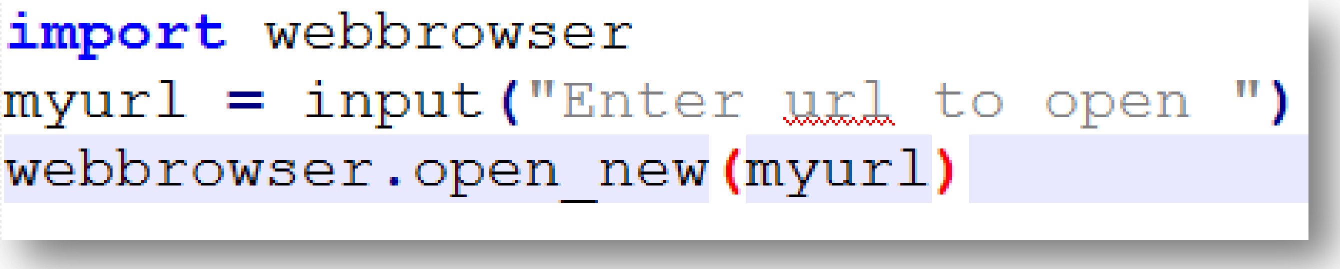
```
(base) C:\Users\vin>python use.py
URL READ : https://www.google.com/
HEADERS : Date: Thu, 16 May 2019 17:14:45 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2019-05-16-17; expires=Sat, 15-Jun-2019 17:14:45 GMT; path=/;
 domain=.google.com
Set-Cookie: NID=183=hBWjl9cMG2hFqDHUVKRsoPE-JvwJRCcz2kSmxL6eWtJNoNozZUj_-zwrM-Ia
uIr7XDb5XHUu8G4alM27IKMETI4GY9l0pA57buFyJIY_CZ2-tYtEeUk1sMB7aExBAigAq2XJgYmMmaF_
ufWmwVxbvpku2quvFSBIiFWj8NcqywQ; expires=Fri, 15-Nov-2019 17:14:45 GMT; path=/;
domain=.google.com; HttpOnly
Alt-Svc: quic=":443"; ma=2592000; v="46,44,43,39"
Accept-Ranges: none
Vary: Accept-Encoding
Connection: close

CODE : 200
```

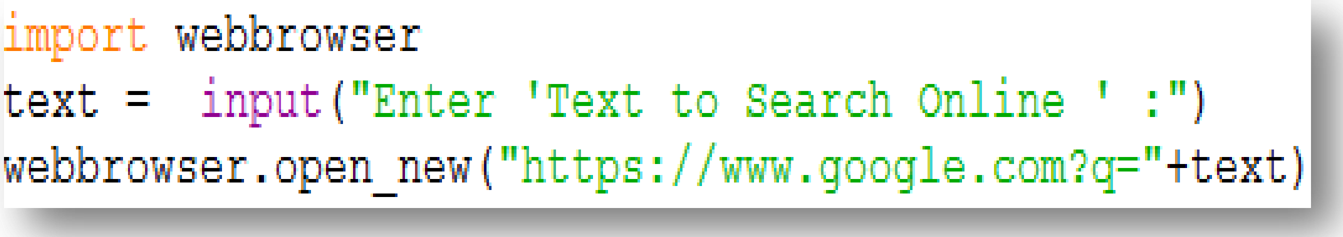
Before execution of the above code, make sure PC is connected to working internet.

WEBBROWSER MODULE

- Provides functionality to open a website in a window or tab or web browser on your computer, from within your program.
- To use webbrowser module we must import the module as:
  - **import webbrowser**



```
import webbrowser
myurl = input("Enter url to open ")
webbrowser.open_new(myurl)
```



```
import webbrowser
text =  input("Enter 'Text to Search Online ' :")
webbrowser.open_new("https://www.google.com?q="+text)
```