##1. ARITHMETIC OPERATION

###Simple Arithmetic

In [1]:
```
a=5
b=6
c=a+b
print(c)
```

```
11
```

###Simple Arithmetic - Get input from User (+,-,*,%)

In [2]:
```
a = int(input("Enter the value of a: "))
b = float(input("Enter the value of b: "))
add=a+b
print(add)
```

```
Enter the value of a: 5
Enter the value of b: 55
60.0
```

In [8]:
```
a = int(input("Enter the value of a: "))
b = int(input("Enter the value of b: "))
add=a+b
sub=a-b
mult=a*b
div=a/b
mod=a%b
print("ADD:",add)
print("sub:", sub)
print("add: {0}, sub: {1}, Mult: {2}, Div:{3}, Mod:{4}".format(add,sub,mult,
```

```
Enter the value of a: 6
Enter the value of b: 5
ADD: 11
sub: 1
one: 11, two: 1, Mult: 30, Div:1.2, Mod:1
```

In [ ]:
```
print("Sum: %d, Diff: %d, Mult: %d, Div:%15.6f, Mod:%d"%(add,sub,mult,div,mo
```

In [ ]:
```
print('{0:<4} | {1:^4} | {2:^4} | {3:>4}'.format('Sum','Diff','Mult','Div'))
print('{0:<4} | {1:^4} | {2:^4} | {3:>4}'.format(add,sub,mult,div))
```

###Simple Arithmetic - Get input from User (power)

In [10]:
```
a=2
b=4
power=a**b #Power
print(power)
```

```
16
```

##2. STRINGS

###Create String

In [13]:
```
a = 'Python '
b = "Bootcamp"
print(a+b)
```

```
Python Bootcamp
```

###Length of String

In [14]:
```
a="Champ"
print(len(a))
```

```
5
```

In [15]:
```
a="S"
b=a*5
print(b)
```

```
SSSSS
```

###String Index

In [2]:
```
a="i am going to college!"
print(a[3]) #identifing the element based on index
print(a[2:]) #Grab the remaing elements except upto the Index
print(a[:2]) #Grab the elements upto the Index
print(a[-1]) #Grab the Last element
print(a[:-1]) #Grab the elements except last element
print(a[::2]) #Grab everything with 2 steps
print(a[::-1]) #Print string backwards


#[start:stop:step]
```

```
m
am going to college!
i
!
i am going to college
ia on oclee
!egelloc ot gniog ma i
```

###String Functions

In [29]:
```
a="Master Class"
print(a.upper()) #Changing to Upper case
print(a.lower()) #Changing to Lowerb= a.split() case
```

```
MASTER CLASS
master class
```

In [33]:
```
b= a.split() #Splitting String
print(b)
print(b[0])#Printing the splitting string based on Index
```

```
['Master', 'Class']
Master
```

In [34]:
```
c="ElonMusk,SteveJobs,BillGates"
d=c.split(",") #Splitting string based on Delimitter
print(d)
print(d[1])
```

```
['ElonMusk', 'SteveJobs', 'BillGates']
SteveJobs
```

In [35]:
```
a="Master Class"
print(f"Welcome to Python {a} !") #Formatting string Literals
```

```
Welcome to Python Master Class !
```

##3. LIST

###Create List

In [4]:
```
a = [1,2,3,4,5]
b = ["Champ",21,99.5]
print(a,b)
print(len(b)) #Length of List
print(b[0]) #Locate list element based on Index
print(a[1:]) #print elements except 1st
print(a[:2]) #Print elements upto 2nd element
print(a+b) #Concatenate 2 list
```

```
[1, 2, 3, 4, 5] ['Champ', 21, 99.5]
3
Champ
[2, 3, 4, 5]
[1, 2]
[1, 2, 3, 4, 5, 'Champ', 21, 99.5]
```

In [9]:
```
a = [1,2,3,4,5]
a.append(6) #inserting new elements to the existing list
print(a)
```

```
[1, 2, 3, 4, 5, 6]
```

In [10]:
```
a = [1,2,3,4,5]
a.reverse() #Reverse list
print(a)
print(min(a)) #Minimum
print(max(a)) #Maximum
```

```
[5, 4, 3, 2, 1]
1
5
```

In [11]:
```
a = [1,2,3,4,5]
from random import shuffle
shuffle(a)
a
```

Out[11]:
```
[1, 4, 5, 3, 2]
```

In [12]:
```
a=[1,2,3]
b=[4,5,6]
c=[a,b] #Nested List Matrix
print(c)
print(c[0]) #Printing row
print(c[0][0]) #Printing 1st element
```

```
[[1, 2, 3], [4, 5, 6]]
[1, 2, 3]
1
```

##4. DICTIONARIES

###Creating Dictionary : Key & Value

In [15]:
```
a = {"Name":"Elon","Age":50,"Company":["SpaceX","Tesla"]}
a
```

Out[15]:
```
{'Name': 'Elon', 'Age': 50, 'Company': ['SpaceX', 'Tesla']}
```

In [16]:
```
a = {"Name":"Elon","Age":50,"Company":["SpaceX","Tesla"]}
print(a["Name"]) #Printing value based on its Key
a["Age"]= 51 #Changing values
a
```

```
Elon
```

Out[16]:
```
{'Name': 'Elon', 'Age': 51, 'Company': ['SpaceX', 'Tesla']}
```

In [27]:
```
b={"Climate":{"Condition":{"Temperature":"38 Degree","Humidity":"70 Percenta
print(b["Climate"]["Condition"]["Temperature"])
```

```
38 Degree
```

In [28]:
```
print(a.keys()) #Printing Keys of the Dictionaries
print(a.values()) #Printing Values of the Dictionaries
print(a.items()) #Printing Tuple of the all items
```

```
dict_keys(['Name', 'Age', 'Company'])
dict_values(['Elon', 51, ['SpaceX', 'Tesla']])
dict_items([('Name', 'Elon'), ('Age', 51), ('Company', ['SpaceX', 'Tesla'])])
```

##5. TUPLES

###Creating Tuples

In [ ]:
```
a = ("Champ",21,99.5)
a
```

print(len(a)) #Length of tuple

#print(a[1])#Printing elements from tuple via index print(a.index("Champ")) #Getting Index of Elements

#index=a.index(21)

##6. SETS

###Creating Set

In [44]:
```
a = set()
```

In [47]:
```
a.add("Champ")
a
```

Out[47]:
```
{'Champ'}
```

In [61]:
```
c=[1,2,3]
c=tuple(c)
c
```

Out[61]:
```
(1, 2, 3)
```

In [49]:
```
b = ["Champ",21,99.5]
set(b)
```

Out[49]:
```
{21, 99.5, 'Champ'}
```

##7. BOOLEAN

###Creating Boolean

In [ ]:
```
a = True
```

In [62]:
```
a = 10
b = 5
c=15
print(a<b)
print(a>b)
print(a==b)
print(a!=b)
print(a<=b)
print(a>=b)
print(a<b and  c>a)
print(a<b or  c>a)
```

```
False
True
False
True
False
True
False
True
```

##8.Python Statements

###If

In [4]:
```
a = False
if a:
 print("Positive")
else:
  print("negative")
```

```
negative
```

In [5]:
```
a=10
b=6
c=6

if a<b:
  print("a less than b")
elif c==b:
  print("a is equal to b")
  print("a is not less than b")
  if c<a:
    print("c is less than b")
else:
  print("a is not less than b")
```

```
a is equal to b
a is not less than b
c is less than b
```

###For

In [7]:
```
a = [1,2,3,4,5,6,7,8,9,10]



for i in a:
  print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [10]:
```
a = [i for i in 'Champ']
a
```

Out[10]:
```
['C', 'h', 'a', 'm', 'p']
```

###Odd & Even Number

In [13]:
```
a = [1,2,3,4,5,6,7,8,9,10]
for i in a:
  if i % 2 == 0:
    print(i,"Even Number")
  else:
    print(i,"Odd Number")
```

```
1 Odd Number
2 Even Number
3 Odd Number
4 Even Number
5 Odd Number
6 Even Number
7 Odd Number
8 Even Number
9 Odd Number
10 Even Number
```

###String - For

In [14]:
```
for a in "Hello all":
  print(a)
```

```
H
e
l
l
o

a
l
l
```

###Dictionary - For

In [17]:
```
a = {"Name":"Elon","Age":50,"Company":["SpaceX","Tesla"]}
for k,v in a.items():
  print(k)
  print(v)
```

```
Name
Elon
Age
50
Company
['SpaceX', 'Tesla']
```

###While

In [19]:
```
a=0
while a <=11:
  print(a)
  a+=1 #a=a+1
```

```
0
1
2
3
4
5
6
7
8
9
10
11
```

While - Break & Continue

In [20]:
```
a=0
while a < 10:
  print(a)
  a+=1
  if a==5:
    print("a is equal to 5")
    break
  else:
    print("Continueeee")
    continue
```

```
0
Continueeee
1
Continueeee
2
Continueeee
3
Continueeee
4
a is equal to 5
```

###Range

In [25]:
```
for b in range (1,6):
  print(b)
```

```
1
2
3
4
5
```

In [31]:
```
for a in range (1,10):

  print(a)
```

```
1
2
3
4
5
6
7
8
9
```

In [32]:
```
for a in range (10,0,-3):
  print(a)
```

```
10
7
4
1
```

In [33]:
```
a = [i**2 for i in range(0,5)]
a
```

Out[33]: [0, 1, 4, 9, 16]

In [34]:
```
a = [i for i in range(10) if i % 2 == 0]
a
```

Out[34]: [0, 2, 4, 6, 8]

Enumerate - To track no. of iteration in the loop, without working with variable increament

In [35]:
```
a=0
for i,a in enumerate("Hello Champ"):
  print(i,a)
```

```
0 H
1 e
2 l
3 l
4 o
5
6 C
7 h
8 a
9 m
10 p
```

Zip - Creating a tuples by zipping 2 list

In [48]:
```
a = ["Name","Age","Country"]
b = ["Champ",27,"India"]
x =list(zip(a,b))
type(x[0])

```

Out[48]:
```
tuple
```

##9. Function

###Initializing & Calling basic function

In [53]:
```
def welcome():
  print("Hello guyz Welcome to Python Bootcamp !!!")
welcome()
```

```
Hello guyz Welcome to Python Bootcamp !!!
```

In [54]:
```
welcome()
```

```
Hello guyz Welcome to Python Bootcamp !!!
```

###Initializing & Calling basic function - With Argument

In [57]:
```
def welcome(a):
  print("Hello {0},Welcome to Python Bootcamp !!!".format(a))
welcome("Champ")
```

```
Hello Champ,Welcome to Python Bootcamp !!!
```

In [56]:
```

```

```
Hello Champ,Welcome to Python Bootcamp !!!
```

###Print & Return

In [65]:
```
def add(a,b):
  add = a+b
  print("Sum of {0} and {1} is {2}".format(a,b,add))
add(a,b)

```

```
Sum of ['Name', 'Age', 'Country'] and ['Champ', 27, 'India'] is ['Name', 'Age',
'Country', 'Champ', 27, 'India']
```

In [73]:
```
def addR(a,b):
  add = a+b
  return add

ans = addR(45,5)
final = ans*500
final
```

Out[73]: 25000

In [69]:
```

```

Out[69]: 25000

Practice Project - Extracting Prime no.

In [ ]:
```
def check(numbers):
    primeNumber = []
    for number in numbers:
      if number>1:
        for i in range(2, int(number/2)+1):
          if (number % i) == 0:#If number is divisible by any number between
              print(number, "is not a prime number")
              break
        else:
          print(number, "is a prime number")
          primeNumber.append(number)
      else:
        pass
    return primeNumber
```

In [ ]:
```
check([2,3,4,5,6,7,8,9])
```

10 Map Function - Map a function to an iterable object

In [ ]:
```
def cubeFn(num):
    return num**3
```

In [ ]:
```
a = [1,2,3,4,5]
list(map(cubeFn,a))
```

##11. Filter Functions - Yields items of iterable in which function is true

In [ ]:
```
def evenFn(num):
    return num % 2 == 0
```

In [ ]:
```
a = [0,1,2,3,4,5,6,7,8,9,10]
print(list(map(evenFn,a)))
print(list(filter(evenFn,a)))
```

12. Lambda Fn - To create anonymous functions, without using def

In [ ]:
```
#Normal Fn
def cubeFn(num):
    return num**3
cubeFn(6)
```

In [ ]:
```
# LAMBDA
cubefn = lambda num: num**3
cubefn(5)
```

##13. *args and *kwargs

In [ ]:
```
#Normal Fn. with Arguements
def addR(a,b,c,d):
  add = a+b+c
  return add
addR(10,20,30,40)
```

In [ ]:
```
#Purpose of args
def addR(*argg):
  return sum(argg)
addR(10,20,30,40,52,75,8,5,26,15)
```

**Kwargs - dictionary of key/value pairs

In [ ]:
```
def func(**kwargs):
    if 'name' in kwargs:
        print("My Name is {0}".format(kwargs['name']))
    if 'age' in kwargs:
        print("My Age is {0}".format(kwargs['age']))
    else:
        print("No Key Found")

func(name='champ',age=24,marks=99)
```

14. Object Oriented Programming

##Class

User defined objects created x using Class- Initializing Class

In [ ]:
```
class Student:
  print("Hello All")
x = Student() #Object Instantiation
print(type(x))
```

In [ ]:
```
class Student:
  def func():
    print("Hello all welcome to the session")
Student.func()
```

In [ ]:
```
class Student:
  def func(self):
    print("Hello all welcome to the session")
x = Student()
x.func()
```

###Class Object attribute - same for any instance of the class

In [ ]:
```
class Student:
  year = '2021'
  def func():
    print("Hello all welcome to the session")
x = Student()
x.year
```

In [ ]:
```
class Student:
  year = '2021'
  def func(self):
    print("Hello all welcome to the session", self.year)
x = Student()
x.func()
```

Attribute of an Object (characteristic of an object) Attribute won't take any argument

In [ ]:
```
class Student:
    def __init__(self,name): #__init__ (Method) to initialize attribute of a
        self.name = name #self.name = attribute initialized
 #(Creating Instance of Student class{Object of certain class})
elon = Student(name='Elon Musk !') #name: argument

print(elon.name) #accessing class attribute through object
```

In [1]:
```
class Student:
    def __init__(self,name):
        self.name = name
elon = Student(name='Elon Musk !')
champ = Student(name='I am Champ !')

print(elon.name)
print(champ.name)
```

```
Elon Musk !
I am Champ !
```

In [2]:
```
class Student:
    def __init__(self,name,age):
        self.name = name
        self.age = age
elon = Student(name='Elon Musk !',age=40)

print(elon.name)
print(elon.age)
```

```
Elon Musk !
40
```

In [ ]:
```
class Student:
    Total = 500

    def __init__(self, marks):
        self.marks=marks
        print("Initialized...")

    def findLoss(self):
        return self.Total - self.marks

    def findPercentage(self):
        return self.marks/self.Total*100

a = Student(marks=450)

print('Total Marks: ',a.Total)
print('Lossed Marks: ',a.findLoss())
print('Percentage is: ',a.findPercentage())
```

##Special Methods

In [ ]:
```
class Student:
    Total = 500

    def __init__(self, name,marks,gender):
        self.name=name
        self.marks=marks
        self.gender=gender
        print("Initialized...")

    def __len__(self):
        return self.marks

    def __str__(self):
        return "Name: %s | Marks: %s | Gender: %s" %(self.name,self.marks,se

    def __del__(self):
        print("Student Database is Deleted")

a = Student('champ',450,'male')

print(a)
print('Marks: ',len(a))
del a
```

##Inheritance - Help to reduce complexity of the program

Base Class & Derived Class

In [ ]:
```
class Elon:
    def __init__(self):
        print("Profile created")

    def name(self):
        print("Elon Musk")

    def age(self):
        print("40")


class SpaceX(Elon):
    def __init__(self):
        Elon.__init__(self)
        print("Company Profile created")

    def name(self):
        print("SpaceX")

    def type(self):
        print("Private Space travel")
```

In [ ]:
```
a = SpaceX()
```

In [ ]:
```
a.name() #Derived class modified behavior of base class
```

In [ ]:
```
a.age()
```

##Polymorphism

Different object classes can share the same method name, and those methods can be called from the same place even though a different objects passed in

In [ ]:
```
class Elon:
    def __init__(self,name):
        self.name = name
    def type(self):
        return "Entrepreneur"
class Sundar:
    def __init__(self,name):
        self.name = name
    def type(self):
        return "CEO"

person1 = Elon('Elon Musk') #name: argument
person2 = Sundar('Sundar Pichai')
```

In [ ]:
```
#returning unique result of object which have same method(type)
print(person1.type())
print(person2.type())
```

In [ ]:
```
for i in [person1,person2]:
  print(i.name)
  print(i.type())
  print("***************")
```

15 Python Decorators - functions which modify the functionality of another function/Class

In [ ]:
```
'''@decName
def func():
    print("statement")

EQUAL TO

def func():
    print("statement")

func = decName(func)'''
```

###FUNCTION AS OBJECT

In [ ]:
```
def welcome(a):
  print("Hello {0},Welcome to Python Bootcamp !!!".format(a))
```

In [ ]:
```
a = welcome #No paranthesis, if its paranthesis it will call fn. - FUNCTION
a("champ")
```

###FUNCTION IN VARIABLE

In [ ]:
```
def lowerCase(text):
  return text.lower()

def upperCase(text):
  return text.upper()

def a(welcome):
  message = welcome("Hello all, Welcome to Python Bootcamp") #FUNCTION IN VA
  print(message)

a(lowerCase)
a(upperCase)
```

In [ ]:
```
def lowerCase(text):
  return text.lower()

def a(welcome):
  message = welcome("Hello all, Welcome to Python Bootcamp") #FUNCTION IN VA
  print(message)

a(lowerCase)
```

###RETURNING FUNCTION FROM ANOTHER FUNCTION

In [ ]:
```
def addMain(a):#child
  def addSub(b): #parent  = a+b
    print(a,b)
    return a+b #apple
  return addSub #return of parent is return of child now - apple {child = re

addition = addMain(100)
print(addition(75))
```

##DECORATORS in Action

In [ ]:
```
def decoratorFunc (welcome):#2. Decorator
  def a(): #3. can access the outer local functions like in this case "welco
    print("Start")
    welcome() #4.calling actual fn.
    print("End")
  return a

def subFunc(): #veg with cheese
  print("Sub fn")

subFunc = decoratorFunc(subFunc) #1. subFunc inside the decorator to control

subFunc()
```

16 Python Generators - to generate as we go along, instead of holding everything in memory & generator functions will automatically suspend and resume their execution and state around the last point of value generation | n. This feature is known as state suspension

In [ ]:
```
def square(n):
  for i in range (n):
    yield i**2
```

In [ ]:
```
for n in square(10):
  print(n)
```

###Builtin Function - next

In [ ]:
```
def square():
  for i in range (n):
    yield i**2
```

In [ ]:
```
a = square()
```

In [ ]:
```
print(next(a)) #After yielding all the values next() caused a StopIteration
```

###Builtin Function - iter

In [ ]:
```
a = "champ"
for i in a:
  print(i)
```

In [ ]:
```
next(a)
```

In [ ]:
```
iterOper = iter(a)
```

In [ ]:
```
next(iterOper)
```

17 Python Modules & Libraries

###Accessing one Python program from another program

In [ ]:
```
import main #importing python file
main.welcome() #parathesis (Func)
x = main.a #no paranthesis (variable)
y = main.b
print(x+y)
```

Accessing python program from another Python program present in another Folder

In [ ]:
```
from sample_data.demo import main
main.welcome()
x = main.a
y = main.b
print(x+y)
```

##Try Except Finally

###Base Exception Errors

In [ ]:

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StandardError
      |    +-- BufferError
      |    +-- ArithmeticError
      |    |    +-- FloatingPointError
      |    |    +-- OverflowError
      |    |    +-- ZeroDivisionError
      |    +-- AssertionError
      |    +-- AttributeError
      |    +-- EnvironmentError
      |    |    +-- IOError
      |    |    +-- OSError
      |    |         +-- WindowsError (Windows)
      |    |         +-- VMSError (VMS)
      |    +-- EOFError
      |    +-- ImportError
      |    +-- LookupError
      |    |    +-- IndexError
      |    |    +-- KeyError
      |    +-- MemoryError
      |    +-- NameError
      |    |    +-- UnboundLocalError
      |    +-- ReferenceError
      |    +-- RuntimeError
      |    |    +-- NotImplementedError
      |    +-- SyntaxError
      |    |    +-- IndentationError
      |    |         +-- TabError
      |    +-- SystemError
      |    +-- TypeError
      |    +-- ValueError
      |         +-- UnicodeError
      |              +-- UnicodeDecodeError
      |              +-- UnicodeEncodeError
      |              +-- UnicodeTranslateError
      +-- Warning
           +-- DeprecationWarning
           +-- PendingDeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
       +-- ImportWarning
       +-- UnicodeWarning
       +-- BytesWarning
```

In [ ]:
```
for i in range(3, -3, -1):
    try:
        print(1.0 / i)
    except ZeroDivisionError:
        print("You're trying to divide by zero. U IDIOT")
```

In [ ]:
```
for i in range(3, -3, -1):
    try:
        print(1.0 / i)
    except ZeroDivisionError as er:
        print('Zero Division Error: ', str(er.args[0]))
```

In [ ]:
```
def check():
    while True:
        try:
            a = int(input("Please Value of a in numbers: "))
            b = int(input("Please enter value of b in numbers: "))
        except:
            print("You did not entered numbers")
            continue
        else:
            print("Yes its Numbers")
            break
        finally:
            print("Execution Successfull")
            print(a+b)

check()
```

###Importing Library

In [ ]:
```
import math #import libraryName
```

###Installing Library

In [ ]:
```
!pip install imutils #pip install libraryName (! only while executing in col
```

###uninstalling Library

In [ ]:
```
!pip uninstall imutils #pip uninstall libraryName
```

In [ ]:
```
import imutils
```

###Installing Library based on Version

In [ ]:
```
!pip install imutils==0.5.4 #pip install libraryName==versionNumber
```

###Checking Installed Library version

In [ ]:
```
import imutils
imutils.__version__
```

##Playing with Libraries

###Math Library

In [ ]:
```
import math
print(dir(math))
print(math.pi)
```

In [ ]:
```
import math as mt
print(dir(mt))
print(mt.pi)
```

###Reduce

In [ ]:
```
from functools import reduce
lst =[47,11,42,13]
reduce(lambda x,y: x+y,lst)
```

###IPython

In [ ]:
```
from IPython.display import Image
Image('https://venturebeat.com/wp-content/uploads/2018/09/ironman.jpg')
```

###Filter

In [ ]:
```
def check(num):
    if num%2 ==0:
        return True
lst =range(20)
list(filter(check,lst))
```

###Counter

In [ ]:
```
from collections import Counter
print(Counter([5,6,8,4,7,5,9,6,2,1,3,5,8,4,5,8,7,2,5,8,6,2,5,4,1,2,6,8,7,4,5
print(Counter('sdasdasdadasdasdasdassdasdassdasdadasdaasdasdasd')) #String
```

###Regular Expression

In [ ]:
```
import re
patt = r'\d{2}-\d{2}-\d{4}'
message = "His birthday is 16-11-2016"
re.findall(patt,message)
```

###Working with Files

In [ ]:
```
f = open("welcome.txt", "r")
print(f.read())
f.close()
```

In [ ]:
```
f = open("welcome.txt", "r")
print(f.read(5))
f.close()
```

In [ ]:
```
f = open("welcome.txt", "r")
print(f.readline())
f.close()
```

In [ ]:
```
f = open("welcome.txt", "r")
for x in f:
  print(x)
```

###File Handling using Pandas

In [ ]:
```
!pip install pandas
```

In [ ]:
```
import pandas as pd
data = pd.read_csv('data.csv')
print(data.to_string())
```

In [ ]:
```
import pandas as pd
data = pd.read_csv('data.csv')
print(data.shape)
print(data.describe())
print(data.head(5))
```

###Working with Directories

In [ ]:
```
import os
print(os.getcwd()) #current working directory
print(os.listdir())
os.mkdir("Junk")
```

In [ ]:
```
import shutil
shutil.move('welcome.txt','/content/Junk')
```

In [ ]:
```
import send2trash
os.listdir()
send2trash.send2trash('Junk/welcome.txt')
```

In [ ]:
```
os.listdir()
```

###Time

In [ ]:
```
import time
print(time.time())#time since epoch
```

In [ ]:
```
import time
print(time.gmtime())
```

In [ ]:
```
print (time.ctime())
```

In [ ]:
```
startTime = time.time()
i=0
while(i<5):
  i+=1
  print(".")
  time.sleep(1) #delay 1 second
stopTime = time.time()
diff = stopTime - startTime
diff
```

###Date & Time

In [ ]:
```
import datetime
x = datetime.datetime.now()
print(x)
```

In [ ]:
```
import datetime
x = datetime.datetime(2021, 12, 16, 12, 30, 51) #(year, month, date, hour, m
print(x)
```

In [ ]:

```
'''%a   Weekday, short version  Wed
%A  Weekday, full version   Wednesday
%w  Weekday as a number 0-6, 0 is Sunday    3
%d  Day of month 01-31  31
%b  Month name, short version   Dec
%B  Month name, full version    December
%m  Month as a number 01-12 12
%y  Year, short version, without century    18
%Y  Year, full version  2018
%H  Hour 00-23  17
%I  Hour 00-12  05
%p  AM/PM   PM
%M  Minute 00-59    41
%S  Second 00-59    08
%f  Microsecond 000000-999999   548513
%z  UTC offset  +0100
%Z  Timezone    CST
%j  Day number of year 001-366  365
%U  Week number of year, Sunday as the first day of week, 00-53 52
%W  Week number of year, Monday as the first day of week, 00-53 52
%c  Local version of date and time  Mon Dec 31 17:41:00 2018
%C  Century 20
%x  Local version of date   12/31/18
%X  Local version of time   17:41:00
%%  A % character   %
%G  ISO 8601 year   2018
%u  ISO 8601 weekday (1-7)  1
%V  ISO 8601 weeknumber (01-53) 01'''
```

In [ ]:

```
import datetime
a = datetime.datetime(2021, 12, 16)
print(a.strftime("%a"))
```