In [8]:
```
print('Hello harsh')
```

```
Hello harsh
```

In [ ]:
```

```

In [9]:
```
2+6
```

Out[9]: 8

In [14]:
```
5/2 # div:  point mai bhi div karega [quotient]
```

Out[14]: 2.5

In [15]:
```
5%2 # modulas : [remainder]
```

Out[15]: 1

In [16]:
```
5//2 # floor division : point mai div nhi hoga
```

Out[16]: 2

In [17]:
```
5*2
```

Out[17]: 10

In [21]:
```
2^4
```

Out[21]: 6

In [20]:
```
2**4 # exponent operator
```

Out[20]: 16

In [ ]:
```

```

Simple interest

In [25]:
```
p = float(input('p = '))
r = float(input('r = '))
t = float(input('t = '))
si = p*r*t
print('Simple Interest : ',si)
```

```
p = 2000
r = 0.055
t = 4
Simple Interest :  440.0
```

Square of a number

In [26]:
```
no = int(input('Enter the no. : '))
sq = no*no
print(f'Square of {no} is ',sq)
```

```
Enter the no. : 4
Square of 4 is  16
```

Area of circle

In [30]:
```
r = float(input('Enter the radius value : '))
area = 3.14*r*r
print('Area of circle is ',area)
```

```
Enter the radius value : 7
Area of circle is  153.86
```

In [ ]:

Types of Operators

```
Arithmatic Operators
Relational Operators
Assignment Operators
Logical Operators
Membership Operators
Identity Operators
```

In [ ]:

In [35]:
```
d1 = 'PYTHON IS AWESOME'
d2 = 'flask'
x = d1.replace('AWESOME','too awesome')
print(x)
```

```
PYTHON IS too awesome
```

In [37]:
```
print(d1.lower())
print(d2.upper())
```

```
python is awesome
FLASK
```

In [38]:
```
d1 = 'PYTHON IS AWESOME'
print(d1.split())
```

```
['PYTHON', 'IS', 'AWESOME']
```

In [65]:
```
d2 = 'flask is'
print(d2.capitalize())
```

```
Flask is
```

In [ ]:

In [40]:
```
d2 = 'flask'
print(d2.center(15))
```

```
      flask
```

In [ ]:
```

```

In [41]:
```
d1 = 'PYTHON IS AWESOME'
print(d1.endswith('me'))
```

```
False
```

In [42]:
```
print(d1.endswith('ME'))
```

```
True
```

In [44]:
```
print(d1.endswith('xx'))
```

```
False
```

In [ ]:
```

```

In [46]:
```
d1 = 'PYTHON IS AWESOME THON'
print(d1.find('ON'))
```

```
4
```

In [48]:
```
print(d1.find('P'))
```

```
0
```

In [49]:
```
print(d1.find('N'))
```

```
5
```

In [ ]:
```

```

In [52]:
```
d1 = 'PYTHON IS AWESOME THON'
print(d1.index('N'))
```

```
5
```

In [ ]:
```

```

In [53]:
```
d1 = 'PYTHON IS AWESOME THON'
print(d1.isalpha())
```

```
False
```

In [54]:
```
d1 = 'PYTHONISAWESOMETHON'
print(d1.isalpha())
```

```
True
```

In [55]:
```
d1 = 'PYTHON00ISAWESOMETHON'
print(d1.isalpha())
```

```
False
```

In [ ]:

In [56]:
```
d1 = 'PYTHON00ISAWESOMETHON'
print(d1.isdigit())
```

```
False
```

In [57]:
```
d1 = '123'
print(d1.isdigit())
```

```
True
```

In [ ]:

In [64]:
```
d1 = '1 23'
print(d1.isspace())
```

```
False
```

In [62]:
```
d1 = ''
print(d1.isspace())
```

```
False
```

In [63]:
```
d1 = ' '
print(d1.isspace())
```

```
True
```

In [ ]:

In [66]:
```
d1 = 'Python'
print(d1.istitle()) # first word capital
```

```
True
```

In [67]:
```
d1 = 'python'
print(d1.istitle())
```

```
False
```

In [ ]:

In [68]:
```
d1 = 'is python is best is'
print(d1.count('is'))
```

```
3
```

In [69]:
```
print(d1.count('t'))
```

```
2
```

In [ ]:

In [3]:
```
st = 'harry is a goog man'
print(st.split())
```

```
['harry', 'is', 'a', 'goog', 'man']
```

In [ ]:
```

```

In [4]:
```
l1 = ['harry', 'is', 'a', 'goog', 'man']
x = '$'.join(l1)
print(x)
```

```
harry$is$a$goog$man
```

In [5]:
```
x = '_'.join(l1)
print(x)
```

```
harry_is_a_goog_man
```

In [ ]:
```

```

Python Collections

```
- List
- Tuple
- Set
- Dictionaries

1:55
```

List

In [ ]:
```
# replace item
```

In [7]:
```
l1 = ['harry', 'is', 'a', 'goog', 'man']
l1[3]='GOOD'
print(l1)
```

```
['harry', 'is', 'a', 'GOOD', 'man']
```

In [8]:
```
# append/add items
```

In [9]:
```
l1 = ['apple','mango','kiwi','peach']
l1.append('kiwi')
print(l1)
```

```
['apple', 'mango', 'kiwi', 'peach', 'kiwi']
```

In [10]:
```
print(l1.count('kiwi'))
```

```
2
```

In [11]:
```
# remvove item
```

In [15]:
```
l1 = ['apple','mango','kiwi','peach']
l1.remove('kiwi')
print(l1)
```

```
['apple', 'mango', 'peach']
```

In [16]:
```
l1.pop()
print(l1)
```

```
['apple', 'mango']
```

In [17]:
```
print(l1[1]) # ordered format
```

```
mango
```

In [ ]:
```
# del a list
```

In [18]:
```
l1 = ['apple','mango','kiwi','peach']
del l1
```

In [19]:
```
print(l1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call l
ast)
Input In [19], in <module>
----> 1 print(l1)

NameError: name 'l1' is not defined
```

In [20]:
```
# empty/clear the list
l1 = ['apple','mango','kiwi','peach']
l1.clear()
print(l1)
```

```
[]
```

In [ ]:
```

```

In [3]:
```
l1 = ['apple','mango','kiwi','peach']
print('#'.join(l1))
```

```
apple#mango#kiwi#peach
```

tuple

In [5]:
```
t1 = ('apple','mango','kiwi','peach')
t1
```

Out[5]:
```
('apple', 'mango', 'kiwi', 'peach')
```

In [6]:
```
t1 = ('apple','mango','kiwi','peach')
t2 = ('a','b','c','d')
t1+t2
```

Out[6]:
```
('apple', 'mango', 'kiwi', 'peach', 'a', 'b', 'c', 'd')
```

In [ ]:
```

```

In [9]:
```
t1 = ('apple','mango','kiwi','peach')
t1.count('kiwi')
```

Out[9]:
```
1
```

In [10]:
```
t2 = ('a','b','c','a','d','a')
t2.count('a')
```

Out[10]:
```
3
```

In [ ]:
```

```

In [11]:
```
t2 = ('a','b','c','a','d','a')
t2.index('a')
```

Out[11]:
```
0
```

In [12]:
```
t1 = ('apple','mango','kiwi','peach')
t1.index('kiwi')
```

Out[12]:
```
2
```

In [ ]:
```

```

Set

In [14]:
```
s1 = {'a','b','c','d','a','b','c','d'}
s1
```

Out[14]:
```
{'a', 'b', 'c', 'd'}
```

In [15]:
```
s1 = {'apple','mango','kiwi','peach'}
s1
```

Out[15]:
```
{'apple', 'kiwi', 'mango', 'peach'}
```

In [ ]:
```

```

In [16]:
```
s1 = {'apple','mango','kiwi','peach'}
s1.add('mango')
s1
```

Out[16]:
```
{'apple', 'kiwi', 'mango', 'peach'}
```

In [17]:
```
s1 = {'apple','mango','kiwi','peach'}
s1.add('berry')
s1
```

Out[17]:
```
{'apple', 'berry', 'kiwi', 'mango', 'peach'}
```

In [ ]:

In [18]:
```
s1 = {'apple','mango','kiwi','peach'}
s1.update(['a','b','c'])
s1
```

Out[18]:
```
{'a', 'apple', 'b', 'c', 'kiwi', 'mango', 'peach'}
```

In [ ]:

Dict

In [20]:
```
d1 = {
    'a':100,
    'b':200,
    'c':'zero'
}
d1
```

Out[20]:
```
{'a': 100, 'b': 200, 'c': 'zero'}
```

In [ ]:

In [23]:
```
d1['a']='updated'
d1
```

Out[23]:
```
{'a': 'updated', 'b': 200, 'c': 'zero'}
```

In [24]:
```
d1['c']='updated'
d1
```

Out[24]:
```
{'a': 'updated', 'b': 200, 'c': 'updated'}
```

In [ ]:

In [26]:
```
x = d1['c']
x
```

Out[26]:
```
'updated'
```

In [ ]:

In [27]:
```
d1['new']='new item added'
d1
```

Out[27]:
```
{'a': 'updated', 'b': 200, 'c': 'updated', 'new': 'new item added'}
```

In [ ]:

In [29]:
```
d1 = {'a': 'updated', 'b': 200, 'c': 'updated', 'new': 'new item added'}
d1.pop('new')
d1
```

Out[29]:
```
{'a': 'updated', 'b': 200, 'c': 'updated'}
```

In [31]:
```
#or
```

In [32]:
```
d1 = {'a': 'updated', 'b': 200, 'c': 'updated', 'new': 'new item added'}
del d1['new']
d1
```

Out[32]:
```
{'a': 'updated', 'b': 200, 'c': 'updated'}
```

In [ ]:
```

```

In [33]:
```
d1 = {'a': 'updated', 'b': 200, 'c': 'updated', 'new': 'new item added'}
del d1
d1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call l
ast)
Input In [33], in <module>
      1 d1 = {'a': 'updated', 'b': 200, 'c': 'updated', 'new': 'new ite
m added'}
      2 del d1
----> 3 d1

NameError: name 'd1' is not defined
```

In [ ]:
```

```

In [35]:
```
d1 = {'a': 'updated', 'b': 200, 'c': 'updated', 'new': 'new item added'}
d1.clear()
d1
```

Out[35]:
```
{}
```

In [ ]:
```

```

In [36]:
```
marks_d1 = {'ram':100,'raju':80}
marks_d2 = marks_d1.copy()
```

In [37]:
```
marks_d1
```

Out[37]:
```
{'ram': 100, 'raju': 80}
```

In [38]:
```
marks_d2
```

Out[38]:
```
{'ram': 100, 'raju': 80}
```

In [ ]:
```

```

Python Conditions

```
if condition
```

In [39]:
```
a = True
if a:
    print('If true then execute')
```

```
If true then execute
```

In [ ]:
```

```

```
if-else condition
```

In [46]:
```
a = False
if a:
    print('If true then execute')
else:
    #pass
    print('if false then execute')
```

```
if false then execute
```

In [ ]:
```

```

```
simple Elif
```

In [49]:
```
a = 10
if a>10:
    print('greater than 10')
elif a<10:
    print('less than 10')
else:
    print('value is 10')
```

```
value is 10
```

In [ ]:
```

```

```
Nested If
```

In [51]:
```
a = True
b = True
if a:
    if b==False:
        print('A is True B is False')
    else:
        print('A is True B is True')
```

```
A is True B is True
```

In [ ]:
```

```

In [57]:
```
a = int(input())
b = int(input())
c = int(input())

if a>b and a>c:
    print(f'{a} is greatest among {a,b,c}')
elif b>a and b>c:
    print(f'{b} is greatest among {a,b,c}')
elif a==b and b==c and c==a:
    print(f'{a,b,c} are same')
else:
    print(f'{c} is greatest among {a,b,c}')
```

```
1
1
1
(1, 1, 1) are same
```

In [ ]:

Python Loops

While Loop

In [59]:
```
i = 1
while i<=10:
    print(i, end=' ')
    i+=1
```

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

In [ ]:

Iterating over a list

In [69]:
```
l1 = ['apple','mango','kiwi','peach','melon','berry']
for i in l1:
    print(i)
```

```
apple
mango
kiwi
peach
melon
berry
```

In [71]:
```
l1 = ['apple','mango','kiwi','peach','melon','berry']
for i in range(4):
    print(l1[i])
```

```
apple
mango
kiwi
peach
```

In [72]:
```
l1 = ['apple','mango','kiwi','peach','melon','berry']
for i in range(0,4):
    print(l1[i])
```

```
apple
mango
kiwi
peach
```

In [73]:
```
l1 = ['apple','mango','kiwi','peach','melon','berry']
for i in range(2,4):
    print(l1[i])
```

```
kiwi
peach
```

In [ ]:

Prog. for printing table for any no

In [74]:
```
table_no = int(input())
for i in range(1,11):
    print(f'{table_no} X {i} = {i*table_no}')
```

```
3
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
3 X 10 = 30
```

In [ ]:

In [76]:
```
table_no = int(input('Enter the number till u wanna generate tables : '))

for i in range(1,table_no+1):
    for j in range(1,11):
        print(f'{i} X {j} = {i*j}')
    print()
```

```
Enter the number till u wanna generate tables : 3
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
1 X 10 = 10

2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20

3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
3 X 10 = 30
```

In [ ]:

In [78]:
```
a = int(input())
b = int(input())
print(f'max of {a} & {b} is {max(a,b)}')
```

```
111
1
max of 111 & 1 is 111
```

In [ ]:

In [80]:
```
# functn to generate table

def table(table_no):
    for i in range(1,11):
        print(f'{table_no} X {i} = {i*table_no}')

n = int(input())
table(n)
```

```
3
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
3 X 10 = 30
```

In [ ]:

In [86]:
```
# fuctn to find percentage

sci = int(input('Enter sci marks : '))
math = int(input('Enter math marks : '))
hindi = int(input('Enter hindi marks : '))
eng = int(input('Enter eng marks : '))
guj = int(input('Enter guj marks : '))

total = sum((sci,math,hindi,eng,guj))

def per(total):
    return f'percentage is {total/500*100}'

# calling the fucntn
per(total)
```

```
Enter sci marks : 23
Enter math marks : 34
Enter hindi marks : 56
Enter eng marks : 78
Enter guj marks : 98
```

Out[86]: 'percentage is 57.8'

In [ ]:

Lambda Fuctn

In [87]:
```
 sum((344,616,449,394,499))
```

Out[87]: 2302

In [ ]:

lambda arguments : expression

- One line function / Anonymous function

In [88]:
```
x = lambda a : a + 10
print(x(5))
```

```
15
```

In [89]:
```
x = lambda a,b,c : a+b+b
print(x(1,2,3))
```

```
5
```

In [3]:
```
def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)

print(mydoubler(11))
```

```
22
```

In [ ]:
```

```

Python Modules

Module is a file consisting of fucntions which can be imported and used in our programs

2 types : Built in, user defined

In [4]:
```
# pre-installed fuctn in py

import time
local_time = time.asctime(time.localtime(time.time()))
print(local_time)
```

```
Thu Jul 14 00:16:53 2022
```

In [7]:
```
# pre-installed fucntn in py

import calendar
cal = calendar.month(2022, 7)
print(cal)
```

```
     July 2022
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
```

In [10]:
```
# importing the calendar module

import calendar
cal = calendar.month(2000, 4)
# importing the month futn from calendar module

print(cal)
```

```
     April 2000
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
```

In [ ]:

In [ ]:

In [12]:
```
from myfunc import *

x = sqroot(2)
x
```

Out[12]: 4

In [13]:
```
y = cube(2)
y
```

Out[13]: 8

or

In [15]:
```
import myfunc as f
cube = f.cube(3)
print(cube)
```

```
27
```

In [17]:
```
import myfunc
cube = myfunc.cube(4)
print(cube)
```

```
64
```

In [ ]:

File handling

In [18]:
```
open('myfunc.py','r')
```

Out[18]: <_io.TextIOWrapper name='myfunc.py' mode='r' encoding='UTF-8'>

In [ ]:

opening a file by passing the access mode

In [22]:
```
f = open('msg.txt','r')
# open is a functn that take the file and the mode type to open the file
myfile = f.read()
# now we are reading the file by storing thefile into the variable
print(myfile)
# its important to close the file once the work has been completed
f.close()
```

```
hey there everyone I am here with you lorem
```

In [ ]:

In [23]:
```
# opening the file name msg2 into the read mode

f = open('msg2.txt')
file = f.read()
print(file)
f.close()
```

```
this  is a second file for the same project
```

In [ ]:

In [ ]:

In [28]:
```
# opening a file name.txt in read mode

# firstly writing a file called name.txt
file = open("name.txt", "w")
file.write("Your text goes baby")
file.close()
```

In [29]:
```
# opening name.txt file

f = open('name.txt','r')
print(f.read())
```

```
Your text goes baby
```

In [ ]:

over-writng the file using the same way we created a file

In [36]:
```
# opening the file in read mode
file = open('name.txt','r')
print(file.read())
file.close()
```

```
Your text goes baby
```

In [37]:
```
# we can also aassign the data of the file opened to the variable
f = open('name.txt','r')
x = f.read()
print('Data of the file : ',x)

f.close()
```

```
Data of the file :  Your text goes baby
```

In [ ]:
```

```

In [54]:
```
# writing data into the file by creating a new file 'newboy.txt'

file = open('nameboy.txt','w')
file.write('Writing some new data here!! ')
file.close()
```

In [55]:
```
# reading the data fromthe 'newboy,txt' into read mode
# opening the file 'newboy.txt' into read only mode
f = open('nameboy.txt','r')
print(f.read())
f.close()
```

```
Writing some new data here!!
```

In [ ]:
```

```

In [56]:
```
# modifying the contents of the file 'newboy.txt'

# Open in read mode
f = open('nameboy.txt','w')
f.write('love you santa')
f.close()
```

In [59]:
```
# checking whether the data has been modifoed successfully or not

    # opening the file into the read mode
f = open('nameboy.txt','r')
print(f.read())
f.close()
```

```
love you santa
```

In [ ]:
```

```

In [66]:
```
# reading only a specific number if characters from the file while openi

# file : 'nameboy.txt'

f = open('nameboy.txt','r')
print(f.read(6))
# here hte spaces are counted when you are accessign the
# specific number of char from the file
f.close()
```

```
love y
```

In [ ]:

if we want it read the file line by lin we gonna use readline() funtn for it

In [76]:
```
f = open('newtext.txt','w')
f.write('this is the first line\n')
f.write('this is the secind line\n')
f.write('this is the third line\n')
f.close()
```

In [77]:
```
# verifying if the data is succcessfully written into the file 'newtext.t

f = open('newtext.txt','r')
print(f.read())
```

```
this is the first line
this is the secind line
this is the third line
```

In [78]:
```
# using readline() for reading the file line by line
f = open('newtext.txt','r')

# reading the first line
f.readline()
```

Out[78]: 'this is the first line\n'

In [81]:
```
# reading the second line
f.readline()
```

Out[81]: 'this is the secind line.\n'

In [82]:
```
# reading the second line
f.readline()

f.close()
```

In [ ]:

Different ways to access/read the data from the file

In [84]:
```
f = open('newday.txt','r')
print(f.read())
f.close()
```

```
this is the first line.
this is the secind line.
this is the third line.
```

In [ ]:

In [87]:
```
f = open('newday.txt','r')
print(f.readline())
print(f.readline())
print(f.readline())
f.close()
```

```
this is the first line.

this is the secind line.

this is the third line.
```

In [ ]:
```

```

In [90]:
```
f = open('newday.txt','r')
for i in f:
    print(i)
```

```
this is the first line.

this is the secind line.

this is the third line.
```

In [ ]:
```

```

Different types of modes in file handling are

```
'r': Read : Open the file for reading file. Error of the file does not
exist

'a': Append : Opens the file for appending, Creates the file if it soes
note exists

'w': Write :  Opens the file for writing, Creates the file if it soes
note exists

'x': Create : Creates the specified file, returns an eroor if the file
already exist
```

In [91]:
```
# creating a file using 'x' mode

f = open('newfile.txt','x')
f.close()
```

In [92]:
```
# trying to create the same file again

f = open('newfile.txt','x')
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
Input In [92], in <module>
      1 # trying to create the same file again
----> 3 f = open('newfile.txt','x')

FileExistsError: [Errno 17] File exists: 'newfile.txt'
```

In [ ]:
```

```

In [114]:
```
# opening the file into write mode to add some data

    # here we are using 'w' mode as the file is empty

f = open('newfile.txt','w')
f.write('hey\n')
f.write('there\n')
f.write('my boy\n')
f.close()
```

In [115]:
```
    # checking whether the data is added to file

f = open('newfile.txt','r')
print(f.read())
```

```
hey
there
my boy
```

In [ ]:
```

```

In [116]:
```
    # Appending new data into the newfile.txt

f = open('newfile.txt','a')
f.write('Steven')
f.close()
```

In [117]:
```
    # checking whether the new data has been added to the file
f = open('newfile.txt','r')
print(f.read())
f.close()
```

```
hey
there
my boy
Steven
```

In [118]:
```
    # Appending new data into the newfile.txt

f = open('newfile.txt','a')
f.write('\njerry')
f.write('\nbenny')
f.close()
```

In [119]:
```
    # checking whether the new data has been added to the file
f = open('newfile.txt','r')
print(f.read())
f.close()
```

```
hey
there
my boy
Steven
jerry
benny
```

In [ ]:
```

```

```
Using with to open the file instead of traditional method
- the main advantage of using woth is we dont need to close the file
when using with statement
```

In [121]:
```
# using traditional way to open the file 'newfile.txt'

f = open('newfile.txt','r')
print(f.read())
f.close()
```

```
hey
there
my boy
Steven
jerry
benny
```

In [123]:
```
# using with statement to open the file

with open('newfile.txt','r') as p:
    print(p.read())
```

```
hey
there
my boy
Steven
jerry
benny
```

In [124]:
```
# usign readline() with [with statement]

with open('newfile.txt','r') as q:
    print(q.readline())
    print(q.readline())
```

```
hey

there
```

In [ ]:
```

```

rename the file name

In [129]:
```
# here we will rename the file newfile.txt to newmsg.txt

    # here we have read the data fromt the file successfully
with open('newfile.txt','r') as f:
    data = f.read()

    # now we will create a new file and write this data to it
with open('newmsg.txt','w') as nm:
    nm.write(data)

    # verifying the data has been written successfully to the new file
with open('newmsg.txt','r') as p:
    print(p.read())

    # the data has been addedd successfully

    # now we will del the old file with file name 'newfile.txt'
import os
os.remove('newfile.txt')
```

```
hey
there
my boy
Steven
jerry
benny
```

In [132]:
```
    # verifying the file has been removed or not
with open('newfile.txt') as x:
        print(x.read())
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call l
ast)
Input In [132], in <module>
      1 # verifying the file has been removed or not
----> 2 with open('newfile.txt') as x:
      3     print(x.read())

FileNotFoundError: [Errno 2] No such file or directory: 'newfile.txt'
```

In [133]:
```
    # WE got an error means the file has been del or removed successfully
```

In [ ]:

Using Class & Object

In [134]:
```
class StoreDetail():
    def getinfo(self):
        print(f'name = {self.name}')
        print(f'rollno = {self.rollno}')

# creating an instance of the class StoreDetail name : 'raj'

raj = StoreDetail()
raj.name = 'Raj'
raj.rollno = 234

raj.getinfo()
```

```
name = Raj
rollno = 234
```

In [135]:
```
# creating an instance of the class StoreDetail name : 'shiv'

shiv = StoreDetail()
shiv.name='Shiva'
shiv.rollno='0'

shiv.getinfo()
```

```
name = Shiva
rollno = 0
```

In [ ]:

```
note :

Jab bhi object create hota hai tab kisi class ke help sai
aur jab wo obj wo class ke koi fucntn ko access karna chata hai tab
object ek default argument pass karta hai is lia apna ko class
ke fuctn mai self include karna padta hai
fucntn ke positional argument mai
```

```
Agar self add nhi karoge toh ye error receive hoga
- class functn() takes 0 positional argument nut 1 was given

Is lia class fucnt mai self ko include karna jaroori hai
```

In [ ]:

```
note :
- Agar koi obj apna instance variabke create karta hai toh priority usko
milta hai as compared to the class variable
- And wo object ke instance varibale ke lia ek personal space alocate
hoti hai
```

In [139]:
```
class Details():
    school = 'DPS'
    def getinfo(self):
        print(f'name = {self.name}')
        print(f'roll no  = {self.age}')

# creating an isntance of the class Details
s1 = Details()
s1.name='Shivaji'
s1.age=35

s1.getinfo()
print('School - ',s1.school)

# as we can see here there is no isntance var with name school
# for object s1, so the obj will move to the class variable
```

```
name = Shivaji
roll no  = 35
School -  DPS
```

In [141]:
```
# now we have created an instance variable for the object with
# school as 'conrell'

# creating a new instance of the class Details namem 's2'
s2 = Details()
s2.name = 'roshi'
s2.age = 34

s2.getinfo()

# initializing an isntance variable here.
# so this will get the 1st priority over the class variable
s2.school = 'Proton'
print(s2.school)
```

```
name = roshi
roll no  = 34
Proton
```

In [ ]:

In [152]:
```
# creating a class prog to make add of 2 numbers

class Calculate:
    # initialzing the cinstructor here
    def __init__(self,a,b):
        self.a = a
        self.b = b
    def add(self):
        return f'Add of {self.a}+{self.b} = {self.a+self.b}'
    def mul(self):
        return f'Mul of {self.a}*{self.b} = {self.a*self.b}'
    def sub(self):
        return f'Sub of {self.a}-{self.b} = {self.a-self.b}'


# using the add functn
ram = Calculate(12,12)
ram.add()
```

Out[152]:
```
'Add of 12+12 = 24'
```

In [153]:
```
# using the mul funcnt
ram.mul()
```

Out[153]:
```
'Mul of 12*12 = 144'
```

In [154]:
```
# usng the sub fuctn
ram.sub()
```

Out[154]:
```
'Sub of 12-12 = 0'
```

In [ ]:

In [ ]:

```
constructor is a special fuctn which get's called automatically
when an obj is created
```

In [156]:
```
class SmartCalc:
    def __init__(self,a,b):
        self.a=a
        self.b=b
        print(f'add of {self.a} & {self.b} = {self.a+self.b}')
        print(f'mul of {self.a} & {self.b} = {self.a*self.b}')
        print(f'sub of {self.a} & {self.b} = {self.a-self.b}')
```

In [157]:
```
# as we can see after creating the obj the specia; fucntn
# got called automatically
# __init__ constructor/ SPecial methoid is called automatically after
# it is created

raju = SmartCalc(12,12)
```

```
add of 12 & 12 = 24
mul of 12 & 12 = 144
sub of 12 & 12 = 0
```

In [ ]:

In [168]:
```
# creating a menu base calc using class

class Calculate():
    def __init__(self,a,b):
        self.a = a
        self.b = b

    def add(self):
        print(f'Add = {self.a+self.b}')
    def mul(self):
        print(f'Mul = {self.a*self.b}')
    def sub(self):
        print(f'Sub = {self.a-self.b}')

while True:
    a = int(input('enter 1st no : '))
    b = int(input('enter 2nd no : '))
    cal = Calculate(a,b)

    choice = input("Enter the choice : ")
    if choice == '1':
        cal.add()

    elif choice == '2':
        cal.mul()

    elif choice =='3':
        cal.sub()
    else:
        break
```

```
enter 1st no : 12
enter 2nd no : 24
Enter the choice : 1
Add = 36
enter 1st no : 12
enter 2nd no : 24
Enter the choice : 3
Sub = -12
enter 1st no : 10
enter 2nd no : 10
Enter the choice : 2
Mul = 100
enter 1st no : 45
enter 2nd no : 45
Enter the choice : 45
```

In [ ]:
```
```

File handling task

In [ ]:
```
# Creating a file using 'x' mode

f = open('aboutme.txt','x')
f.close()
```

In [7]:
```
# witing a data to the file using 'w' mode
f = open('aboutme.txt','w')
f.write('\nmy name is charles')
f.write('\nI am son of god')
f.close()
```

In [8]:
```
f = open('aboutme.txt','r')
print(f.read())
```

```
my name is charles
I am son of god
```

In [ ]:
```
```

In [10]:
```
# appending new into the file using 'a' mode
f= open('aboutme.txt','a')
f.write('\nI will not lose')
f.write('\nI will defeat every challenge')
f.close()
```

In [14]:
```
# reading the data from file line by line
f = open('aboutme.txt','r')
print(f.readline())
print(f.readline())
print(f.readline())
print(f.readline())
f.close()
```

```
my name is charles

I am son of god

I will not lose
```

In [15]:
```
# reading all the data from the file at once
f = open('aboutme.txt','r')
f.read()
```

Out[15]:
```
'\nmy name is charles\nI am son of god\nI will not lose\nI will defeat
every challenge'
```

In [18]:
```
f = open('aboutme.txt','r')
print(f.read())
f.close()
```

```
my name is charles
I am son of god
I will not lose
I will defeat every challenge
```

In [ ]:
```
```

class & objects

Inheritance => concept of code resuability

In [27]:
```
class PersonalInfo():
    def printdetails(self):
        print(f'Employee Id is {self.empId}')
        print(f'Employee Name is {self.empName}')
        print(f'Employee Age is {self.empAge}')

class Salary(PersonalInfo):
    def calcSalary(self):
        onedaysal = self.msal/30
        working_days = 30-self.leave
        totalsalary = working_days*onedaysal
        print('your salary is ',totalsalary)


# creating an instance of the salary class
harsh = Salary()
harsh.msal = 50000
harsh.leave = 3

# calling the calcSalary fuctn of the Salary class
harsh.calcSalary()

# Accesssing the printdetails fuctn of the parent class
harsh.empId=121
harsh.empName='harsh carter'
harsh.empAge = 22

harsh.printdetails()
```

```
your salary is  45000.0
Employee Id is 121
Employee Name is harsh carter
Employee Age is 22
```

In [ ]:

ex2 : parent and child class

In [5]:
```
class Parent():
    def name(self,name):
        self.name = name
    def age(self,age):
        self.age=age
    def gender(self,gen):
        self.gender = gen

class Child(Parent):
    def show_details(self):
        print('My name is ',self.name)
        print('My age is ',self.age)
        print('My gender is ',self.gender)

# creating an instance of the child class
adam = Child()

# passing all the paramters of the aprent class

adam.name = 'Adam'
adam.age = 32
adam.gender = 'Male'

# accessing the methods of the  child class show_details

adam.show_details()
```

```
My name is  Adam
My age is  32
My gender is  Male
```

In [ ]:
```

```

```
Inheritance example
```

In [7]:
```
# creating a parent class

class Person():
    def __init__(self,fname,lname):
        self.fname=fname
        self.lname=lname

    def show(self):
        print('first name is ',self.fname)
        print('last name is ',self.lname)


# creating a child class
class Student(Person):
    def msg(self):
        print(f'{self.fname} {self.lname} welcome to the class of 2022')

# creating an instance of the child class

s1 = Student('ram','gopal')
s1.show()

s1.msg()
```

```
first name is  ram
last name is  gopal
ram gopal welcome to the class of 2022
```

In [ ]:

use of super keyword

The Python super() function returns objects represented in the parent’s class and enables multiple inheritances.

In [16]:
```
# creatinf an employee class

class Emp():
    def __init__(self,ID,name,add):
        self.ID = ID
        self.name = name
        self.add = add


# creating a child class

class Freelancer(Emp):
    def __init__(self,ID,name,add,email,gyear):
        super().__init__(ID,name,add)
        self.email = email
        self.gyear = gyear

    def show(self):
        print('Id is ',self.ID)
        print('Name is ',self.name)
        print('Address is ',self.add)
        print('Graduation year is ',self.gyear)


# creating an instance of the child class

harsh = Freelancer(123,'harsh watson','ahmedabad','harsh@gmail.com',2022)

harsh.show()
```

```
Id is  123
Name is  harsh watson
Address is  ahmedabad
Graduation year is  2022
```

In [ ]:

In [ ]:

Iterators in Python

Lists, tuples, dictionaries, and sets are all iterable objects. They are iterable containers which you can get an iterator from.

All these objects have a iter() method which is used to get an iterator:

Iterators are mainly of 2 types
__iter__()
__next__()

Q. Iterable vs Iterators in python

In [29]:
```
tup = ('I','am','a','good','man')
tupit = iter(tup)

print(next(tupit))
```

```
I
```

In [30]:
```
print(next(tupit))
print(next(tupit))
print(next(tupit))
print(next(tupit))
#print(next(tupit))
```

```
am
a
good
man
```

In [ ]:
```

```

In [31]:
```
st = 'hey there'
for i in st:
    print(i)
```

```
h
e
y

t
h
e
r
e
```

In [32]:
```
tup = ('I','am','a','good','man')
print(iter(tup))
```

```
<tuple_iterator object at 0x7f72fac98e50>
```

In [33]:
```
tup = ('I','am','a','good','man')
for i in tup:
    print(i)
```

```
I
am
a
good
man
```

In [ ]:
```

```

```
Using the generator fucntion
```

In [41]:
```

def store(a,b):
    yield a
    yield b

x = store(1,2)
print('Generator Obj : ',x)
```

```
Generator Obj :  <generator object store at 0x7f72facf1cf0>
```

In [42]:
```
# using next on the generator obj to print the elements

print(type(next(x)))
print(next(x))
```

```
<class 'int'>
2
```

In [ ]:
```

```

Another way to access the data from the generaor object

In [45]:
```
def store(a,b):
    yield a
    yield b

# converting the generator obj into the list to print the data
y = store(3,4)
print(list(y))
```

```
[3, 4]
```

In [ ]:
```

```

Yield -> generator function to generaot obj

In [54]:
```
# creating a range of yield stateents i.e generator fuctn

def show(a,b):
    while a<b:
        # generator function
        # yield generate a sequence of elements
        yield a
        a+=1

x = show(1,5)
print('Generator object ',x)

print(next(x))
print(next(x))


# as we can see once we access the value from the generator
# it gets removed from the generator object

print(list(x))
```

```
Generator object  <generator object show at 0x7f72fac88f90>
1
2
[3, 4]
```

In [ ]:
```

```

In [3]:
```
# practise yeild and generator fucntion

def genfucn(a,b):
    while a<b:
        yield a
        # generator fucntion generates the generator obj
        a+=1

# generating a sequence of numbers using the generator fucn
genfucn(1,6)
```

Out[3]:
```
<generator object genfucn at 0x7f1dd0258ba0>
```

In [5]:
```
# storing the data inside the generator fucn
l1 = genfucn(1,6)
print(l1)

# printing the data from the generator fucntion

for i in l1:
    print(i)
```

```
<generator object genfucn at 0x7f1dd0258ac0>
1
2
3
4
5
```

In [9]:
```
# another way of generating data from the generator fucn
l = genfucn(1,10)

# printing al; the data from the generator func using thee next statement
a = 1
b = 10

while a<b:
    print(next(l))
    a+=1
```

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

In [ ]:

In [ ]:

Exceptional Handling

```
Exception - runtime error, an abnormal conditon in a program
```

In [1]:
```
while True:
    try:
        a = int(input('\nEnter 1st number : '))
        b = int(input('Enter 2nd number : '))
        c = a+b
        print('Output is ',c)
    except ValueError:
        print('only integers accepted !!')
    else:
        print('Thank you')
        break
```

```
Enter 1st number : k
only integers accepted !!

Enter 1st number : 12
Enter 2nd number : 24
Output is  36
Thank you
```

In [ ]:
```

```

In [11]:
```
while True:
    try:
        a = int(input('\nEnter 1st number : '))
        b = int(input('Enter 2nd number : '))
        c = a/b
        print('Output is ',c)
    except ValueError:
        print('only integers accepted !!')
    except ArithmeticError:
        print('Divide by 0 not accepted')
    else:
        print('Thank you')
        break
```

```
Enter 1st number : 1
Enter 2nd number : 0
Divide by 0 not accepted

Enter 1st number : 1
Enter 2nd number : 2
Output is  0.5
Thank you
```

In [ ]:
```

```

banking example with exceeption handling

In [14]:
```
# define Python user-defined exceptions
class Error(Exception):
    """Base class for other exceptions"""
    pass


class ValueTooSmallError(Error):
    """Raised when the input value is too small"""
    pass


class ValueTooLargeError(Error):
    """Raised when the input value is too large"""
    pass

savings = 5000
while True:
    try:
        print('\nAvailable Balance : ',savings)
        print('-----------------------------------------')
        choice = input('\nEnter to Continue. y to exit ? ')
        print('-----------------------------------------')
        if choice=='y':
            break
        else:
            withdrawl = int(input('enter the amount : '))
            if withdrawl<=0:
                raise ValueTooSmallError
            elif withdrawl>savings:
                raise ValueTooLargeError
            else:
                savings = savings-withdrawl
                print(f'\n{withdrawl} has been debited!!')
                print('Updated Balance = ',savings)

    except ValueError:
        print('only integers accepted !!')
        print()
    except ValueTooSmallError:
        print("Amount should be greater than 0, try again!")
        print()
    except ValueTooLargeError:
        print("balance is Low, try again!")
        print('Current balacne = ',savings)
        print()
```

```
Available Balance :  5000
-----------------------------------------

Enter to Continue. y to exit ?
-----------------------------------------
enter the amount : 100

100 has been debited!!
Updated Balance =  4900

Available Balance :  4900
-----------------------------------------
```

```
Enter to Continue. y to exit ?
-----------------------------------------
enter the amount : 0
Amount should be greater than 0, try again!


Available Balance :  4900
-----------------------------------------

Enter to Continue. y to exit ?
-----------------------------------------
enter the amount : 4901
balance is Low, try again!
Current balacne =  4900


Available Balance :  4900
-----------------------------------------

Enter to Continue. y to exit ?
-----------------------------------------
enter the amount : 4900

4900 has been debited!!
Updated Balance =  0

Available Balance :  0
-----------------------------------------

Enter to Continue. y to exit ? y
-----------------------------------------
```

In [ ]:

Handling IOError using Exceptionhandling

In [18]:
```
# handling the exception in a file
# while writing the data to the file in 'r' mode

# 1] creating a test file

f = open('testfile.txt','w')
f.write('hello world')
f.close()
```

In [20]:
```
# 2] writeing the data inside the file in 'r' mode

try:
    f = open('testfile.txt','r')
    f.write('new data')
    f.close()

except IOError:
    print("Error can't write data inside the file")
else:
    print('data written successfully')
```

```
Error can't write data inside the file
```

In [21]:
```
# 3] Fix bu opening the file in 'a' mode

try:
    f = open('testfile.txt','a')
    f.write('new data')
    f.close()

except IOError:
    print("Error can't write data inside the file")
else:
    print('data written successfully')
```

```
data written successfully
```

In [23]:
```
with open('testfile.txt','r') as f:
    print(f.read())
```

```
hello worldnew data
```

In [ ]:
```

```