Introduction to Basic Python

In [1]:
```
print("nikhil")
```

```
nikhil
```

In [9]:
```
from platform import python_version
print(python_version())
```

```
3.11.3
```

In [10]:
```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'clas
s', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'r
aise', 'return', 'try', 'while', 'with', 'yield']
```

In [11]:
```
# print f()
print("hello world")
```

```
hello world
```

In [17]:
```
a=45
#a
print(a)
a
```

```
45
```

Out[17]:
```
45
```

In [18]:
```
print("a:",a,10,"nikhil",14.6)
```

```
a: 45 10 nikhil 14.6
```

In [19]:
```
# input f()

name =input()
print(name)
```

```
nikhil
nikhil
```

In [20]:
```
no = input("Enter a No. :")
print(no)
```

```
Enter a No. :154
154
```

In [23]:
```
name = input("Enter your name :")
r_no = input("enter Roll no. :")
print(name,r_no)
```

```
Enter your name :vibhishan
enter Roll no. :001
vibhishan 001
```

In [24]:
```
k
# j
# l
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[24], line 1
----> 1 k

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

In [29]:
```
# sep parameter

print(11,11,sep="?")
```

```
11?11
```

In [32]:
```
# escape sequences

print("hi\nharry")
print()
print("whats\tup")
```

```
hi
harry

whats	up
```

In [34]:
```
# output in ""

print("nikhil is \"student\"")
```

```
nikhil is "student"
```

In [ ]:

Practice Programs Using Input & Print

In [1]:
```
# input takes data as string

n1 =input("enter no 1 :")
n2 =input("enter no 2 :")
print(int(n1)+int(n2))
```

```
enter no 1 :5
enter no 2 :6
11
```

In [2]:
```
# average of three no.

a1 =int(input("enter no 1 :"))
a2 =int(input("enter no 2 :"))
a3 = int(input("enter no 3:"))
print((a1+a2+a3)/3)
```

```
enter no 1 :5
enter no 2 :6
enter no 3:9
6.666666666666667
```

In [ ]:
```

```

Fundamental (Basic) Datatypes

In [2]:

```
x=20 # int variable
print(x)
print(id(x))# address of variable
print(type(x)) # type of variable class
```

```
20
140735144564104
<class 'int'>
```

In [5]:

```
# binary data

n=0b100 # default value is decimal
print(n)
print(bin(n))
print(id(n))
print(type(n))
```

```
4
0b100
140735144563592
<class 'int'>
```

In [6]:

```
# octal data

n=0o1007
print(n)
print(oct(n))
print(id(n))
print(type(n))
```

```
519
0o1007
1678436928080
<class 'int'>
```

In [7]:

```
# hexa decimal data

n=0x1007
print(n)
print(hex(n))
print(id(n))
print(type(n))
```

```
4103
0x1007
1678436928208
<class 'int'>
```

In [8]:

```
hx = 0x1007
print(x)
print("binary :",bin(hx))
print("octal :",oct(hx))
print("hexa decimal :",hex(hx))
print(id(hx))
print(type(hx))
```

```
20
binary : 0b1000000000111
octal : 0o10007
hexa decimal : 0x1007
1678436928720
<class 'int'>
```

In [10]:

```
a = 6.8 #float data type
print(a)
print(id(a))
print(type(a))
```

```
6.8
1678434748880
<class 'float'>
```

In [12]:

```
cmp = 3+6j  # complex datatype
print(cmp)
print(cmp.real,cmp.imag)  # gives real & imaginary part separately
print(id(cmp))
print(type(cmp))
```

```
(3+6j)
3.0 6.0
1678436929392
<class 'complex'>
```

In [28]:

```
kl = 3+6.2j  # complex datatype
oct_cmp = 0x10065+5j
print(cmp)
print(type(cmp.real)) # type is float
print(type(cmp.imag))
print(type(kl.real))
print(type(kl.imag))
print(oct_cmp.real)
print(type(oct_cmp.real))
print(oct_cmp.imag)
print(type(oct_cmp.imag))
```

```
(3+6j)
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
65637.0
<class 'float'>
5.0
<class 'float'>
```

In [18]:

```
b=True # boolian datatype (True/False)
print(b)
print(id(b))
print(type(b))
```

```
True
140735143094816
<class 'bool'>
```

In [30]:

```
s = "nikhil vishwakarma"  # string data type
print(s)
print(id(s))
print(type(s))
```

```
nikhil vishwakarma
1678450690928
<class 'str'>
```

In [31]:

```
s = 'nikhil vishwakarma'  # string data type
print(s)
print(id(s))
print(type(s))
```

```
nikhil vishwakarma
1678450694928
<class 'str'>
```

In [32]:

```
s = '''nikhil vishwakarma'''  # string data type
print(s)
print(id(s))
print(type(s))
```

```
nikhil vishwakarma
1678438175072
<class 'str'>
```

In [33]:

```
s = """nikhil vishwakarma""" # string data type
print(s)
print(id(s))
print(type(s))
```

```
nikhil vishwakarma
1678450701008
<class 'str'>
```

In [35]:

```
# multi line string is allowed in python
poem = """twincle twincle little star
how i wonder what you are
up above the sky"""

print(poem)
```

```
twincle twincle little star
how i wonder what you are
up above the sky
```

In [ ]:

```

```

Advance Data Types

In [5]:
```
# list []

x = [1,3,6,["str",[10,52,True],"like",25.6,False],None]

print(x)
print(type(x))
print(x[1])
print(x[1])
print(x[3][1])
print(x[3][1][2])
```

```
[1, 3, 6, ['str', [10, 52, True], 'like', 25.6, False], None]
<class 'list'>
3
3
[10, 52, True]
True
```

In [27]:
```
k =[]  # empty list
print(k)
print(id(k))
print(type(k))
```

```
[]
2027985036672
<class 'list'>
```

In [28]:
```
k =[1,2],"string",56.3  # list
print(k)
print(id(k))
print(type(k))
```

```
([1, 2], 'string', 56.3)
2027984362368
<class 'tuple'>
```

In [11]:
```
x =[1,4.6,True,"hi"]
print(x)
x.append("good morning")  # add elements in the end
print(x)
x.append("good morning")
print(x)
```

```
[1, 4.6, True, 'hi']
[1, 4.6, True, 'hi', 'good morning']
[1, 4.6, True, 'hi', 'good morning', 'good morning']
```

In [12]:
```
# tupple ()
t = (1,2,3,4)
print(t)
print(id(t))
print(type(t))
```

```
(1, 2, 3, 4)
2027984268128
<class 'tuple'>
```

In [13]:
```
t = (1,2,"string",True)
print(t)
print(id(t))
print(type(t))
```

```
(1, 2, 'string', True)
2027984267728
<class 'tuple'>
```

In [15]:
```
print(t)
print(t.append(53))  # throws an error
```

```
(1, 2, 'string', True)

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[15], line 2
      1 print(t)
----> 2 print(t.append(53))  # throws an error

AttributeError: 'tuple' object has no attribute 'append'
```

Tupple mein modification nhi kar skte -> immutable List mein modification kar skte h -> mutable

In [18]:
```
# set {}
# set is un-ordered whereis -> List and Tupples are ordered

s = {1,2,3,4,5}
print(s)
print(id(s))
print(type(s))
```

```
{1, 2, 3, 4, 5}
2027978961888
<class 'set'>
```

In [20]:
```
p = {1,2,3,4,5}
print(p)
print(id(p))
print(type(p))
# access is not be able element wise
print(p[1])
```

```
{1, 2, 3, 4, 5}
2027978963456
<class 'set'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 6
      4 print(type(p))
      5 # access is not be able element wise
----> 6 print(p[1])

TypeError: 'set' object is not subscriptable
```

In [21]:
```
p = {1,2,3,5,4,8,9,3,3}
print(p)
print(id(p))
print(type(p))
```

```
{1, 2, 3, 4, 5, 8, 9}
2027984929280
<class 'set'>
```

In [23]:
```
p = {1,2,3,4,5,"hello"}
print(p)
print(id(p))
print(type(p))
```

```
{1, 2, 3, 4, 5, 'hello'}
2027984931072
<class 'set'>
```

In [26]:
```
# Set is mutable
p = {1,2,3,4,5,"hello"}
p.add("world")  # to add elements in set
print(p)
print(id(p))
print(type(p))
```

```
{1, 2, 3, 4, 5, 'world', 'hello'}
2027978963456
<class 'set'>
```

In [30]:
```
# frozenset is immutable (freezed set)
s= {5,3,4,2,5,1}
print(s)
print(type(s))
f = frozenset(s)  # after declaring frozenset we cannot update anything in set
print(f)
print(type(f))
```

```
{1, 2, 3, 4, 5}
<class 'set'>
frozenset({1, 2, 3, 4, 5})
<class 'frozenset'>
```

In [31]:
```
# Dictionary (key pair values)

d = {1:"vasima",2:"swati",3:"ravi"}
print(d)
print(id(d))
print(type(d))
```

```
{1: 'vasima', 2: 'swati', 3: 'ravi'}
2027984973696
<class 'dict'>
```

In [32]:
```
d = {"python":"vasima","physics":"swati","data":[1,4,6]}
print(d)
print(id(d))
print(type(d))
```

```
{'python': 'vasima', 'physics': 'swati', 'data': [1, 4, 6]}
2027984201792
<class 'dict'>
```

In [33]:
```
d["data"]
```

Out[33]:
```
[1, 4, 6]
```

In [35]:
```
d["data"]=[1,"hello"]
print(d)
```

```
{'python': 'vasima', 'physics': 'swati', 'data': [1, 'hello']}
```

only (list, set, dictionary) are mutable

whenever int,float,string,complex,boolian,tuple,frozenset are immutable

In [38]:
```
# immutability of integer
x=7
print(id(x))
y=7
print(id(y))
x=99
print(id(x))
```

```
140724185240552
140724185240552
140724185243496
```

In [39]:
```
# mutability of list
l=[7,5,5.6]
print(l)
print(id(l))
l.append(88)
print(l)
print(id(l))
```

```
[7, 5, 5.6]
2027985001024
[7, 5, 5.6, 88]
2027985001024
```

In [40]:
```
# None datatype
x = None
print(x)
print(id(x))
print(type(x))  # it works like a holder, later on any of datatype will be store in i
```

```
None
140724183853808
<class 'NoneType'>
```

In [43]:
```
# unicode
chr(65)
```

Out[43]:
```
'A'
```

In [45]:
```
ord("G")
```

Out[45]:
```
71
```

In [46]:
```
chr(0)
```

Out[46]:
```
'\x00'
```

In [44]:
```
print("\N{grinning face}")
```

😀

In [47]:
```
print("\N{slightly smiling face}")
```

🙂

In [49]:
```
# range f() -> it is a f() (considered as data type)

for i in range(0,10):
    print(i,end=" ")
```

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

In [51]:
```
for i in range(0,20,2): # 0 to (n-1) ,skiprange
    print(i,end=" ")
```

```
0 2 4 6 8 10 12 14 16 18
```

In [ ]:

Practice Programs on Datatypes & Type()

In [3]:
```
# write a program to check type of pi value

import math
print(math.pi)
x = math.pi
print(x)
print(type(x))
```

```
3.141592653589793
3.141592653589793
<class 'float'>
```

In [4]:
```
# write a program to check type of boolian value

t = True
f = False
print("t =",t,"f =",f)
print(type(t))
```

```
t = True f = False
<class 'bool'>
```

In [ ]:
```

```

Type Casting

In [7]:
```
# type casting
x=6.9
print(x)
print(type(x))
print(int(x))  # it truncate(cut) the float value
print(type(int(x)))
```

```
6.9
<class 'float'>
6
<class 'int'>
```

In [4]:
```
y = int(x) # storing value as integer
print(y)
print(type(y))
```

```
6
<class 'int'>
```

In [8]:
```
import math
print(math.floor(x)) # it rounds down the float value
print(type(math.floor(x)))
print(int(x))
print(type(int(x)))
```

```
6
<class 'int'>
6
<class 'int'>
```

In [12]:
```
s= "11"
# print(math.floor(s))  # throws error
# print(type(math.floor(s)))
print(int(s))
print(type(int(s)))
```

```
11
<class 'int'>
```

In [15]:
```
x = -3.4
print(math.floor(x)) # it gives smaller value
print(type(math.floor(x)))
print(int(x))  # it discard float value
print(type(int(x)))
```

```
-4
<class 'int'>
-3
<class 'int'>
```

In [14]:
```
x = 6
print(math.floor(x))
print(type(math.floor(x)))
print(int(x))
print(type(int(x)))
```

```
6
<class 'int'>
6
<class 'int'>
```

In [16]:
```
x="15"  # covertable string to int
print(x)
print(type(x))
print(int(x))
print(type(int(x)))
```

```
15
<class 'str'>
15
<class 'int'>
```

In [18]:
```
x="nikhil"  # non-covertable string to int
print(x)
print(type(x))
# print(int(x))  # throws error
# print(type(int(x)))
```

```
nikhil
<class 'str'>
```

In [19]:
```
x="11.95"  # non-covertable string to int
print(x)
print(type(x))
# print(int(x))  # throws error
# print(type(int(x)))
```

```
11.95
<class 'str'>

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[19], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(int(x))  # throws error
      5 print(type(int(x)))

ValueError: invalid literal for int() with base 10: '11.95'
```

In [20]:
```
x=True  # bool to int
print(x)
print(type(x))
print(int(x))  # convert True -> 1
print(type(int(x)))
```

```
True
<class 'bool'>
1
<class 'int'>
```

In [21]:
```
x=False  # bool to int
print(x)
print(type(x))
print(int(x))  # throws error
print(type(int(x)))
```

```
False
<class 'bool'>
0
<class 'int'>
```

In [23]:
```
x=5+6j  # complex to int
print(x)
print(type(x))
print(int(x))  # throws error
print(type(int(x)))
```

```
(5+6j)
<class 'complex'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[23], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(int(x))  # throws error
      5 print(type(int(x)))

TypeError: int() argument must be a string, a bytes-like object or a real number, n
ot 'complex'
```

In [25]:
```
# type casting in float
x=56  # int to float
print(x)
print(type(x))
print(float(x))
print(type(float(x)))
```

```
56
<class 'int'>
56.0
<class 'float'>
```

In [27]:
```
x=True  # bool to float
print(x)
print(type(x))
print(float(x))  # it converts True -> 1.0
print(type(float(x)))
```

```
True
<class 'bool'>
1.0
<class 'float'>
```

In [28]:
```
x=False  # bool to float
print(x)
print(type(x))
print(float(x))  # it converts False -> 0.0
print(type(float(x)))
```

```
False
<class 'bool'>
0.0
<class 'float'>
```

In [29]:
```
x="11.8"  # appropriate string to float
print(x)
print(type(x))
print(float(x))
print(type(float(x)))
```

```
11.8
<class 'str'>
11.8
<class 'float'>
```

In [31]:
```
x="nikhil"  # inappropriate string to float
print(x)
print(type(x))
print(float(x))  # throws error
print(type(float(x)))
```

```
nikhil
<class 'str'>

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[31], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(float(x))  # throws error
      5 print(type(float(x)))

ValueError: could not convert string to float: 'nikhil'
```

In [32]:
```
x="11"  # appropriate string to float
print(x)
print(type(x))
print(float(x))
print(type(float(x)))
```

```
11
<class 'str'>
11.0
<class 'float'>
```

In [35]:
```
x=11.6+2.9j  # complex to float
print(x)
print(type(x))
print(float(x))  # throws error
print(type(float(x)))
```

```
(11.6+2.9j)
<class 'complex'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(float(x))  # throws error
      5 print(type(float(x)))

TypeError: float() argument must be a string or a real number, not 'complex'
```

In [36]:
```
# type casting in complex

x=7  # int to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
7
<class 'int'>
(7+0j)
<class 'complex'>
```

In [37]:
```
x=74.5  # float to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
74.5
<class 'float'>
(74.5+0j)
<class 'complex'>
```

In [38]:
```
x=True  # bool to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
True
<class 'bool'>
(1+0j)
<class 'complex'>
```

In [39]:
```
x=False  # bool to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
False
<class 'bool'>
0j
<class 'complex'>
```

In [40]:
```
x="54"  # approprite string to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
54
<class 'str'>
(54+0j)
<class 'complex'>
```

In [43]:
```
x="54"  # approprite string to complex
print(x)
print(type(x))
y=complex(x)
print(y)
print(type(y))
print(y.real,y.imag)
print(type(y.real),type(y.imag))
```

```
54
<class 'str'>
(54+0j)
<class 'complex'>
54.0 0.0
<class 'float'> <class 'float'>
```

In [41]:
```
x="54.7"  # approprite string to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
54.7
<class 'str'>
(54.7+0j)
<class 'complex'>
```

In [44]:
```
x="vasima"  # inapproprite string to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
vasima
<class 'str'>

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[44], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(complex(x))
      5 print(type(complex(x)))

ValueError: complex() arg is a malformed string
```

In [45]:
```
# typr casting in bool
x=1  # int to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
1
<class 'int'>
True
<class 'bool'>
```

In [47]:
```
x=0  # int to bool
print(x)
print(type(x))
print(bool(x))  # only for zero gives false in bool f()
print(type(bool(x)))
```

```
0
<class 'int'>
False
<class 'bool'>
```

In [48]:
```
x=562  # int to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
562
<class 'int'>
True
<class 'bool'>
```

In [49]:
```
x=-213  # int to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
-213
<class 'int'>
True
<class 'bool'>
```

In [50]:
```
x=25.36  # float to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
25.36
<class 'float'>
True
<class 'bool'>
```

In [51]:
```
x=25+56j  # complex to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
(25+56j)
<class 'complex'>
True
<class 'bool'>
```

In [52]:
```
x=25+0j  # complex to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
(25+0j)
<class 'complex'>
True
<class 'bool'>
```

In [53]:
```
x=0+56j  # complex to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
56j
<class 'complex'>
True
<class 'bool'>
```

In [54]:
```
x=0+0j  # complex to bool
print(x)
print(type(x))
print(bool(x))  # only 0+0j gives False
print(type(bool(x)))
```

```
0j
<class 'complex'>
False
<class 'bool'>
```

In [55]:
```
x="oi"  # string to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
oi
<class 'str'>
True
<class 'bool'>
```

In [56]:
```
x="256"  # string to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
256
<class 'str'>
True
<class 'bool'>
```

In [57]:
```
x="0"  # string to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
0
<class 'str'>
True
<class 'bool'>
```

In [58]:
```
x="1"  # string to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
1
<class 'str'>
True
<class 'bool'>
```

In [66]:
```
x=""  # string to bool
print(x)
print(type(x))
print(bool(x))  # it returns False
print(type(bool(x)))
```

```
<class 'str'>
False
<class 'bool'>
```

In [60]:
```
# string type casting string

x = 88  # int to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
88
<class 'int'>
88
<class 'str'>
```

In [61]:
```
x = 88.26  # float to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
88.26
<class 'float'>
88.26
<class 'str'>
```

In [62]:
```
x = 88+6j  # complex to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
(88+6j)
<class 'complex'>
(88+6j)
<class 'str'>
```

In [63]:
```
x = True  # bool to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
True
<class 'bool'>
True
<class 'str'>
```

Programs on Type Casting

In [70]:
```
# write a program to calculate Area of circle. Data should be taken from user but
# the value of "pi"taken by python itself

import math
# print(math.pi)
r=float(input("enter radius of a circle:"))
area = math.pi*r**2
print("area of circle is:",area)
```

```
3.141592653589793
enter radius of a circle:5
area of circle is: 78.53981633974483
```

In [3]:
```
# write a program to calculate the area of triangle, length of side should
# be given by user and square root f() taken by python itself

import math
print(math.sqrt(25))

a=float(input("enter length of side one:"))
b=float(input("enter length of side two:"))
c=float(input("enter length of side three:"))

s = (a+b+c)/2
area =(s*(s-a)*(s-b)*(s-c))
print("area of triangle is:",math.sqrt(area))
```

```
5.0
enter length of side one:10
enter length of side two:12
enter length of side three:9
area of triangle is: 44.039045175843675
```

In [2]:
```
# write a program to calculate SI, data should be taken from the user

p=float(input("Enter principle amount:"))
r=float(input("Enter rate of interest:"))
t=int(input("enter time:"))
si=(p*r*t)/100
print("your SI is:",si)
```

```
Enter principle amount:10000
Enter rate of interest:15
enter time:2
your SI is: 3000.0
```

Operators in Python

In [4]:
```
# arithmatic operator
# (+, -, *, /, //, %, **)

x = 20
y = 4
print("value of x+y :",x+y)
print("value of x-y :",x-y)
print("value of x*y :",x*y)
print("value of x/y :",x/y)  # gives float in output
print("value of x//y :",x//y) # float division operator (it rounds down the value)
print("value of x%y :",x%y)   # modulo operator gives reminder
print("value of x**y :",x**y) # x^y
```

```
value of x+y : 24
value of x-y : 16
value of x*y : 80
value of x/y : 5.0
value of x//y : 5
value of x%y : 0
value of x**y : 160000
```

In [5]:
```
33.7+5
```

Out[5]:
```
38.7
```

In [6]:
```
True+6
```

Out[6]:
```
7
```

In [7]:
```
print("nikhil"+"vishwakarma") # concatienation
```

```
nikhilvishwakarma
```

In [8]:
```
print("nikhil"+"vishwakarma"+"aids")
```

```
nikhilvishwakarmaaids
```

In [9]:
```
print("nikhil"+"vishwakarma"+88)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 print("nikhil"+"vishwakarma"+88)

TypeError: can only concatenate str (not "int") to str
```

In [10]:
```
"s"*6  # imp
```

Out[10]:
```
'ssssss'
```

In [11]:
```
"nikhil "*10
```

Out[11]:
```
'nikhil nikhil nikhil nikhil nikhil nikhil nikhil nikhil nikhil nikhil '
```

In [13]:
```
print("nikhil\n"*10)
```

```
nikhil
nikhil
nikhil
nikhil
nikhil
nikhil
nikhil
nikhil
nikhil
nikhil
```

In [14]:
```
"nikhil "*10*" "*9
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 "nikhil "*10*" "*9

TypeError: can't multiply sequence by non-int of type 'str'
```

In [20]:
```
print(30/4)  # always gives float value
print(30.6/4)
print(30//4)   # perform division ,floors down the value,
                # return int bcz both operants are int
print(30.6//4)  # perform division ,floors down the value,
                 # return float bcz one operant is float
print(30.6//4.2)  # perform division ,floors down the value,
                   # return float bcz both operants are float
print(30//4.2)     # perform division ,floors down the value,
                    # return float bcz one operant is float
```

```
7.5
7.65
7
7.0
7.0
7.0
```

In [26]:
```
print(66%6)
print(66.7%6)
print(66%6.2)
print(66.5%6.1)
print(-66%6)
print(67%-6)
print(-67%-6)
print(-66.45%6)
```

```
0
0.7000000000000028
3.9999999999999982
5.5000000000000036
0
-5
-1
5.549999999999997
```

In [27]:
```
# relational (comparison) operator

#  (<, >, >=, <=, ==, !=) it also returns boolean value

a = 44
b =  6
print(a<b)
print(a>b)
print(a<=b)
print(a>=b)
print(a==b)
print(a!=b)
```

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

In [29]:
```
a = "m"  # it uses unicode for comparison
b =  "n"
print(a<b)
print(a>b)
print(a<=b)
print(a>=b)
print(a==b)
print(a!=b)
```

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

In [30]:
```
a = "Nikhil"  # it uses unicode for comparison
b =  "nikhil"
print(a<b)
print(a>b)
print(a<=b)
print(a>=b)
print(a==b)
print(a!=b)
```

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

In [32]:
```
a = "Nikhim"
b =  "Nikhil"
print(a<b)
print(a>b)
print(a<=b)
print(a>=b)
```

```
False
True
False
True
```

In [34]:
```
a = "Nikhim"
b =  "Nikhil"
print(a==b)
print(a!=b)
```

```
False
True
```

In [35]:
```
a = "Nikhim"
b =  5
print(a<b)
print(a>b)
print(a<=b)
print(a>=b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 3
      1 a = "Nikhim"
      2 b =  5
----> 3 print(a<b)
      4 print(a>b)
      5 print(a<=b)

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [36]:
```
a = "N"
b = 14
print(a<b)  # it see datatype
print(a>b)  # it see datatype
print(a<=b)  # it see datatype
print(a>=b)  # it see datatype
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 3
      1 a = "N"
      2 b = 14
----> 3 print(a<b)
      4 print(a>b)
      5 print(a<=b)

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [38]:
```
a = "N"
b =  14
print(a==b)  # it does not see datatype
print(a!=b)  # it does not see datatype
```

```
False
True
```

In [39]:
```
# chaining is allowed in python

1<2<3<5<7
```

Out[39]:
```
True
```

In [40]:
```
1<2<3<5>7
```

Out[40]:
```
False
```

In [41]:
```
x= 1
y= 5
z = 9
print(x<y<z)
print(x<y>z)
print(x>y>z)
print(x>y<z)
```

```
True
False
False
False
```

In [42]:
```
# logical operators (and , or, not)
# it applies on bool

print(True and True)
print(True and False)
print(False and True)
print(False and False)
```

```
True
False
False
False
```

In [43]:
```
print(True or True)
print(True or False)
print(False or True)
print(False or False)
```

```
True
True
True
False
```

In [44]:
```
print(not True)
print(not False)
```

```
False
True
```

In [46]:
```
# python speciality
x = 3
y = 6
print(x and y)
print(x or y)
```

```
6
3
```

In [50]:
```
x = 6
y = 3
print(x and y)
print(x or y)
```

```
3
6
```

In [47]:
```
x = 0
y = 6
print(x and y)
print(x or y)
```

```
0
6
```

In [48]:
```
x = 3
y = 0
print(x and y)
print(x or y)
```

```
0
3
```

In [54]:
```
5<10>15<33  #(True>15 is False) if any of condition is false then the output is False
```

Out[54]:
```
False
```

In [52]:
```
"Red"or"White"
```

Out[52]:
```
'Red'
```

In [53]:
```
"Red"and"White"
```

Out[53]:
```
'White'
```

In [55]:
```
# assignment operator
# =, Compound assignment oprator (+=, -=, *=, /=, )

x = 66  # assign the value 66 in identifier x
print(x)
x+=2
print(x)
x-=3
print(x)
x*=2
print(x)
x/=2
print(x)
x//=2
print(x)
```

```
66
68
65
130
65.0
32.0
```

In [56]:
```
a,b,c=10,20,30
print(a)
print(b)
print(c)
```

```
10
20
30
```

In [57]:
```
x++  # not available in python
```

```
  Cell In[57], line 1
    x++  # not available in python
         ^
SyntaxError: invalid syntax
```

In [59]:
```
# swap two values
print(a)
print(b)
a,b=b,a
print(a)
print(b)
```

```
20
10
10
20
```

In [3]:
```
a =True
b =True
print(a-b)
```

```
0
```

In [5]:
```
a =False
b =True
print(a-b)
```

```
-1
```

In [6]:
```
b =False
a =True
print(a-b)
```

```
1
```

In [8]:
```
# bitwise Operator

print(4&5) # 4->100, 5->101 it apply bit by bit (1&1)->1, (0&0)->0, (0&1)->0 ==> 100-
print(4|5) # 4->100, 5->101 it apply bit by bit (1|1)->1, (0|0)->0, (0|1)->1 ==> 101-
```

```
4
5
```

In [10]:
```
print(8&4)
print(8|4)
```

```
0
12
```

In [11]:
```
# identity operator
# (is)-> it checks identity
a=5
b=6
print(id(a))
print(id(b))
a is b
```

```
140708480324520
140708480324552
```

Out[11]:
```
False
```

In [13]:
```
a=5
b=5
print(id(a))
print(id(b))
a is b
```

```
140708480324520
140708480324520
```

Out[13]:
```
True
```

In [14]:
```
l =[1,5,7]
k =[2,5,6]
print(id(l))
print(id(k))
l is k
```

```
1640695947328
1640696045568
```

Out[14]:
```
False
```

In [15]:
```
l =[1,5,7]
k =[1,5,7]
print(id(l))
print(id(k))
l is k
```

```
1640696530752
1640695662720
```

Out[15]:
```
False
```

In [16]:
```
l ={1,5,7}
k ={2,5,6}
print(id(l))
print(id(k))
l is k
```

```
1640695194880
1640695805120
```

Out[16]:
```
False
```

In [17]:
```
l ={1,5,7}
k ={1,5,7}
print(id(l))
print(id(k))
l is k
```

```
1640695806240
1640695194880
```

Out[17]:
```
False
```

In [18]:
```
d1 ={"s":1001,"u":1002,"v":1003}
d2 ={"a":1009,"b":1008,"c":1007}
print(id(d1))
print(id(d2))
d1 is d2
```

```
1640696163712
1640696534592
```

Out[18]: False

In [19]:
```
d1 ={"s":1001,"u":1002,"v":1003}
d2 ={"s":1001,"u":1002,"v":1003}
print(id(d1))
print(id(d2))
d1 is d2
```

```
1640695670912
1640695662272
```

Out[19]: False

In [22]:
```
f1 =frozenset({"s","u","v"})
f2 =frozenset({"s","u","v"})
print(id(f1))
print(id(f2))
f1 is f2
```

```
1640696721952
1640696723072
```

Out[22]: False

In [23]:
```
t1 =("s","u","v")
t2 =("s","u","v")
print(id(t1))
print(id(t2))
t1 is t2
```

```
1640702208960
1640702163456
```

Out[23]: False

In [24]:
```
n1 =None
n2 =None
print(id(n1))
print(id(n2))
n1 is n2
```

```
140708478937840
140708478937840
```

Out[24]: True

In [25]:
```
s1 ="nikhil"
s2 ="nikhil"
print(id(s1))
print(id(s2))
s1 is s2
```

```
1640701770480
1640701770480
```

Out[25]: True

In [26]:
```
c1 =5+6j
c2 =5+6j
print(id(c1))
print(id(c2))
c1 is c2
```

```
1640698850480
1640698850160
```

Out[26]: False

In [30]:
```
n1 =85.6
n2 =85.6
print(id(n1))
print(id(n2))
n1 is n2
```

```
1640695589904
1640695581616
```

Out[30]: False

In [29]:
```
n1 =True
n2 =True
print(id(n1))
print(id(n2))
n1 is n2
```

```
140708478855712
140708478855712
```

Out[29]: True

In [28]:
```
# in operator
print("n" in "nikhil")
```

```
True
```

In [31]:
```
for i in range(2,6):
    print(i)
```

```
2
3
4
5
```

In [ ]:

Programs on Operators in Python

In [69]:
```
# 1) write a program to remove last digit from a given number

n = int(input("Enter a number:"))
k = n
k //=10
print("Your value is:",k)
```

```
Enter a number:254
Your value is: 25
```

In [70]:
```
# 2) write a program to get last digit from a given number

n = int(input("Enter a number:"))
k = n
k %=10
print("Last digit is:",k)
```

```
Enter a number:25634
Last digit is: 4
```

In [73]:
```
# 3) write a program which takes a 3 digit number from user and display only its 1st

n = int(input("Enter a number:"))
k = n
k //=10
k //=10
print("1st digit is:",k)
```

```
Enter a number:125
1st digit is: 1
```

In [72]:
```
# 4) write a program which takes a 3 digit number from user and display only its midd

n = int(input("Enter a number:"))
k = n
k //=10
k %=10
print("Middle digit is:",k)
```

```
Enter a number:245
Middle digit is: 4
```

In [75]:
```
# 5) write a program to swap data of two variables

a1 = input("Enter 1st value:")
b1 = input("Enter 2nd value:")
k = a1
a1 = b1
b1 = k
print("After Swapping\n1st value:",a1)
print("2nd value:",b1)
```

```
Enter 1st value:256
Enter 2nd value:nikhil
After Swapping
1st value: nikhil
2nd value: 256
```

In [2]:
```
# ^ swap values without using temporary variable
a =10
b =20
print(a,b)
a=a+b
b=a-b
a=a-b
print(a,b)
```

```
10 20
20 10
```

Programs of in & is Operator

In [33]:
```
# 6) write a program to print "True" if the string entered by user containing "py"
s = input("Enter a text:")
print("py" in s)
```

```
Enter a text:hgpyjhuykp
True
```

In [42]:
```
# 7) write a program to input two strings from the user and display whether the two v
s1 = input("Enter text1:")
s2 = input("Enter text1:")
print(id(s1))
print(id(s2))
print(s1 is s2)
```

```
Enter text1:nikhil
Enter text1:nikhil
1640702212912
1640702213552
False
```

In [41]:
```python
# 8) write a program to input five integers from the user and execute this expression
a = int(input("enter 1st no.:"))
b = int(input("enter 2nd no.:"))
c = int(input("enter 3rd no.:"))
d = int(input("enter 4th no.:"))
e = int(input("enter 5th no.:"))

print(a<b<c>d<e)
```

```
enter 1st no.:5
enter 2nd no.:6
enter 3rd no.:7
enter 4th no.:5
enter 5th no.:10
True
```

In [ ]:

Decision Control

--: if Statement

In [1]:
```
# IF
a = int(input("enter a no.:"))
if a>0:
    print("Number is Positive")
```

```
enter a no.:2
Number is Positive
```

```
Python follows indent(indentation)(tab)

block of code represents by ":" -> colon
```

In [3]:
```
if a>0:
print("Number is Positive")
```

```
  Cell In[3], line 2
    print("Number is Positive")
    ^
IndentationError: expected an indented block after 'if' statement on line 1
```

In [5]:
```
b = int(input("enter a no.:"))
if b>0:
    print("Number is Positive")
    print("inside if")
print("outside if")
```

```
enter a no.:0
outside if
```

In [8]:
```
if b>=0:
    print("Number is Positive")
    print("inside if")
print("outside if")
```

```
Number is Positive
inside if
outside if
```

--: if-else Statement

In [9]:
```
# IF-ELSE
a = int(input("enter a no.:"))
if a>0:
    print("Number is Positive")
else:
    print("Number is Nagative")
```

```
enter a no.:12
Number is Positive
```

In [10]:
```
a = int(input("enter a no.:"))
if a>0:
    print("Number is Positive")
else:
    print("Number is Nagative")
```

```
enter a no.:-12
Number is Nagative
```

--: if-elif-else Statement

In [14]:
```
# IF-ELIF-ELSE

a = int(input("enter a no.:"))
if a>0:
    print("Number is Positive")
elif a<0:
    print("Number is Nagative")
else:
    print("Number is Zero")
```

```
enter a no.:-5
Number is Nagative
```

--: nested if-else Statement

In [1]:
```
# Nested IF-ELSE
i=int(input("Enter a no.: "))
if(i<10):
        if i%2==0:
            print("Even below 10")
        else:
            print("odd below 10")
else:
    if i%2==0:
        print("Even over 10")
    else:
        print("odd over 10")
```

```
Enter a no.: 25
odd over 10
```

Programs of Decision Control

In [14]:
```
# 1) write a program to check whether the given number is +ve, -ve, or zero

z = int(input("Enter a no.:"))
if z>0:
    print("Given no. is Positive")
elif z<0:
    print("Given no. is Negative")
else:
    print("Given no. is Zero")
```

```
Enter a no.:0
Given no. is Zero
```

In [15]:
```
# 2) write a program to check whether the given number is divisible by 5 or not

c=int(input("Eneter a no.:"))
if(c%5)==0:
    print("yes")
else:
    print("no")
```

```
Eneter a no.:564
no
```

In [16]:
```
# 3) write a program to check whether  a number is even or odd
x=int(input("Enter a no.:"))
if x%2==0:
    print("Number is  Even")
else:
    print("Number is  Odd")
```

```
Enter a no.:254
Number is  Even
```

In [17]:
```
# 4) write a program to print the grater b/w two no. if both the numbers
# are equal then print the number only once

a=int(input("Enter 1st no. :"))
b=int(input("Enter 2nd no. :"))
if a>b:
    print("-->",a)
else:
    print("-->",b)
```

```
Enter 1st no. :12
Enter 2nd no. :16
--> 16
```

In [20]:
```
# 5) write a program to print two given words from user in dictionary order

s=input("Enter text 1 :")
t=input("Enter text 2 :")
if s>t:
    print(s)
    print(t)
else:
    print(t)
    print(s)
```

```
Enter text 1 :nikhil
Enter text 2 :nilikh
nilikh
nikhil
```

In [22]:
```
# 6) write a program to check whether the given no. is a 3 digit no. or not
y=int(input("Enter a no.:"))
if y>=100 and y<=999:
    print("You entered 3 digit no.")
else:
    print("Wrong input")
```

```
Enter a no.:452
You entered 3 digit no.
```

In [25]:
```
# 7) write a program to check whether the quardatric eq. has two real & distinct root
# real & equal roots or imaginary roots
a=int(input("Enter Multiplier of x^2 :"))
b=int(input("Enter Multiplier of x :"))
c=int(input("Enter Constant value :"))
root = b**2-(4*a*c)
if root<0:
    print("imaginary roots")
elif root>0:
    print("two real and distinct roots")
else:
    print("real and equal roots")
```

```
Enter Multiplier of x^2 :5
Enter Multiplier of x :6
Enter Constant value :9
imaginary roots
```

In [7]:
```
# 8) write a program to check whether the given year is leap year or not
y=int(input("Enter year"))
if(y%100)>0 and y%4==0:
    print("Year is Leap Year")
elif(y%100)==0 and y%400==0:
    print("Year is Leap Year")
else:
    print("Year is not a leap year")
```

```
Enter year2600
Year is not a leap year
```

In [12]:
```python
# 9) write a program to print greatest out of three no. if all the no. are
# same you need to print it once
x=int(input("Enter no. 1:"))
y=int(input("Enter no. 2:"))
z=int(input("Enter no. 3:"))
if x>=y and x>=z:
    print(x)
elif x<=y and y>=z:
    print(y)
else:
    print(z)
```

```
Enter no. 1:8
Enter no. 2:8
Enter no. 3:8
8
```

In [ ]:

Loop Statement

In [1]:
```
# Loops in python -> FOR LOOP and WHILE LOOP

s = "nikhil"
for i in s:
    print(i)
```

```
n
i
k
h
i
l
```

In [2]:
```
s = "nikhil"
for _ in s:
    print(_)
```

```
n
i
k
h
i
l
```

In [4]:
```
l=[1,2,3,4,5,6]
for i in l:
    print("list item",i,":",i)
```

```
list item 1 : 1
list item 2 : 2
list item 3 : 3
list item 4 : 4
list item 5 : 5
list item 6 : 6
```

In [5]:
```
# implementing for loop using range f()

for i in range(0,5):
    print(i)
```

```
0
1
2
3
4
```

In [13]:
```
# print factorial of a given no.

n=int(input("Enter a no. :"))
f=1
if n==0 or n==1:
    f=1
else:
    for i in range(1,n+1):
        f*=i
print("factorial of",n,":",f)
```

```
Enter a no. :7
factorial of 7 : 5040
```

In [14]:
```
n=int(input("Enter a no. :"))
f=1
for i in range(1,n+1):
    f*=i
print("factorial of",n,":",f)
```

```
Enter a no. :5
factorial of 5 : 120
```

```
#    Nesting of For loops
```

In [15]:
```
for i in range(0,4):
    for j in range(0,5):
        print("i:",i,"j:",j)
```

```
i: 0 j: 0
i: 0 j: 1
i: 0 j: 2
i: 0 j: 3
i: 0 j: 4
i: 1 j: 0
i: 1 j: 1
i: 1 j: 2
i: 1 j: 3
i: 1 j: 4
i: 2 j: 0
i: 2 j: 1
i: 2 j: 2
i: 2 j: 3
i: 2 j: 4
i: 3 j: 0
i: 3 j: 1
i: 3 j: 2
i: 3 j: 3
i: 3 j: 4
```

In [4]:
```
n=int(input("Enter a no. :"))

for i in range(1,11):
    print(n,"X",i,"=",n*i)
```

```
Enter a no. :5
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
```

In [5]:
```
n=int(input("Enter a no. :"))

for i in range(2,n+1):
    for j in range(1,11):
        print(i,"X",j,"=",j*i)
    print()
```

```
Enter a no. :3
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 [3]:
```
n=int(input("Enter a no. :"))

for i in range(1,11):
    for j in range(2,n+1):
        print(j,"X",i,"=",j*i,end="")
        if j*i<=9:
            print(end="   |  ")
        elif i==10:
            print(end=" |  ")
        else:
            print(end="  |  ")
    print()
```

```
Enter a no. :6
2 X 1 = 2   |  3 X 1 = 3   |  4 X 1 = 4   |  5 X 1 = 5   |  6 X 1 = 6   |
2 X 2 = 4   |  3 X 2 = 6   |  4 X 2 = 8   |  5 X 2 = 10  |  6 X 2 = 12  |
2 X 3 = 6   |  3 X 3 = 9   |  4 X 3 = 12  |  5 X 3 = 15  |  6 X 3 = 18  |
2 X 4 = 8   |  3 X 4 = 12  |  4 X 4 = 16  |  5 X 4 = 20  |  6 X 4 = 24  |
2 X 5 = 10  |  3 X 5 = 15  |  4 X 5 = 20  |  5 X 5 = 25  |  6 X 5 = 30  |
2 X 6 = 12  |  3 X 6 = 18  |  4 X 6 = 24  |  5 X 6 = 30  |  6 X 6 = 36  |
2 X 7 = 14  |  3 X 7 = 21  |  4 X 7 = 28  |  5 X 7 = 35  |  6 X 7 = 42  |
2 X 8 = 16  |  3 X 8 = 24  |  4 X 8 = 32  |  5 X 8 = 40  |  6 X 8 = 48  |
2 X 9 = 18  |  3 X 9 = 27  |  4 X 9 = 36  |  5 X 9 = 45  |  6 X 9 = 54  |
2 X 10 = 20 |  3 X 10 = 30 |  4 X 10 = 40 |  5 X 10 = 50 |  6 X 10 = 60 |
```

```
# While loop
```

In [34]:
```
# while loop
# x=True
# while x:    # infinite loop
#     print(x)
c =0
while c<7:
    print(c)
    c+=1  # updation (increment/decrement) is importatnt
```

```
0
1
2
3
4
5
6
```

```
Break and Continue Statement
```

In [38]:
```
# break statement terminates the loop if centain condition is fullfilled
per=[10,20,30,40,50,777,60,70]
for i in per:
    if i<100:
        print(i)
    else:
        break
print("End of program")
```

```
10
20
30
40
50
End of program
```

In [40]:
```
# Continue statement carry out(terminates) from the iteration if centain condition
# is fullfilled but loop will be continued
per=[10,20,30,40,50,777,60,70]
for i in per:
    if i<100:
        print(i)
    else:
        continue
print("End of program")
```

```
10
20
30
40
50
60
70
End of program
```

In [43]:
```
per=[10,20,30,40,50,777,60,70]
for i in per:
    if i>100:
        print("wrong per ->",end=" ")
        print(i)
    else:
        print(i)
print("End of program")
```

```
10
20
30
40
50
wrong per -> 777
60
70
End of program
```

In [46]:
```
per=[10,20,30,40,50,777,60,70]
for i in per:
    if i>100:
        print("wrong per")
        continue
    print(i)
print("End of program")
```

```
10
20
30
40
50
wrong per
60
70
End of program
```

In [47]:
```
per=[10,20,30,40,50,777,60,70]
for i in per:
    if i>100:
        print("wrong per")
        break
    print(i)
print("End of program")
```

```
10
20
30
40
50
wrong per
End of program
```

```
Pass Statement
```

In [50]:
```
#pass
i=input("Enetr a value:")
if i<8:

else:
    print("Wrong value")
```

```
  Cell In[50], line 5
    else:
    ^
IndentationError: expected an indented block after 'if' statement on line 3
```

In [55]:
```
#pass
i=int(input("Enetr a value:"))
if i<8:
    pass
else:
    print("Wrong value")
```

```
Enetr a value:5
```

In [56]:
```
#pass
i=int(input("Enetr a value:"))
if i<8:
    pass
else:
    print("Wrong value")
```

```
Enetr a value:88
Wrong value
```

```
### FOR ELSE (it is a construct)
```

In [64]:
```python
# when for loops condition is false then else part is executed

n=int(input("Enter a number:"))
for i in range(0,n-1):
    if n<=0:
        print(i)
    else:
        break
else:
    print("End of program")
```

```
Enter a number:-7
End of program
```

In [ ]:
```python

```

Program of loops

In [12]:
```
# 1) write a program to print first n even natural no.

n=int(input("Enter a no.:"))
count=0
i=1
while count!=n:
    if i%2==0:
        print(i)
        count+=1
    if count==n:
        break
    i+=1
```

```
Enter a no.:12
2
4
6
8
10
12
14
16
18
20
22
24
```

In [4]:
```
# 2) write a program to print first n even natural no. in reverse order

n=int(input("Enter a no.:"))
count=0
i=n*2
while i>1:
    if i%2==0:
        print(i, end="  ")
        count+=1
    i-=1
```

```
Enter a no.:12
24  22  20  18  16  14  12  10  8  6  4  2
```

In [7]:
```
# 3) write a program to print cubes of first n natural no.

n=int(input("Enter a no.:"))
for i in range(1,n+1):
    print("Cube of",i,"=",i**3)
```

```
Enter a no.:15
Cube of 1 = 1
Cube of 2 = 8
Cube of 3 = 27
Cube of 4 = 64
Cube of 5 = 125
Cube of 6 = 216
Cube of 7 = 343
Cube of 8 = 512
Cube of 9 = 729
Cube of 10 = 1000
Cube of 11 = 1331
Cube of 12 = 1728
Cube of 13 = 2197
Cube of 14 = 2744
Cube of 15 = 3375
```

In [23]:
```
# 4) write a program to print only vowels of a given string

s=input("Enter a Text:")
v="aeiou"
for i in s:
        if i in v:
            print(i)
        elif i in v.upper():
            print(i)
```

```
Enter a Text:Nikhil Ik
i
i
I
```

In [3]:
```
# 5) write a program to print unique digits of a given integer
n=input("Enter a No.:")
l=[]
l.append(n[0])
print(l[0], end="  ")
for i in n:
    if i in l:
        continue
    else:
        l.append(i)
    print(i,end="  ")
```

```
Enter a No.:122465124123
1  2  4  6  5  3
```

In [21]:
```
# 6) write a program to print the count of spaces of a given string
s=input("Enter a Text:")
print("' ' is :",s.count(" "))
```

```
Enter a Text:nikhil jhjf dshdie   hvuidh
' ' is : 5
```

Slicing in String

In [1]:
```
l=[1,5,8,"nikhil","pankaj",56]
l[:]
```

Out[1]:
```
[1, 5, 8, 'nikhil', 'pankaj', 56]
```

In [2]:
```
s="we are the students of AIDS Department"
s[7]
```

Out[2]:
```
't'
```

In [3]:
```
s[0:5]  # -> gives 0,1,2,3,4 th idexed value
```

Out[3]:
```
'we ar'
```

In [5]:
```
s[0:]   # -> gives 0 to len(s)th indexed value
```

Out[5]:
```
'we are the students of AIDS Department'
```

In [6]:
```
s[:10]   # -> gives 0 to 10 th indexed value
```

Out[6]:
```
'we are the'
```

In [7]:
```
s[:]  # gives complete string
```

Out[7]:
```
'we are the students of AIDS Department'
```

In [8]:
```
print(s[])  # throws error
```

```
  Cell In[8], line 1
    print(s[])  # throws error
          ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

In [9]:
```
print(s[:])  # gives complete string [0:len(s)]
```

```
we are the students of AIDS Department
```

In [23]:
```
print(s[2:18:2])  # s[starting point:(ending point-1):increment by (1 is default)]
print(s[18:2:-2])
print(s[:])
print(s[::-1])  # reverse the string
print(s[2:18:-2])  # logical error
print(s[18:2])  # logical error
```

```
 r h tdn
sndt h r
we are the students of AIDS Department
tnemtrapeD SDIA fo stneduts eht era ew
```

In [30]:
```
# backward indexing
print(len(s))
print(s[-38:-1])  # -> it gives (n-1)--->(-1-1)=(-2)th index
print(s[-1:-5])  # logical error
print(s[-38:-1:-1])  # logical error
print(s[-38:0:1])  # logical error (0 is not allowed in backward indexing)
```

```
38
we are the students of AIDS Departmen
```

In [20]:
```
print(s[-1:-38:-1]) # reverse order
```

```
tnemtrapeD SDIA fo stneduts eht era e
```

Functions of String

In [31]:
```
print(len(s))
```

```
38
```

In [34]:
```
print("are" in s)  # "are" is a Substring
print("hgjhkj" in s)
```

```
True
False
```

In [38]:
```
print(s.find("are")) # it gives index of substring if present in string else return -
print(s.find("dbhjdf"))
```

```
3
-1
```

In [40]:
```
print(s.index("are")) # it gives index of substring if present in string else throws
print(s.index("dbhjdf"))
```

```
3

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[40], line 2
      1 print(s.index("are")) # it gives index of substring if present in string el
se throws an error
----> 2 print(s.index("dbhjdf"))

ValueError: substring not found
```

In [43]:
```
s = "     Vaishnavi mahira       "
print(s)
print(s.strip())  # strip in both side(remove extra spaces)
print(s.rstrip())  # strip in right side but left side remains same
print(s.lstrip())  # strip in left side but right side remains same
```

```
     Vaishnavi mahira
Vaishnavi mahira
     Vaishnavi mahira
Vaishnavi mahira
```

In [44]:
```
s = "!!! Vaishnavi mahira !!!!!"
print(s)
print(s.strip("!"))  # strip in both side(remove substring)
print(s.rstrip("!"))  # strip in right side but left side remains same
print(s.lstrip("!"))  # strip in left side but right side remains same
```

```
!!! Vaishnavi mahira !!!!!
 Vaishnavi mahira
!!! Vaishnavi mahira
 Vaishnavi mahira !!!!!
```

In [48]:
```
s = "-    Vaishnavi mahira     -"
print(s)
print(s.strip(" "))
print(s.rstrip(" "))  #no change in output
print(s.lstrip(" "))
```

```
-    Vaishnavi mahira     -
-    Vaishnavi mahira     -
-    Vaishnavi mahira     -
-    Vaishnavi mahira     -
```

In [49]:
```
s = "-    Vaishnavi mahira     -"
print(s)
print(s.strip("i"))
print(s.rstrip("i"))  # no change in output
print(s.lstrip("i"))
```

```
-    Vaishnavi mahira     -
-    Vaishnavi mahira     -
-    Vaishnavi mahira     -
-    Vaishnavi mahira     -
```

In [50]:
```
s = "-    Vaishnavi mahira     -"
print(s)
print(s.strip("-"))  # strip in both side(remove substring)
print(s.rstrip("-"))  # strip in right side but left side remains same
print(s.lstrip("-"))  # strip in left side but right side remains same
```

```
-    Vaishnavi mahira     -
    Vaishnavi mahira
-    Vaishnavi mahira
    Vaishnavi mahira     -
```

In [52]:
```
s="hello oo Hello oo Hello oo Hello"
print(s.count("hello"))
print(s.count("Hello"))  # gives the count of substring in base string it is case sen
```

```
1
3
```

In [53]:
```
print(s.replace("Hell","Hoo"))
```

```
hello oo Hooo oo Hooo oo Hooo
```

In [56]:
```
s="hello oo Hello oo Hello oo Hello"
print(s)
print(id(s))
s = s.replace("Hell","Hoo")
print(s)
print(id(s))
```

```
hello oo Hello oo Hello oo Hello
2461963308656
hello oo Hooo oo Hooo oo Hooo
2461969054448
```

In [58]:
```
s="abf52436dfsghdf"
t="dfhdgbhdjThd"
u="dshjvbjh!#@!hvg3165146"
n="123654125"
print(s.isalpha())  # it checks the string is alphabatical or not (a-z)(A-Z)
print(t.isalpha())
print(u.isalpha())
print(n.isalpha())
```

```
False
True
False
False
```

In [59]:
```
s="abf52436dfsghdf"
t="dfhdgbhdjThd"
u="dshjvbjh!#@!hvg3165146"
n="123654125"
print(s.isalnum())  # it checks the string is alpha numerical or not (a-z)(A-Z)(0-9)
print(t.isalnum())
print(u.isalnum())
print(n.isalnum())
```

```
True
True
False
True
```

In [60]:
```
s="abf52436dfsghdf"
t="dfhdgbhdjThd"
u="dshjvbjh!#@!hvg3165146"
n="123654125"
print(s.isnumeric())  # it checks the string is numerical or not (a-z)(A-Z)
print(t.isnumeric())
print(u.isnumeric())
print(n.isnumeric())
```

```
False
False
False
True
```

In [62]:
```
s=" nikhil vishwakrama"
t="       \n\n"
g="            "
u="nikhilvishwakarma"
print(s.isspace())  # it checks the string has only spaces or not
print(t.isspace())
print(t.isspace())
print(u.isspace())
```

```
False
True
True
False
```

In [64]:
```
s="knfbgjdkbn njdfb"
t="HJDFSVB DSVJ  FSN"
u="tFGBJdd DEbdfb"
n="1236541SBNJKNB  fdbg 25"
print(s.isupper())  # it checks the string is in capital letters or not (A-Z)
print(t.isupper())
print(u.isupper())
print(n.isupper())
```

```
False
True
False
False
```

In [65]:
```
s="knfbgjdkbn njdfb"
t="HJDFSVB DSVJ  FSN"
u="tFGBJdd DEbdfb"
n="1236541SBNJKNB  fdbg 25"
print(s.islower())  # it checks the string is in small letters or not (a-z)
print(t.islower())
print(u.islower())
print(n.islower())
```

```
True
False
False
False
```

In [66]:
```
s="nikhil vishwakarma"

print(s.upper())  # it change the string in capital letters (upper case) or not (A-Z)
print(s.lower())
print(s.capitalize())
```

```
NIKHIL VISHWAKARMA
nikhil vishwakarma
Nikhil vishwakarma
```

In [70]:
```
s="nikhil vishwakarma from aids barnch 2022-26 batch"
l=list(s)
print(l)
print(s)
print(s.split()) # it gives the list
print(s.split("-"))
```

```
['n', 'i', 'k', 'h', 'i', 'l', ' ', 'v', 'i', 's', 'h', 'w', 'a', 'k', 'a', 'r',
'm', 'a', ' ', 'f', 'r', 'o', 'm', ' ', 'a', 'i', 'd', 's', ' ', 'b', 'a', 'r',
'n', 'c', 'h', ' ', '2', '0', '2', '2', '-', '2', '6', ' ', 'b', 'a', 't', 'c',
'h']
nikhil vishwakarma from aids barnch 2022-26 batch
['nikhil', 'vishwakarma', 'from', 'aids', 'barnch', '2022-26', 'batch']
['nikhil vishwakarma from aids barnch 2022', '26 batch']
```

In [78]:
```
x=["1","2","3","4","5","6","7"]
s="*".join(x)
t="".join(x)
u="_".join(x)
print(s)
print(type(s))
print(print(u))
print(print(t))
```

```
1*2*3*4*5*6*7
<class 'str'>
1_2_3_4_5_6_7
None
1234567
None
```

In [ ]:
```
```

Programs of String

In [2]:
```
# 1) write a program to check given string has only alphabets
s=input("Enter a text:")
if s.isalpha():
    print("Alphabetical String")
else:
    print("Not a Alphabetical String")
```

```
Enter a text:nikhilVishwakarma
Alphabetical String
```

In [4]:
```
# 2) write a program to check if a given character is present in given string
s=input("Enter a text:")
c=input("Enter a Character:")
if c in s:
    print("Character is Present")
else:
    print("Character is  Not Present")
```

```
Enter a text:nikhil
Enter a Character:i
Character is Present
```

In [6]:
```
# 3) write a program to count vowels present in  a given string
s=input("Enter a text:")
v="aeiou"
c=0
for i in s:
    if i in v:
        c+=1
    elif i in v.upper():
        c+=1
print(c,"vowels present")
```

```
Enter a text:nikhil VISHWAKARMA Aids
8 vowels present
```

In [10]:
```
# 4) write a program to count words in a given string
s=input("Enter a text:")
v=input("Enter a word:")
print("Count of \"",v,"\" is",s.count(v))
```

```
Enter a text:hello hello Hello HE is she he
Enter a word:he
Count of " he " is 4
```

In [1]:
```
# 5) write a program to reverse a string
s = input("Enter a text:")
print(s[-1:-(len(s)+1):-1])
```

```
Enter a text:nikhil vishwakarma
amrakawhsiv lihkin
```

In [8]:
```
# 6) write a program to reverse a string word wise
s = input("Enter a text:")
l=s.split()
for i in l:
    r=i
    print(r[-1:-(len(r)+1):-1],end=" ")
```

```
Enter a text:sandeep nikhil ankit rahul
peednas lihkin tikna luhar
```

In [9]:
```
# 7) write a program to extract no. from a given string and store all the no. 
s = input("Enter a text:")
n="0123456789"
l=[]
for i in s:
    if i in n:
        l.append(i)
        print(i)
```

```
Enter a text:nikhil 67 jrd 2 tata ok 4
6
7
2
4
```

In [13]:
```
# 8) write a program to check whether a string is pllindron or not
s = input("Enter a text:")
if s[-1:-(len(s)+1):-1] == s[0:len(s)+1]:
    print("String is Pallindrom")
else:
    print("Not a Pallindrom")
```

```
Enter a text:nitin
String is Pallindrom
```

In [14]:
```
# 8) write a program to transform a string in uppercase
s = input("Enter a text:")
print(s.upper())
```

```
Enter a text:nikhil vishwakarma
NIKHIL VISHWAKARMA
```

In [15]:
```python
# 9) write a program to find max length words in a given string
s = input("Enter a text:")
l=s.split()
r = "a"
for i in l:
    if len(r)<=len(i):
        r=i
    else:
        continue
print(r)
```

```
Enter a text:nikhil third sem aids department sistec gandhi nagar
department
```

In [ ]:

List and its Functions

In [15]:
```
l=[1,10,2,3,"CSE","AI&DS",["a","b"],(3,7)]
print(l)
print(id(l))
print(type(l))
```

```
[1, 10, 2, 3, 'CSE', 'AI&DS', ['a', 'b'], (3, 7)]
2371603179648
<class 'list'>
```

In [16]:
```
t=(1,2,3,4,5)
ll=list(t)
print(ll)
```

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

In [17]:
```
print(l)
print(l[4])
print(l[6][0])
```

```
[1, 10, 2, 3, 'CSE', 'AI&DS', ['a', 'b'], (3, 7)]
CSE
a
```

In [18]:
```
len(l)
```

Out[18]: 8

In [19]:
```
l.append("nikhil")  # this f( make the list mutable)
print(l)
```

```
[1, 10, 2, 3, 'CSE', 'AI&DS', ['a', 'b'], (3, 7), 'nikhil']
```

In [20]:
```
l.append(t)  # it aad on the element in the last
print(l)
```

```
[1, 10, 2, 3, 'CSE', 'AI&DS', ['a', 'b'], (3, 7), 'nikhil', (1, 2, 3, 4, 5)]
```

In [21]:
```
l.count()  # throws error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[21], line 1
----> 1 l.count()

TypeError: list.count() takes exactly one argument (0 given)
```

In [23]:
```
l.count(1)
```

Out[23]: 1

In [26]:
```
print(l)
print(l.index("nikhil"))  # gives index value if element is present in list else give
print(l.index(8))
```

```
[1, 10, 2, 3, 'CSE', 'AI&DS', ['a', 'b'], (3, 7), 'nikhil', (1, 2, 3, 4, 5)]
8

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[26], line 3
      1 print(l)
      2 print(l.index("nikhil"))  # gives index value if element is present in list
else gives error
----> 3 print(l.index(8))

ValueError: 8 is not in list
```

In [28]:
```
print(l)
l.insert(3,"hi")   # it insert the elment on selsected index (index,value)
print(l)
```

```
[1, 10, 2, 'hi', 3, 'CSE', 'AI&DS', ['a', 'b'], (3, 7), 'nikhil', (1, 2, 3, 4, 5)]
[1, 10, 2, 'hi', 'hi', 3, 'CSE', 'AI&DS', ['a', 'b'], (3, 7), 'nikhil', (1, 2, 3,
4, 5)]
```

In [31]:
```
x=[1,2,3,4,5]
y=["nikhil","pankaj","saurabh"]
x.append(y)  # append complete list as a single element
print(x)
```

```
[1, 2, 3, 4, 5, ['nikhil', 'pankaj', 'saurabh']]
```

In [32]:
```
x=[1,2,3,4,5]
y=["nikhil","pankaj","saurabh"]
x.extend(y)  # it extend list x and add element of list y in list x
print(x)
```

```
[1, 2, 3, 4, 5, 'nikhil', 'pankaj', 'saurabh']
```

In [38]:
```
x=[1,2,3,4,5]
y=("nikhil","pankaj","saurabh")
# 1st object should be list
x.extend(y)  # it extend list x and add element of list y in list x
print(x)
```

```
[1, 2, 3, 4, 5, 'nikhil', 'pankaj', 'saurabh']
```

In [41]:
```
x=[1,2,3,4,5]
y={"nikhil","pankaj","saurabh"}
# 1st object should be list
x.extend(y)  # it extend list x and add element of list y in list x
print(x)
```

```
[1, 2, 3, 4, 5, 'nikhil', 'saurabh', 'pankaj']
```

In [43]:
```
x=["nikhil","nikhil","vivek","geetesh"]
print(x)
x.remove("nikhil")  # removes the element
print(x)
```

```
['nikhil', 'nikhil', 'vivek', 'geetesh']
['nikhil', 'vivek', 'geetesh']
```

In [44]:
```
print(x)
x.remove("saurabh")  # throws an error
print(x)
```

```
['nikhil', 'vivek', 'geetesh']

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[44], line 2
      1 print(x)
----> 2 x.remove("saurabh")  # throws an error
      3 print(x)

ValueError: list.remove(x): x not in list
```

In [45]:
```
print(x)
x.remove("vivek","nikhil")  # only one argument is allowed
print(x)
```

```
['nikhil', 'vivek', 'geetesh']

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[45], line 2
      1 print(x)
----> 2 x.remove("vivek","nikhil")  # only one argument is allowed
      3 print(x)

TypeError: list.remove() takes exactly one argument (2 given)
```

In [49]:
```
x=[1,2,3,4,5,6,7]
x.remove(1)
print(x)
x.pop(3) # it remove the element on index
print(x)
```

```
[2, 3, 4, 5, 6, 7]
[2, 3, 4, 6, 7]
```

In [50]:
```
x=[1,4,88,4,9,1,55,7,8]
print(x)
x.sort()  # accending order is default
print(x)
```

```
[1, 4, 88, 4, 9, 1, 55, 7, 8]
[1, 1, 4, 4, 7, 8, 9, 55, 88]
```

In [51]:
```
print(x)
x.sort(reverse=True)  # it return list in decending order
print(x)
```

```
[1, 1, 4, 4, 7, 8, 9, 55, 88]
[88, 55, 9, 8, 7, 4, 4, 1, 1]
```

In [53]:
```
print(l)
l.sort()  # throws error because it works on homogeneous values(list) but l has hecto
print(l)
```

```
[1, 2, 10, 'hi', 'hi', 3, 'CSE', 'AI&DS', ['a', 'b'], (3, 7), 'nikhil', (1, 2, 3,
4, 5)]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 2
      1 print(l)
----> 2 l.sort()  # throws error because it works on homogeneous values(list) but l
has hectogeneous values
      3 print(l)

TypeError: '<' not supported between instances of 'str' and 'int'
```

Aliasing And Cloning

```
List Aliasing
```

In [56]:
```
x=[1,5,8,2,6,8]
y=x  # in aliasing we have single original copy
print(x)
print(id(x))
print(type(x))
print(y)
print(id(y))
print(type(y))
# y.append(66)
x.append(56)
print(x)
print(id(x))
print(type(x))
print(y)
print(id(y))
print(type(y))
```

```
[1, 5, 8, 2, 6, 8]
2371604298176
<class 'list'>
[1, 5, 8, 2, 6, 8]
2371604298176
<class 'list'>
[1, 5, 8, 2, 6, 8, 56]
2371604298176
<class 'list'>
[1, 5, 8, 2, 6, 8, 56]
2371604298176
<class 'list'>
```

```
# List Cloning
```

In [58]:
```
# Case 1:
x=[1,5,8,3,4,6,55,12]
y=x[:]  # it access the element of x then store it to y
        # in cloning we have multiple copy
print(x)
print(id(x))
print(type(x))
print(y)
print(id(y))
print(type(y))
x.append(56)
y.append(65)
print(x)
print(id(x))
print(type(x))
print(y)
print(id(y))
print(type(y))
```

```
[1, 5, 8, 3, 4, 6, 55, 12]
2371604296832
<class 'list'>
[1, 5, 8, 3, 4, 6, 55, 12]
2371604296128
<class 'list'>
[1, 5, 8, 3, 4, 6, 55, 12, 56]
2371604296832
<class 'list'>
[1, 5, 8, 3, 4, 6, 55, 12, 65]
2371604296128
<class 'list'>
```

In [59]:
```
# Case 2:
x=[1,5,8,3,4,6,55,12]
y=x.copy()
print(x)
print(id(x))
print(type(x))
print(y)
print(id(y))
print(type(y))
x.append(56)
y.append(65)
print(x)
print(id(x))
print(type(x))
print(y)
print(id(y))
print(type(y))
```

```
[1, 5, 8, 3, 4, 6, 55, 12]
2371604290752
<class 'list'>
[1, 5, 8, 3, 4, 6, 55, 12]
2371604338560
<class 'list'>
[1, 5, 8, 3, 4, 6, 55, 12, 56]
2371604290752
<class 'list'>
[1, 5, 8, 3, 4, 6, 55, 12, 65]
2371604338560
<class 'list'>
```

In [60]:
```
# Operators in list
x=[1,2,3,4,5]
y=[5,6,7,8]
print(x+y)
```

```
[1, 2, 3, 4, 5, 5, 6, 7, 8]
```

In [62]:
```
# Operators in list
x=[1,2,3,4,"nikhil"]
y=[5,6,7.8,8]
print(x+y)
print(x*2)
```

```
[1, 2, 3, 4, 'nikhil', 5, 6, 7.8, 8]
[1, 2, 3, 4, 'nikhil', 1, 2, 3, 4, 'nikhil']
```

In [64]:
```
print(x*y) # Throws an error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[64], line 1
----> 1 print(x*y)

TypeError: can't multiply sequence by non-int of type 'list'
```

In [65]:
```
# + Concatanate list
# * repeate a list by multiplied times(int)
```

In [67]:
```
x=["11","22","33","44"]
y=["11","22","33","44"]
print(x==y)
# 1st compairing length of lists
# 2nd compairing datatypes of list items respectively
# 3rd compairing content(value) of list items
```

```
True
```

In [68]:
```
x=["11","22","33","44"]
y=["11","22","33","55"]
print(x==y)
```

```
False
```

In [69]:
```
x=["11","22","33","44"]
y=["11","22","33","55"]
print(x<y)
```

```
True
```

In [70]:
```
x=["11","22","33","44"]
y=["11","22","33","55"]
print(x>y)
```

```
False
```

In [72]:
```
# comparison operators will be applied on respective same datatypes list
x=["11","22","33",44]
y=["11","22","33",44]
print(x==y)
```

```
True
```

In [77]:
```
x=["11","22","33",44]
y=["11","22","33","44"]
print(x==y)
print(x<=y)
```

```
False

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[77], line 4
      2 y=["11","22","33","44"]
      3 print(x==y)
----> 4 print(x<=y)

TypeError: '<=' not supported between instances of 'int' and 'str'
```

In [78]:
```
x=[1,6,7,8]
y=[2,6,8]
print(x==y)
print(x<=y)
```

```
False
True
```

In [80]:
```
l=[[[[1,5,7],[2,4,6]],[[1,8,9],5]]]
l[0][0][0][0]
```

Out[80]:
```
1
```

In [81]:
```
l=[1][0][0][0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[81], line 1
----> 1 l=[1][0][0][0]

TypeError: 'int' object is not subscriptable
```

In [82]:
```
# write a program to add elements in a list
l=[]
for i in range(1,10):
    l.append(i)
print(l)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

List Comprehension

In [83]:
```
x=[x for x in range(1,10)]
print(x)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [84]:
```
x=[x for x in range(1,10) if x%2==0]
print(x)
```

```
[2, 4, 6, 8]
```

In [85]:
```
x=[i for i in range(1,10) if i%2>0]
print(x)
```

```
[1, 3, 5, 7, 9]
```

In [ ]:
```

```

Programs of List

In [89]:
```
# 1) write a program to calculate sum of elements of the list
n=input("Enter no. for sum by giving space:")
n=n.split()
l=[int(i) for i in n]
print(sum(l))
```

```
Enter no. for sum by giving space:1 2 3 5 4 6 9 8 7
45
```

In [15]:
```
# 2) write a program to create a list of squares of no. of a given list
n=int(input("Enter a no.:"))
l=[i*i for i in range(1,n+1)]
print(l)
```

```
Enter a no.:25
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361,
400, 441, 484, 529, 576, 625]
```

In [12]:
```
# 3) write a program to sort list in decending order
n=input("Enter no. by giving space:")
n=n.split()
l=[int(i) for i in n]
l.sort(reverse=True)
print(l)
```

```
Enter no. by giving space:10 36 54 875 12 45 12 102 45 12 41 2 36 9 6
[875, 102, 54, 45, 45, 41, 36, 36, 12, 12, 12, 10, 9, 6, 2]
```

In [21]:
```
# 4) write a program to create a list of 1st n prime no.
n=int(input("Enter a no.:"))
l=[]
i=2
while len(l)<n:
    c=0
    for j in range(1,i+1):
        if i%j==0:
            c+=1
    if c==2:
        l.append(i)
    i+=1
print(l)
```

```
Enter a no.:100
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 7
9, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 16
7, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257,
263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 35
9, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457,
461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541]
```

In [13]:
```
# 5) write a program to create two list from a given list of no. in such a way that t
# list can have  non +ve no.

n=input("Enter Integers by giving space:")
n=n.split()
ln=[int(i) for i in n if int(i)<=0]
lp=[int(i) for i in n if int(i)>0]
print("Psitive integers:",lp)
print("Psitive integers:",ln)
```

```
Enter Integers by giving space:25 -5 0 36 -6 45 -9 5 69 -12 9 26 -25
Psitive integers: [25, 36, 45, 5, 69, 9, 26]
Psitive integers: [-5, 0, -6, -9, -12, -25]
```

Tupple and Its Functions

In [8]:
```
t1=()  # empty tupple
print(t1)
t2=(1)
print(t2)
t=(1,2,5,8,4,6,"nikhil","vivek")
print(t)
print(id(t))
print(type(t))
```

```
()
1
(1, 2, 5, 8, 4, 6, 'nikhil', 'vivek')
2783256325792
<class 'tuple'>
```

In [3]:
```
t[6][1]
```

Out[3]:
```
'i'
```

In [4]:
```
len(t)
```

Out[4]:
```
8
```

In [11]:
```
tup=(1,7,3,1,2,2,2,2,2,6,"nikhil")
tup.count(2)
```

Out[11]:
```
5
```

In [12]:
```
tup.index(2)  # it returns from first index where the element was found
```

Out[12]:
```
4
```

In [14]:
```
t1=(2,4,6)
t2=(5,66)
print(t1+t2)
print(t2+t1)
```

```
(2, 4, 6, 5, 66)
(5, 66, 2, 4, 6)
```

In [16]:
```
t1=(2,4,6)
t2=(5,66)
print(t2*2)
print(t1*t2)   # throws an error
```

```
(5, 66, 5, 66)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 4
      2 t2=(5,66)
      3 print(t2*2)
----> 4 print(t1*t2)

TypeError: can't multiply sequence by non-int of type 'tuple'
```

Tupple Packing and Unpacking

In [17]:
```
t=1,2,3,4,5,"vivek",75.6
print(t)
print(type(t))
```

```
(1, 2, 3, 4, 5, 'vivek', 75.6)
<class 'tuple'>
```

In [20]:
```
t=1,2,3,4,5,"n"  # tupple packing
a,b,c,d,e,f=t           # tupple unpacking
print(t)
print(type(t))
print(a,b,c,d,e,f)
print(type(a))
print(type(f))
```

```
(1, 2, 3, 4, 5, 'n')
<class 'tuple'>
1 2 3 4 5 n
<class 'int'>
<class 'str'>
```

In [22]:
```
# tupple comprehension is not possible in tupple
x=(i for i in range(1,10)) # do not generate tupple
print(x)
print(type(x))
```

```
<generator object <genexpr> at 0x00000288057F98A0>
<class 'generator'>
```

In [ ]:
```

```

Set and its Functions

In [23]:
```
# Empty set is not allowed in python
s={}
print(s)
print(type(s))
```

```
{}
<class 'dict'>
```

In [26]:
```
s={1,23,54,6,15, "nikhil",56,1,0,25}
print(s)
print(type(s))
```

```
{0, 1, 6, 15, 54, 23, 'nikhil', 25, 56}
<class 'set'>
```

In [27]:
```
p={1,2,3,4,6}
q={4,9,1,0,5}
print(p.intersection(q))  # common variables
print(p.union(q))
print(p.difference(q))
```

```
{1, 4}
{0, 1, 2, 3, 4, 5, 6, 9}
{2, 3, 6}
```

In [28]:
```
p.add("sistec")
print(p)
```

```
{1, 2, 3, 4, 6, 'sistec'}
```

Set Comprehension

In [29]:
```
x={i for i in range(1,11) if i%2==0}
print(x)
```

```
{2, 4, 6, 8, 10}
```

Dictionary and its Functions

In [31]:
```
d={"key":"value"}
print(d)
print(id(d))
print(type(d))
```

```
{'key': 'value'}
2783270690496
<class 'dict'>
```

In [45]:
```
d={1:"nikhil",2:"sandeep",3:"ankit",4:"Rahul"}
print(d)
print(id(d))
print(type(d))
```

```
{1: 'nikhil', 2: 'sandeep', 3: 'ankit', 4: 'Rahul'}
2783270691968
<class 'dict'>
```

In [46]:
```
print(d[0])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[46], line 1
----> 1 print(d[0])

KeyError: 0
```

In [47]:
```
print(d[1])  # key is the index
```

```
nikhil
```

In [48]:
```
d.keys()
print(d)
print(d.keys())
print(type(d.keys()))
```

```
{1: 'nikhil', 2: 'sandeep', 3: 'ankit', 4: 'Rahul'}
dict_keys([1, 2, 3, 4])
<class 'dict_keys'>
```

In [49]:
```
d.values()
```

Out[49]:
```
dict_values(['nikhil', 'sandeep', 'ankit', 'Rahul'])
```

In [50]:
```
d.get(2)
```

Out[50]:
```
'sandeep'
```

In [51]:
```
print(d.items())
```

```
dict_items([(1, 'nikhil'), (2, 'sandeep'), (3, 'ankit'), (4, 'Rahul')])
```

In [52]:
```
d.pop(2)  # delete the the value of that key
```

Out[52]:
```
'sandeep'
```

In [53]:
```
d.pop("a")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[53], line 1
----> 1 d.pop("a")

KeyError: 'a'
```

In [54]:
```
d.clear()  # clear all the elements of dictionary and return empty dictionary
```

In [55]:
```
print(d)
print(type(d))
```

```
{}
<class 'dict'>
```

In [57]:
```
l=["nikhil","monu","sonu"]
d=dict(l)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[57], line 2
      1 l=["nikhil","monu","sonu"]
----> 2 d=dict(l)

ValueError: dictionary update sequence element #0 has length 6; 2 is required
```

In [58]:
```
d1={1:{"n":"nikhil"},2:{"b":"bittu"}}  # nesting of dictionary
print(d1)
```

```
{1: {'n': 'nikhil'}, 2: {'b': 'bittu'}}
```

In [ ]:

In [ ]:

Programs of Tuple, Set & Dictionary

In [2]:
```
# 1) write a program to create a tuple from a given list & reverse it

l=[5,4,7,8,7]
t=tuple(l)
print(l," -->",t)
print("In reverse order:",t[::-1])
```

```
[5, 4, 7, 8, 7]  --> (5, 4, 7, 8, 7)
In reverse order: (7, 8, 7, 4, 5)
```

In [11]:
```
# 2) write a program to create list of tuples where each tuple is a pair of elements
# 1st element is uppercase Character & second element is its unicode
l=[]
a=ord("A")
z=ord("Z")
for i in range(a,z+1):
    t=(chr(i), i)
    l.append(t)
print(l)
```

```
[('A', 65), ('B', 66), ('C', 67), ('D', 68), ('E', 69), ('F', 70), ('G', 71), ('H',
72), ('I', 73), ('J', 74), ('K', 75), ('L', 76), ('M', 77), ('N', 78), ('O', 79),
('P', 80), ('Q', 81), ('R', 82), ('S', 83), ('T', 84), ('U', 85), ('V', 86), ('W',
87), ('X', 88), ('Y', 89), ('Z', 90)]
```

In [16]:
```
# 3) create two sets from a given set of no. to separate even and odd no.
s=input("Enter Elements of set by giving space:")
s=s.split()
s_base={int(i) for i in s}
s_even={int(i) for i in s if int(i)%2==0}
s_odd={int(i) for i in s if int(i)%2>0}
print("given Set Elements:",s_base)
print("Set of Even Elements:",s_even)
print("Set of odd Elements:",s_odd)
```

```
Enter Elements of set by giving space:1 25 4 6 2 1 2 3 6 4 5 7 8 9 6 5 4  2 6 5 5 5
5 6
given Set Elements: {1, 2, 3, 4, 5, 6, 7, 8, 9, 25}
Set of Even Elements: {8, 2, 4, 6}
Set of odd Elements: {1, 3, 5, 7, 9, 25}
```

In [1]:
```
# 4) sort a dictionary by its keys in decending order
d={1:"Aakash",2:"Aliya",10:"Ankush",3:"Ashutosh",5:"Abhishek",
   7:"Aman",9:"Ankit",4:"Abhay",11:"Arslaan",6:"Afiya",8:"Animesh"}
print(d)
# print(d.keys())
d=d.items()
d=list(d)
d.sort(reverse=True)
# print(d)
d=dict(d)
print(d)
```

```
{1: 'Aakash', 2: 'Aliya', 10: 'Ankush', 3: 'Ashutosh', 5: 'Abhishek', 7: 'Aman', 9:
'Ankit', 4: 'Abhay', 11: 'Arslaan', 6: 'Afiya', 8: 'Animesh'}
{11: 'Arslaan', 10: 'Ankush', 9: 'Ankit', 8: 'Animesh', 7: 'Aman', 6: 'Afiya', 5:
'Abhishek', 4: 'Abhay', 3: 'Ashutosh', 2: 'Aliya', 1: 'Aakash'}
```

In [54]:
```
# 5) write a program to find maximum size batch code from a dictionary where key
# values in the dictionary are batch codes and data values are size of the batch

s= input("Enter Batch codes for Dictionary by giving space: ")
s=s.split()
l=[i for i in s]
d=[(i,len(i)) for i in l]
d=dict(d)
print(d)
m=d.values()
mx=max(list(m))
i=list(m).index(mx)
print(mx,i)
m=list(d.keys())
mx=m[i]
print("Max size Batch code: '",mx,"'\t\tSize:",d.get(mx))
```

```
Enter Batch codes for Dictionary by giving space: nikhil_32 ankush_10 sandeep_52 pa
nkaj_36 nikhlesh_34 suhani_35
{'nikhil_32': 9, 'ankush_10': 9, 'sandeep_52': 10, 'pankaj_36': 9, 'nikhlesh_34': 1
1, 'suhani_35': 9}
11 4
Max size Batch code: ' nikhlesh_34 '            Size: 11
```

In [ ]:

INPUT

In [3]:
```
i=input("Enter number: ")
print(i)
print(type(i))
```

```
Enter number: 45
45
<class 'str'>
```

In [4]:
```
i=input("Enter number: ")
print(i)
print(type(i))
j=int(i)
print(j)
print(type(j))
```

```
Enter number: 45
45
<class 'str'>
45
<class 'int'>
```

In [5]:
```
x=input("Enter number: ")
y=input("Enter number: ")
z=input("Enter number: ")
```

```
Enter number: 5
Enter number: 54
Enter number: 1
```

In [6]:
```
print("Enter 3 no.:")
for i in range(0,3):
    input()
```

```
Enter 3 no.:
5
6
8
```

In [7]:
```
x=input("Enter the no.:").split()
print(x)
print(type(x))
```

```
Enter the no.:5 4 8 9 6 5 2 3 6 4
['5', '4', '8', '9', '6', '5', '2', '3', '6', '4']
<class 'list'>
```

In [9]:
```
x=int(input("Enter two no.:").split())
print(x)
print(type(x))
```

```
Enter two no.:88 2

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 x=int(input("Enter two no.:").split())
      2 print(x)
      3 print(type(x))

TypeError: int() argument must be a string, a bytes-like object or a real number, n
ot 'list'
```

In [11]:
```
x=int(input("Enter two no.:"))
print(x)
print(type(x))
```

```
Enter two no.:5 6

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[11], line 1
----> 1 x=int(input("Enter two no.:"))
      2 print(x)
      3 print(type(x))

ValueError: invalid literal for int() with base 10: '5 6'
```

In [10]:
```
x,y=input("Enter two no.:").split()
print(int(x),int(y))
print(type(x),type(y))
```

```
Enter two no.:54 65
54 65
<class 'str'> <class 'str'>
```

In [13]:
```
x,y=[int(i) for i in input("Enter two no.:").split()]
print(x,y)
print(type(x),type(y))
print(x+y)
```

```
Enter two no.:5 6
5 6
<class 'int'> <class 'int'>
11
```

In [14]:
```
x=[int(i) for i in input("Enter numbers with spaces:").split()]
print(x,y)
print(type(x))
print(sum(x))
```

```
Enter numbers with spaces:5 6 3 2 1 4 7 8 9 5
[5, 6, 3, 2, 1, 4, 7, 8, 9, 5] 6
<class 'list'>
50
```

In [16]:
```
# Print average of a list by taking input
x=[int(i) for i in input("Enter numbers with spaces:").split()]
print(x)
print(type(x))
print(sum(x)/len(x))
```

```
Enter numbers with spaces:5 6 7 8 9 10 11 12
[5, 6, 7, 8, 9, 10, 11, 12]
<class 'list'>
8.5
```

In [17]:
```
x=input("Enter text:")
print(x)
print(x.split())
```

```
Enter text:nikhil aids third sem 2023
nikhil aids third sem 2023
['nikhil', 'aids', 'third', 'sem', '2023']
```

OUTPUT

In [21]:
```
# formating of string
name=input("enter name:")
age=int(input("enter age:"))
weight=float(input("enter weight:"))
print(" This is name: {}\n This is age: {} \n This is weight: {}".format(name,age,wei
# it put down the values in {} orderly
# format is f() it called by . operator
```

```
enter name:Nikhil
enter age:18
enter weight:49.26
 This is name: Nikhil
 This is age: 18
 This is weight: 49.26
```

In [35]:
```
name="Nikhil"
age=18
weight=49.26
print(" This is name: {}\n This is age: {} \n This is weight: {}".format(name,age,wei
print("##########")
print(" This is name: %s\n This is age: %d \n This is weight: %f"%(name,age,weight))
# %i --> int
# %d --> decimal
# %hx --> hexa decimal
# %o --> octal
# %s --> string
# %f --> float
```

```
 This is name: Nikhil
 This is age: 18
 This is weight: 49.26
##########
 This is name: Nikhil
 This is age: 18
 This is weight: 49.260000
```

In [39]:
```
x=888888888.264
print("This is float value: {}".format(x))
print("This is float value: {:f}".format(x))
print("This is float value: {:8.3f}".format(x))
print("This is float value: {:12.9f}".format(x))
print("This is float value: {:2.15f}".format(x))
```

```
This is float value: 888888888.264
This is float value: 888888888.264000
This is float value: 888888888.264
This is float value: 888888888.264000058
This is float value: 888888888.264000058174133
```

In [40]:
```
x=[54,69,42]
print("This is value: {}".format(x))
```

```
This is value: [54, 69, 42]
```

In [41]:
```
y={1:"Nikhil",2:"Pankaj",3:"Ankush"}
print("This is value: {}".format(y))
```

```
This is value: {1: 'Nikhil', 2: 'Pankaj', 3: 'Ankush'}
```

In [43]:
```
# F-string
name="Nikhil"
age=18
weight=49.26
print(f" This is name: {name}\n This is age: {age} \n This is weight: {weight}")
```

```
 This is name: Nikhil
 This is age: 18
 This is weight: 49.26
```

In [44]:
```
print(" This is name: {name}\n This is age: {age} \n This is weight: {weight}") # no
```

```
 This is name: {name}
 This is age: {age}
 This is weight: {weight}
```

In [ ]:
```

```

Function

In [45]:
```
# 1st F() Definition

# def say_Hi(arguments):    # def is a keword to define f()
                                    # -> name of f() like a identifier
                                            # -> Argument is optional
#     return value          # return is a keyword which is optional it gives a value

# f() Calling

def function1():
    print("My 1st Function")   # f() doesn't execute on definition time
```

In [46]:
```
function1()   # function calling
```

```
My 1st Function
```

In [47]:
```
def add():
    a=int(input("enter 1st no.:"))
    b=int(input("enter 2nd no.:"))
    print("sum is:",a+b)

add()
```

```
enter 1st no.:12
enter 2nd no.:45
sum is: 57
```

In [50]:
```
def add(a,b):
    print("sum is:",a+b)

add()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[50], line 4
      1 def add(a,b):
      2     print("sum is:",a+b)
----> 4 add()

TypeError: add() missing 2 required positional arguments: 'a' and 'b'
```

In [51]:
```
def add(a,b):
    print("sum is:",a+b)

add(4,9)
```

```
sum is: 13
```

In [52]:
```
def add(a,b):
    print("sum is:",a+b)

a=int(input("enter 1st no.:"))
b=int(input("enter 2nd no.:"))
add(a,b)
```

```
enter 1st no.:5
enter 2nd no.:6
sum is: 11
```

In [54]:
```
def add(a,b):
    print("sum is:",a+b)  # print sum of previous values of a & b

    a=int(input("enter 1st no.:"))
    b=int(input("enter 2nd no.:"))
    print("inside f()")
print("outside f()")
add(a,b)
```

```
outside f()
sum is: 11
enter 1st no.:2
enter 2nd no.:4
inside f()
```

In [56]:
```
def add(a,b):
    print("sum is:",a+b)

    a=int(input("enter 1st no.:"))
    b=int(input("enter 2nd no.:"))
    print("inside f()")
print("outside f()")
add(m,n)  # throws error
```

```
outside f()

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[56], line 8
      6     print("inside f()")
      7 print("outside f()")
----> 8 add(m,n)

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

In [58]:
```
def add(a,b):
    print("sum is:",a+b)

    a=int(input("enter 1st no.:"))
    b=int(input("enter 2nd no.:"))
    print("inside f()")
print("outside f()")
m=9
n=8
add(m,n)
```

```
outside f()
sum is: 17
enter 1st no.:4
enter 2nd no.:7
inside f()
```

In [61]:
```
def add(a,b):
    print("inside f()")
    return a+b

print("outside f()")
a=int(input("enter 1st no.:"))
b=int(input("enter 2nd no.:"))
x=add(a,b)
print("x :",x)
print("add(a,b) :",add(a,b))
```

```
outside f()
enter 1st no.:12
enter 2nd no.:13
inside f()
x : 25
inside f()
add(a,b) : 25
```

In [62]:
```
def square(n):
    return n**2
s=square(7)
print(s)
```

```
49
```

In [ ]:

Programs of Input, Output and Functions

In [2]:
```
# 1) write a program to take two input from the user calculate their product and
# display the output
x,y=[int(i) for i in input("Enter two no. by giving space: ").split()]
p=x*y
print(f"Product of {x} and {y} : {p}")
```

```
Enter two no. by giving space: 5 6
Product of 5 and 6 : 30
```

In [4]:
```
# 2) write a program to take input from the user. The input will be a string
# containing characters and you have to print the unicode corresponding to the charac
s=input("enter your string: ")
for i in s:
    print(f"{ord(i)}",end=" ")
```

```
enter your string: nikhil vishwakarma
110 105 107 104 105 108 32 118 105 115 104 119 97 107 97 114 109 97
```

In [7]:
```
# 3) write a python f() to calculate area of circle
def area_Circle(r):
    return 3.14*(r**2)
r=int(input("Enter radius of Circle: "))
print(f"Area of circle: {area_Circle(r)}")
```

```
Enter radius of Circle: 7
Area of circle: 153.86
```

In [9]:
```
# 4) write a python f() to calculate Compound Interest
def ci(p,r,t):
    s=p*((1+(r/100))**n)
    return s-p
p=float(input("Enter Principal Amount: "))
r=float(input("Enter Rate of Interest: "))
n=int(input("Enter Time Period: "))
print(f"Area of circle: {ci(p,r,n)}")
```

```
Enter Principal Amount: 1000
Enter Rate of Interest: 12
Enter Time Period: 2
Area of circle: 254.4000000000001
```

In [14]:
```
# 5) write a python f() to calculate average of n no.
def average_N(n):
    k=0
    for i in range(n+1):
        k+=i
    return k/n
n=int(input("Enter a no.: "))
print(f"Average of {n} no.: {average_N(n)}")
```

```
Enter a no.: 15
Average of 15 no.: 8.0
```

In [5]:
```python
# 6) write a python f()to calculate volume of cuboid
def vol_Cube(a,b,c):
    return a*b*c

l,b,h=[int(i) for i in input("Enter side of Cuboid by giving space: ").split()]
print(f"volume of cube: {vol_Cube(l,b,h)}")
```

```
Enter side of Cuboid by giving space: 10 12 12
volume of cube: 1440
```

In [ ]:

Programs on Function

In [14]:
```
# 1) write a python f() to calculate volume of cuboid using Taking nothing, Giving nothing concept
def vol_cuboid():
    l,b,h=[int(i) for i in input("Enter length,breadth,height of cuboid by giving space").split()]
    print(f"Volume of Cuboid is : {l*b*h} unit cube")
vol_cuboid()
```

```
Enter length,breadth,height of cuboid by giving space10 12 15
Volume of Cuboid is : 1800 unit cube
```

In [15]:
```
# 2) write a python f() to calculate volume of cuboid using Taking something, Giving nothing concep
def vol_cuboid(l,b,h):
    print(f"Volume of Cuboid is : {l*b*h} unit cube")
l,b,h=[int(i) for i in input("Enter length,breadth,height of cuboid by giving space").split()]
vol_cuboid(l,b,h)
```

```
Enter length,breadth,height of cuboid by giving space14 15 12
Volume of Cuboid is : 2520 unit cube
```

In [16]:
```
# 3) write a python f() to calculate volume of cuboid using Taking something, Giving something conc
def vol_cuboid(l,b,h):
    return l*b*h
l,b,h=[int(i) for i in input("Enter length,breadth,height of cuboid by giving space").split()]
print(f"Volume of Cuboid is : {vol_cuboid(l,b,h)} unit cube")
```

```
Enter length,breadth,height of cuboid by giving space12 45 23
Volume of Cuboid is : 12420 unit cube
```

In [17]:
```
# 4) write a python f() to calculate volume of cuboid using Taking nothing, Giving something concep
def vol_cuboid():
    l,b,h=[int(i) for i in input("Enter length,breadth,height of cuboid by giving space").split()]
    return l*b*h
print(f"Volume of Cuboid is : {vol_cuboid()} unit cube")
```

```
Enter length,breadth,height of cuboid by giving space25 46 12
Volume of Cuboid is : 13800 unit cube
```

In [ ]:
```
```

Function -: Input --- Calling --- Output

In [2]:
```
# def h(formal parameters):
#     return x
# h(actual parameters)


# Taking nothing, Giving nothing

def hi():
    print("hi")

# Taking something, Giving nothing

def hello(name):
    print(f"Hello {name}!")

# Taking something, Giving something

def square(n):
    return n*n

# Taking nothing, Giving something

def vowels():
    return "aeiou"
```

In [3]:
```
hi()
hello("Nikhil")
s=square(5)
print(s)
v=vowels()
print(f"{v} : {list(v)}")
```

```
hi
Hello Nikhil!
25
aeiou : ['a', 'e', 'i', 'o', 'u']
```

In [4]:
```
def add(a,b,c):
    return a+b+c
x=add(5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 3
      1 def add(a,b,c):
      2     return a+b+c
----> 3 x=add(5)

TypeError: add() missing 2 required positional arguments: 'b' and 'c'
```

In [5]:
```
x=add(1,5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 x=add(1,5)

TypeError: add() missing 1 required positional argument: 'c'
```

In [6]:
```
x=add(1,2,5)
print(x)
```

```
8
```

Types of Argument

--: Default Arguments

In [7]:
```
# default argument in f()
def jodo(a=1,b=2,c=3):
    return a+b+c
jodo()
```

Out[7]:
```
6
```

In [8]:
```
jodo(10)  # it takes value of a = 10
```

Out[8]:
```
15
```

In [9]:
```
def jodo_1(a,b=2,c=3):
    return a+b+c
jodo_1(6)
```

Out[9]:
```
11
```

In [10]:
```
jodo_1()  # throws error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 jodo_1()  # throws error

TypeError: jodo_1() missing 1 required positional argument: 'a'
```

In [12]:
```
def jodo_2(a=1,b=2,c): # Not allowed in python bcz non default argument is given firs
    return a+b+c       #  on giving the default argument
jodo_2(6,7,8)
jodo_2(6)
```

```
  Cell In[12], line 1
    def jodo_2(a=1,b=2,c): # Not allowed in python
                           ^
SyntaxError: non-default argument follows default argument
```

In [29]:
```
# case 1:
def are_of_Circle(r=1,p=3.14):
    return r*r*p
are_of_Circle(7)
```

Out[29]:
```
153.86
```

In [26]:
```
# case 2:
def area_of_Circle(r,p=3.14):
    return r*r*p
area_of_Circle(7)
```

Out[26]:
```
153.86
```

In [28]:
```
# case 3:
def area_of_Circle(p=3.14,r):  # wrong practice
    return r*r*p
area_of_Circle(7)
```

```
  Cell In[28], line 2
    def area_of_Circle(p=3.14,r):  # wrong practice
                                   ^
SyntaxError: non-default argument follows default argument
```

Positional Argument Vs Keyword Argument

In [30]:
```
# default arguments are positional argument
```

In [31]:
```
# positional Argument

def rectangle(l,b):
    print(f"l: {l}\t b: {b}")
    return l*b

ar=rectangle(5,6)
print(ar)
```

```
l: 5     b: 6
30
```

In [32]:
```
# positional Argument
# case: 1
def rectangle(l,b):    # l & b are keywords
    print(f"l: {l}\t b: {b}")
    return l*b

ar=rectangle(b=5,l=6)
print(ar)
```

```
l: 6     b: 5
30
```

In [33]:
```
# positional Argument
# case: 1
def rectangle(l,b):    # l & b are keywords
    print(f"l: {l}\t b: {b}")
    return l*b

ar=rectangle(5,b=6)  # it is allowed
print(ar)
```

```
l: 5     b: 6
30
```

In [35]:
```
# positional Argument
# case: 2
def rectangle(l,b):    # l & b are keywords
    print(f"l: {l}\t b: {b}")
    return l*b

ar=rectangle(l=5,6)  # it is not allowed
print(ar)
```

```
  Cell In[35], line 7
    ar=rectangle(l=5,6)  # it is not allowed
                     ^
SyntaxError: positional argument follows keyword argument
```

In [37]:
```
# positional Argument
# case: 3
def rectangle(l,b):    # l & b are keywords
    print(f"l: {l}\t b: {b}")
    return l*b

ar=rectangle(5,l=6)  # throws error
print(ar)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[37], line 7
      4     print(f"l: {l}\t b: {b}")
      5     return l*b
----> 7 ar=rectangle(5,l=6)  # throws error
      8 print(ar)

TypeError: rectangle() got multiple values for argument 'l'
```

Variable Length Arguments

In [39]:
```
# Case: 1
def display(*t):   # it takes arguments as tupple & no. of arguments are not defined
    print(t)

display(6,5)
display(4,7,9)
display(4,7,9,6)
```

```
(6, 5)
(4, 7, 9)
(4, 7, 9, 6)
```

In [14]:
```
# average of given arguments
def average(*t):
    s=0
    t=list(t)
    for i in t[0]:
        s+= i
    avg=s/len(t)
    print(f"sum of {t}: {avg}")

s=[int(i) for i in input("Enter no. for sum by giving space:").split()]
s=tuple(s)
print(s)
average(s)
```

```
Enter no. for sum by giving space:1 5 4 2 3 6 9 8 7
(1, 5, 4, 2, 3, 6, 9, 8, 7)
sum of [(1, 5, 4, 2, 3, 6, 9, 8, 7)]: 45.0
```

Keyword Variable Length Argument

In [5]:
```
# Case: 2
def display(**d):   # it takes arguments as dictionary & "kwarg" is general name whic
    print(d)
    print(type(d))

display(a=6,b=9,c=5)  #  keyword is used to give values
```

```
{'a': 6, 'b': 9, 'c': 5}
<class 'dict'>
```

In [ ]:
```

```

Programs of Function

In [10]:
```
# 1) write a python f() to calculate LCM of two no.
def lcm(a,b):
    i=2
    lcm=max(a,b)
    while (lcm%a==0 and lcm%b==0)<1:
        x=a
        if x>b:
            x=b
        if x%a==0 and x%b==0:
            lcm=x
        elif (x*i)%a==0 and (x*i)%b==0:
            lcm=x*i
        else:
            i+=1
    return lcm
a,b=[int(i) for i in input("Enter two no. by giving space: ").split()]
lcm=lcm(a,b)
print(lcm)
```

```
Enter two no. by giving space: 10 12
60
```

In [23]:
```
# 2) write a python f() to count words in a string
def word_Count(s):
    s=s.split()
    c=len(s)
    return c

x=input("Enter text for counting of words: ")
print(f"Words in your text: {word_Count(x)}")
```

```
Enter text for counting of words: nikhil bittu pankaj lucky hai
Words in your text: 5
```

In [13]:
```
# 3) write a python f() to find no. in a given text, store no. in a list
# & return list
def no_in_Text(s):
    n="0123456789"
    l=[]
    for i in s:
        if i in n:
            l.append(i)
    return l
s=input("Enter a text containing no.: ")
print(f"Your text contains {no_in_Text(s)}")
```

```
Enter a text containing no.: nikhil 1032 vishwakarm0 ank0s4 oksb1
Your text contains ['1', '0', '3', '2', '0', '0', '4', '1']
```

In [31]:
```
# 4) write a python f() which recieves variable length arguments to find
# greatest element it will return the greatest element

def greatest_Element(*t):
    g=max(t[0])
    return g

s=[int(i) for i in input("Enter no. by giving space:").split()]
x=greatest_Element(s)
print(f"Greatest no. : {x}")
```

```
Enter no. by giving space:4 5 2 6 9 78 45 123 698 5 4 4 12 36 45
Greatest no. : 698
```

In [1]:
```
# 5) write a python f() which takes variable length arguments to recieve strings
# return the list of maximum length string or strings if multiple strings have the sa
def max_len_Str(*s):
    k = [len(i) for i in s[0]]
    m=max(k)
    k.clear()
    l=[]
    for i in s[0]:
        if len(i)==m:
            l.append(i)
    return l

s=input("Enter your text: ").split()
x=max_len_Str(s)
print(f"Maximum Length String is : {x}")
```

```
Enter your text: Nikhil Ankit Ankush Krish Balram Pankaj Shyam Tanish Rahul Vishal
Maximum Length String is : ['Nikhil', 'Ankush', 'Balram', 'Pankaj', 'Tanish', 'Vish
al']
```

In [3]:
```
def func(*t):
    print(t)

l=[1,2,3,4]
func(*l)
```

```
(1, 2, 3, 4)
```

In [ ]:

Local & Global Variable

In [1]:
```
# global
a=100
def f1():
    print("Function 1:",a)
def f2():
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 100
Function 3: 100
```

In [4]:
```
# Local
a=100
def f1():
    print("Function 1:",a)
def f2():
    a=666
    print("Function 2:",a)
def f3():
    a=555
    print("Function 3:",a)

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 666
Function 3: 555
```

In [1]:
```
def f2():
    a=768
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f2()
f3()
```

```
Function 2: 768

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 8
      5     print("Function 3:",a)
      7 f2()
----> 8 f3()

Cell In[1], line 5, in f3()
      4 def f3():
----> 5     print("Function 3:",a)

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

In [2]:
```
a=99  # global variable
def f2():
    a=768   # f() call is gives the priority to local variable
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f2()
f3()
```

```
Function 2: 768
Function 3: 99
```

In [4]:
```
a=99  # global variable
def f2():
    a=768
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

a=200  # updating global variable --> gglobally

f2()
f3()
```

```
Function 2: 768
Function 3: 200
```

In [5]:
```
# Global - Local variable concept works when the both variable has same identifier
```

In [2]:
```
def f2():
    a=768   # f() call is gives the priority to local variable
    print("Function 2:",a)
def f3():
    print("Function 3:",a)
a=200    # global variable
f2()
f3()
```

```
Function 2: 768
Function 3: 200
```

In [1]:
```
def f1():
    a=100
    print("Function 1:",a)
def f2():
    a=987
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f1()
f2()
f3()  # throws error
```

```
Function 1: 100
Function 2: 987

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 12
     10 f1()
     11 f2()
---> 12 f3()

Cell In[1], line 8, in f3()
      7 def f3():
----> 8     print("Function 3:",a)

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

In [2]:
```
# global variable inside f()
def f1():
    global a
    a=100
    print("Function 1:",a)
def f2():
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 100
Function 3: 100
```

In [3]:
```
# global and local variable inside f()
def f1():
    global a
    a=100
    print("Function 1:",a)
def f2():
    a=125
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 125
Function 3: 100
```

In [5]:
```
# Accesing global variable in presence of local variable inside f()
def f1():
    global a
    a=100
    print("Function 1:",a)
def f2():
    a=125
    print("Function 2:",a)
def f3():
    a=659
    print("Function 3 - Local:",a)
    print("Function 3 - Global:",globals()["a"])  # printing global variable

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 125
Function 3 - Local: 659
Function 3 - Global: 100
```

In [7]:
```
# updating global variable inside f() in presence of local variable
def f1():
    global a
    a=100
    print("Function 1:",a)
def f2():
    a=125
    print("Function 2:",a)
def f3():
    a=659
    print("Function 3 - Local:",a)
    print("Function 3 - Global:",globals()["a"])
    a=333
    print("Function 3 - Global:",globals()["a"])
    globals()["a"]=414
    print("Function 3 - Global:",globals()["a"])


f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 125
Function 3 - Local: 659
Function 3 - Global: 100
Function 3 - Global: 100
Function 3 - Global: 414
```

In [10]:
```
# updating global variable inside f() in presence of local variable
def f1():
    global a
    a=100
    print("Function 1:",a)
def f3():
    a=655
    globals()["a"]=414
    print("Function 3 - Local:",a)
    print("Function 3 - Global:",globals()["a"])


f1()
f3()
print(a)
```

```
Function 1: 100
Function 3 - Local: 655
Function 3 - Global: 414
414
```

In [12]:
```
# order of f() calling and chane in output
def f1():
    global a
    a=100
    print("Function 1:",a)
def f3():
    a=655
    globals()["a"]=414
    print("Function 3 - Local:",a)
    print("Function 3 - Global:",globals()["a"])


f3()
f1()
print(a)
```

```
Function 3 - Local: 655
Function 3 - Global: 414
Function 1: 100
100
```

Lambda / Anonymous Function

In [13]:
```
def square(n):
    return n*n

print(square(6))
```

```
36
```

In [14]:
```
# (lambda_keyword parameter:operation)(calling)
(lambda n:n*n)(7)
```

Out[14]:
```
49
```

In [15]:
```
(lambda a,b,c:(a+b+c)-(a-b-c))(1,2,3)
```

Out[15]:
```
10
```

In [16]:
```
(lambda a,b:a if a>b else b)(10,12)
```

Out[16]:
```
12
```

In [17]:
```
lambda a,b:a if a>b else b  # definition of lambda f()
```

Out[17]:
```
<function __main__.<lambda>(a, b)>
```

In [18]:
```
# storing lambda f() for calling
a=lambda a,b:a if a>b else b
print(a)
print(a(45,56))
```

```
<function <lambda> at 0x0000022047F172E0>
56
```

In [21]:
```
# lambda f() for variable length argument
x=lambda *arg:print(arg)
x(4,7,2,4,6)
(x)(4,7,2,4,6)
```

```
(4, 7, 2, 4, 6)
(4, 7, 2, 4, 6)
```

In [22]:
```
x=lambda *arg:print(arg)
l=[1,5,4,6,2,3,6,9,8,7]
x(l)  # passing complete list
x(*l) # passing list elements
```

```
([1, 5, 4, 6, 2, 3, 6, 9, 8, 7],)
(1, 5, 4, 6, 2, 3, 6, 9, 8, 7)
```

In [ ]:
```

```

Programs of Lambda Function

In [30]:
```
# 1) Write a Lambda f() to print even no. in a given list
even=(lambda *t:print([i for i in t if i%2==0]))
l=[1,2,4,5,15,7,8,9,6,54,46,1,2,3,6,5,4]
even(*l)
```

```
[2, 4, 8, 6, 54, 46, 2, 6, 4]
```

In [36]:
```
# 2) write a lambda expression to calculate area of a circle
r=float(input("Enter Radius of Circle: "))
(lambda r,p=3.14:print(f"Area of Circle: {p*r*r}"))(r)
```

```
Enter Radius of Circle: 7
Area of Circle: 153.86
```

In [7]:
```
# 3) write a lambda f() to find HCF of two no.
l=[int(i) for i in input("Enter 2  no. to find HCF by giving space: ").split()]
hcf=(lambda *t:[i for i in range(min(t),0,-1) if t[0]%i==0 and t[1]%i==0])(*l)
print(hcf[0])
```

```
Enter 2  no. to find HCF by giving space: 36 27
9
```

In [33]:
```
# 4) write a lambda f() to count words in a given text
cw=(lambda s:print(f"Your text has {len(s.split())} words"))
txt=input("Enter your text to count words: ")
cw(txt)
```

```
Enter your text to count words: nikhil vishwakarma cse ai&ds third sem sistec gandhi n
agar
Your text has 9 words
```

In [ ]:
```

```

Programs of Function

In [3]:
```
# write a python f() to print cube of a given no.
def cube():
    n=int(input("Enter a no. to find Cube: "))
    print(f"Cube of {n} is: {n*n}")

cube()
```

```
Enter a no. to find Cube: 5
Cube of 5 is: 25
```

In [13]:
```
# 1) write a f() to check whether a no. is even or not

def even():
    n=int(input("Enter a no.: "))
    if n%2==0:
        print(f"{n} is Even")
    else:
        print(f"{n} is Not Even")
even()
```

```
Enter a no.: 65
65 is Not Even
```

In [21]:
```
# 2) write a python f() to check whether a number is prime or not

def prime():
    prime=0
    n=int(input("Enter a no.: "))
    for i in range(1,n+1):
        if n%i==0:
            prime+=1
    if prime==2:
        print(f"{n} is Prime")
    else:
        print(f"{n} is Not Prime")
prime()
```

```
Enter a no.: 17
17 is Prime
```

In [31]:
```
# 3) write a python f() to create  list of prime no. b/w two given no.

def prime_no_List():
    prime=0
    sp,ep=[int(i) for i in input("Enter two no. by giving space to get l[] of prime no.: ").split()]
    l=[]
    n=sp+1
    while n<ep:
        for i in range(2,n):
            if n%i==0:
                prime+=1
        if prime==0:
            l.append(n)
        prime=0
        n+=1
    print(f"Prime no Between {sp} to {ep} : {l}")
prime_no_List()
```

```
Enter two no. by giving space to get l[] of prime no.: 10 20
Prime no Between 10 to 20 : [11, 13, 17, 19]
```

In [35]:
```
# 4) write a python f() to remove duplicate elements from a given list
def remove_Duplicate():
    s=[i for i in input("Enter Elements by giving space: ").split()]
    for i in s:
        if s.count(i)>1:
#             duplicate_Index = s.index(i)
            s.remove(i)
#             print(s)
#             s.insert(duplicate_Index,i)
    print(s)
remove_Duplicate()
```

```
Enter Elements by giving space: nikhil 25 monu 12 nikhil 12 kjgh gb 25
['monu', 'nikhil', '12', 'kjgh', 'gb', '25']
```

In [11]:
```
# 5) write a python f() which receives variable length argument to filter odd & even no.
def filter_even_odd(*t):
    odd = [i for i in t if i%2>0]
    even = [i for i in t if i%2==0]
    print("Odd no. :",odd)
    print("Even no. :",even)

n = [int(i) for i in input("Enter no. by giving space to filter odd & even no.: ").split()]
filter_even_odd(*n)
```

```
Enter no. by giving space to filter odd & even no.: 1 2 5 4 7 8 9 6 5 4 78 45 69 52 3 6 9 7 4 5 54 5 4
6 3 2 1
Odd no. : [1, 5, 7, 9, 5, 45, 69, 3, 9, 7, 5, 5, 3, 1]
Even no. : [2, 4, 8, 6, 4, 78, 52, 6, 4, 54, 4, 6, 2]
```

In [13]:
```
# 6) write a lambda f() to print all factors of a given no. in the form of list


n = int(input("Enter a no. to find Factors : "))
x = (lambda a:[i for i in range(1,a+1) if a%i==0])(n)
print(x)
```

```
Enter a no. to find Factors : 1024
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
```

In [ ]:
```

```

Recursion

In [1]:
```python
# a function calls itself -> termination condition to prevent errors

def fact(n):
    if n==1:
        return 1
    f = n * fact(n-1)
    return f

fact(6)
```

Out[1]: 720

In [3]:
```python
def printN(n):
    if n>0:
        printN(n-1)
        print(n, end=" ")

printN(10)
```

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

In [ ]:
```python

```