PYTHON LIBRARIES

Module

- Modularity-it means act of portioning a program into individual components
- Modules logically organize our python code
- In module we are grouping related code into a module makes the code easier to understand.
- Module can define funcitons,variables and classes
- A module is a normal .py file.
  - E.g. module math is math.py file
    - Module random is random.py file

Library

- Library is a collection of modules
- Library caters to specific type of requirement.
- Python standard library-math,random,matplotlib,tkinter,numpy,urllib

Structure of Python Module

1. Docstring-triple quoted comments(“”” “”” or ‘’’ ‘’’) ,useful for documentation , used in a function/class or before the function/class.
2. Variables and constants-labels for data.
3. Classes-blueprint to create objects
4. Objects-instances of classes
5. Statements-instructions
6. Functions-named group of instructions

Example module:fun.py

```
""" function to display sum of 2 nos"""
def sum(a,b):
    """accepts 2 nos from the user to calculate sum"""
    return a+b

def diff(a,b):
    """accepts 2 nos from the user to calculate subtraction of 2 nos"""
    return a-b
```

dir () and help()

python will display all docstring along with module name, filename, functions name and constant

- import math
- dir(math)

['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

- help(pow)

Help on built-in function pow in module builtins:

pow(base, exp, mod=None)

Equivalent to base**exp with 2 arguments or base**exp % mod with 3 arguments

Some types, such as ints, are able to use a more efficient algorithm when

invoked using the three argument form.

Namespace

It is a space tht holds a bunch of names.

e.g. three stream (commerce,humanities,science)

50 students each

commom name-deepak (in each stream)

three namespaces

Rules are same as LEGB(local embedded global built-in),

(*discussed in function chapter…for more see function ppt on blog*)

Importing modules in a python program-3 ways

1. **Importing entire module**
2. **Importing selected function/object from a module**
3. **Importing all function/obejcts of a module**

First method of importing module

1. **Importing entire module**

```
import math
print(math.sqrt(4))
```

output: 2

Dot operator

Processing of Import <module>

1. Code of imported module is interpreted and executed
2. Defined functions and variables created in the module are now available
3. A new namespace is setup(same name as module)

Second method of importing module

2. **Importing selected function/object from a module**

```
from math import sqrt
print(sqrt(4))                    output:2

from math import sqrt,pow
print(sqrt(4))                    output:2
print(pow(3,2))                   output:9
```

Processing of from <module> import <object/function>

1. Code of imported module is interpreted and executed
2. Only asked function and variables are now available
3. **No new namespace** is created (imported definition is added in the current namespace).

E.g.

```
from math import sqrt
d=sqrt(4)
print(d)
```

output:2

Third method of importing module

3. **Importing all function/obejcts of a module**

```
from math import *
print(sqrt(4))          output:2
print(pow(3,2))         output:9
```

Using Python Built-in functions

```
x=4.451
print(int(x))                output:4

x=4.451
print(round(x,1))            output:4.5

x=11
print(oct(x))                output: '0o13'
print(hex(x))                output: '0xb'
```

Using Python string Built-in functions

<string>.join()

print("**".join("hello")) output: h**e**l**l**o

```
d=["hello","world"]
z="**".join(d)
print(z)
```

output : hello**world

OR

```
s="**"
d=["hello","world"]
print(x.join(d))
```

output : hello**world

Note: we can use only tuple,we can't use dictionary. Given sequence should have all values as string

Using string Built-in functions

<string>.split()

x="hello world"

print(x.split()) output: ['hello','world']

---

x="hello world"

print(x.split('o')) output : ['hell', ' w', 'rld']

Using string Built-in functions

<string>.replace()

x="hello world"

print(x.replace ("hello","outer")) output: 'outer world'

OR

"hello to class xii ".replace("hello","welcome") output : 'welcome to class xii '

Using random module

1. **random()**
   - **it returns a random number in range(0.0 - 1.0).**
   - **In this lower range limit is inclusive.**
   - **Returns floating point number.**

```
import  random
print(random.random())
```

output : 3.4100256441147248...

17.519619328076985....

Using random module

2. **randint()**
   - **it returns a random number in given range.**
   - **both range limit are inclusive.**
   - **Returns integer number.**

```
import  random
print(random.randint(4,20))          output : 7
print(random.randint(4,20))          output : 4
                      OR
D=random.randint(4,20))
print(D)                             output : 13
```

Using Webbrowser module

Open()

- **it opens a website in a window in computer**

```
import  webbrowser
webbroswer.open("www.google.com")
```

Using urllib module

urllib.request.urlopen(<url>)

.read()

.getcode()

.geturl()

.headers

.info()

Using urllib module

urllib.request.urlopen(<url>)- it open a website ,for reading and returns a file using which other functions will be used.

<urlopen'sreturnval>.read()-returns html or the source code of the url

<urlopen'sreturnval>.getcode()-returns http status code

<urlopen'sreturnval>.geturl()-returns url

<urlopen'sreturnval>.headers-stores metadata about the opened URL

<urlopen'sreturnval>.info()-returns same information as stored by headers

Using urllib module

```
import urllib.request
import webbrowser
wu=urllib.request.urlopen("https://www.google.com/")
h=wu.read()
data=wu.getcode()
url=wu.geturl()
head=wu.headers
inf=wu.info()
print(h)
print(data)
print(url)
print(head)
print(inf)
webbrowser.open(url)
```

OUTPUT:

```
200
https://www.google.com/
Date: Fri, 10 Jul 2020 10:56:11 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=2020-07-10-10; expires=Sun, 09-Aug-2020 10:56:11 GMT; path=/;
domain=.google.com; Secure
Set-Cookie:
NID=204=IeEBYQGsj9b7SAcxF_mx4deohaQU_SiiH53xFrw5waKhXYeclSfKG2MdE0GFN8jENMIc4CfrEwB
CSV3bwWHF-
CFKGCYKFy9cLzpLAQZpIWUE6oaxP5ueTwiZW8DMcSviubcN6yypKT54KLJNttyjrwNEoa5IM7ahfW8BUHM
bytw; expires=Sat, 09-Jan-2021 10:56:11 GMT; path=/; domain=.google.com; HttpOnly
Alt-Svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-25=":443"; ma=2592000,h3-
T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443";
ma=2592000,quic=":443"; ma=2592000; v="46,43"
Accept-Ranges: none
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked


Date: Fri, 10 Jul 2020 10:56:11 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=2020-07-10-10; expires=Sun, 09-Aug-2020 10:56:11 GMT; path=/;
domain=.google.com; Secure
Set-Cookie:
NID=204=IeEBYQGsj9b7SAcxF_mx4deohaQU_SiiH53xFrw5waKhXYeclSfKG2MdE0GFN8jENMIc4CfrEwB
CSV3bwWHF-
CFKGCYKFy9cLzpLAQZpIWUE6oaxP5ueTwiZW8DMcSviubcN6yypKT54KLJNttyjrwNEoa5IM7ahfW8BUHM
bytw; expires=Sat, 09-Jan-2021 10:56:11 GMT; path=/; domain=.google.com; HttpOnly
Alt-Svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-25=":443"; ma=2592000,h3-
T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443";
ma=2592000,quic=":443"; ma=2592000; v="46,43"
Accept-Ranges: none
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
```

Creating a Python Library

Library and package terms are used interchangeably

Package:

- A package is a collection of python modules.
- Common namespace
- Packages place different modules on a single directory.
  - _init_.py must be present.

Procedure for Creating a Python Library

1. Find site package folder in python installation folder.
2. Create a folder e.g. abc(library/package)
3. Create _init_.py file in package.
4. Create module e.g. xyz.py
5. Write functions in xyz.py
6. Import a module from library/package

    from abc import xyz

sys module

```
import sys
print(sys.path)
```

OUTPUT

```
'C:/Users/user/AppData/Local/Programs/Python/Python38-32',
'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python38-32\\Lib\\idlelib',
'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python38-32\\python38.zip',
'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python38-32\\DLLs',
'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python38-32\\lib',
'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python38-32',
'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python38-32\\lib\\site-packages']
```