In [ ]:
```
#Day-12
  #Today Agenda:
                1. Packages and Modules in Python
                2. Regular Expressions
                3. Problem Solving.
```

In [ ]:
```
#Module: Its Nothing but a python file containing python statements,and definitions,functions..etc
simple its a .py file.
1. User Defined Modules - Created by Users or Programmers
2. Pre Defined Modules  - Created by Developers at Construsting the Code
```

In [1]:
```
#How to create user defined module.
def evenadd(n):
    if n %2==0:
        print("Even")
    else:
        print("Odd")
evenadd(6)
```

```
Even
```

In [7]:
```
#How to Use user defined module?
#By using import statement we can use user defined modules.
import even                                          #import module name
even.evenodd(n=int(input("enter a number")))         #modulename.functionname(parameters)
```

```
enter a number6
```

Out[7]:
```
True
```

In [9]:
```
#Import a module
import calc
calc.mul(3,8)
```

Out[9]:
```
24
```

In [13]:
```
from calc import add,power,sub,mul     #from calc import * - for import all
print(calc.add(5,6))
print(power(2,5))
print(calc.sub(100,90))
print(calc.mul(10,9))
```

```
11
32
10
90
```

In [27]:
```
import fact1  #from fact1 import ispalindrome,factorial, from fact1 import *
print(fact1.factorial(6))
print(fact1.factorial(9))
print(fact1.ispalindrome("apssdc"))
print(fact1.ispalindrome("surya"))
print(fact1.ispalindrome("madam"))
```

```
720
362880
False
False
True
```

In [28]:
```
#Search Path for PreDefined Modules
import sys
sys.path
```

Out[28]:
```
['C:\\Users\\Mission Impossible\\Desktop\\Python-Online Workshop Content\\Day-6 (Functions And Problem Solvin
g) [29-08-2020]',
 'C:\\ProgramData\\Anaconda3\\python37.zip',
 'C:\\ProgramData\\Anaconda3\\DLLs',
 'C:\\ProgramData\\Anaconda3\\lib',
 'C:\\ProgramData\\Anaconda3',
 '',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\win32',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\win32\\lib',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\Pythonwin',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\IPython\\extensions',
 'C:\\Users\\Mission Impossible\\.ipython']
```

In [ ]:
```
#Predefined Modules:
1. math
2. os module
3. random
4. sys
5. statistics
```

In [38]:
```
#math:
import math
print(math.degrees(2))
print(math.sqrt(64))
print(math.radians(60))
print(math.sin(2))
print(math.cos(0.5))
print(math.tan(0.35))
print(math.factorial(5))
```

```
114.59155902616465
8.0
1.0471975511965976
0.9092974268256817
0.8775825618903728
0.36502849483042454
120
```

In [46]:
```
#time
from datetime import date
#print(time.ctime())
print(date.fromtimestamp(6574748494))
```

```
2178-05-06
```

In [53]:
```
#statistics module:
import statistics
print(statistics.mean([3,4,5,6,6,7,7,8,9,2,2,3,34,55]))
print(statistics.median([23,4,5,5,6,76,77,7,9]))
print(statistics.mode([2,2,2,3,3,3,5,5,5,5,6,7,8,9]))
print(statistics.stdev([23,56,89,90,23,90]))
```

```
10.785714285714286
7
5
32.786684289001634
```

In [ ]:
```
#How to create user defined packages?
  A Package can contains sub packages and any number of modules.
```

In [66]:
```
#import user defined package:
#from packagename import modulename
from userPackage import functions3
print(functions3.cube(5))
print(functions3.sqrt(100))
```

```
125
10000
```

In [71]:
```
#We can view the all directories of User Defined Packages and Modules
print(dir(functions3),end=" ")
print(functions3.__doc__)
print(functions3.__package__)
print(functions3.__file__)
print(functions3.__name__)
```

```
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'cu
be', 'even', 'sqrt'] None
userPackage
C:\Users\Mission Impossible\Desktop\Python-Online Workshop Content\Day-6 (Functions And Problem Solving) [29-0
8-2020]\userPackage\functions3.py
userPackage.functions3
```

In [61]:
```
print(dir(list),end=" ")
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__form
at__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str_
_', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'rev
erse', 'sort']
```

In [ ]:
```
#Regular Expressions:
What is regular expressions?
  - Its a sequence of characters that defines a search pattern.
  - Its called as RegEx or re module
#What is the use of re module?
  - It can be used to search, edit and manipulate text/string.
```

In [72]:
```
#How to use the re module?
import re
  # search pattern can be formed along with some rules:
      those rules are including
          1. Meta Characters - [],.,^,%,+,?,{},(),\,|

          2. Special Characters - \A,\a,\S,\s,\B,\d,|D...etc

          3. sets ------ [a-z],{0-9}...etc
```

In [ ]:
```
#MetaCharacters:
#Character        Description

[]               A set of characters
\                Signals a special sequence (can also be used to escape special characters.
.                Any character (except newline character)
^                Starts with "^hello"
$                Ends with   "world$"
*                Zero or more occurrences    "aix*"
+                One or more occurrences    "aix+"
{}               Exactly the specified number of occurrences    "al{2}"
|                Either or  "falls|stays"
()               Capture and group
```

In [ ]:
```
#Special Sequences:
#Character                          Description
\A                      Returns a match if the specified characters are at the beginning of the string.
\b                 Returns a match where the specified characters are at the beginning or at the end of a word
                                    (the "r" in the beginning is making sure that the string is being treated as

\B                                   Returns a match where the specified characters are present,
                                      but NOT at the beginning (or at the end) of a word
                                      (the "r" in the beginning is making sure that the string is being treated as
\d                                    Returns a match where the string contains digits (numbers from 0-9)
\D                                  Returns a match where the string DOES NOT contain digits
\s                              Returns a match where the string contains a white space character
\S                         Returns a match where the string DOES NOT contain a white space character
\w                                     Returns a match where the string contains any word characters
                                       (characters from a to Z, digits from 0-9, and the underscore _ character)
\W                                     Returns a match where the string DOES NOT contain any word characters
\Z                                     Returns a match if the specified characters are at the end of the string.
```

In [ ]:
```
#Sets
A set is a set of characters inside a pair of square brackets [] with a special meaning:

#Set                            Description
[arn]                       Returns a match where one of the specified characters (a, r, or n) are present
[a-n]                       Returns a match for any lower case character, alphabetically between a and n
[^arn]                      Returns a match for any character EXCEPT a, r, and n
[0123]                      Returns a match where any of the specified digits (0, 1, 2, or 3) are present
[0-9]                       Returns a match for any digit between 0 and 9
[0-5][0-9]                  Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]                    Returns a match for any character alphabetically between a and z, lower case OR upper
[+]                         In sets, +, *, ., |, (), $,{} has no special meaning,
                            so [+] means: return a match for any + character in the string
```