Basic ¶

In [1]:
```
3+5 #addition
```

Out[1]: 8

In [2]:
```
6-4 #subtraction
```

Out[2]: 2

In [3]:
```
4*4 #multiplication
```

Out[3]: 16

In [4]:
```
16/4 #division
```

Out[4]: 4.0

In [5]:
```
8/5 #float division
```

Out[5]: 1.6

In [6]:
```
4//2 #integer division
```

Out[6]: 2

In [7]:
```
15//4
```

Out[7]: 3

In [8]:
```
(5*5)+6-2 #BODMAS (bracket-orders-division-multiplication-addition-subtraction)
```

Out[8]: 29

In [9]:
```
3*3*3*3 #exponential
```

Out[9]: 81

In [10]:
```
3 * 4
```

Out[10]: 12

In [12]:
```
3**4
```

Out[12]: 81

In [13]:
```
15%3 #modulus
```

Out[13]: 0

In [14]:
```
19%2
```

Out[14]: 1

In [15]:
```
a,b,c,d= 1,'YOGITA', 1+2j ,5.5
print(a)
print(b)
print(c)
print(d)
```

```
1
YOGITA
(1+2j)
5.5
```

In [16]:
```
print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

```
<class 'int'>
<class 'str'>
<class 'complex'>
<class 'float'>
```

In [17]:
```
2+4
```

Out[17]: 6

In [18]:
```
a=4
b=6
```

In [19]:
```
a+b
```

Out[19]: 10

In [20]:
```
c=a+b
```

In [21]:
```
print(c)
```

```
10
```

In [33]:
```
# we have to print "Its's very "beautiful""
print('It's very "beautiful"')
```

```
  Cell In[33], line 2
    print('It's very "beautiful"')
                                  ^
SyntaxError: unterminated string literal (detected at line 2)
```

In [34]:
```
 #\ has some special meaning to ignore the error
print('It\'s very "beautiful"')
```

```
It's very "beautiful"
```

In [35]:
```
print("It's very 'beautiful' ")
```

```
It's very 'beautiful'
```

In [36]:
```
print("It's very "beautiful"")
```

```
  Cell In[36], line 1
    print("It's very "beautiful"")
                    ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

In [38]:
```
print("It's very \"beautiful\"")
```

```
It's very "beautiful"
```

In [39]:
```
'good'+'good'
```

Out[39]:
```
'goodgood'
```

In [41]:
```
'good ' + 'good' #space
```

Out[41]: 'good good'

In [42]:
```
5* ' good'
```

Out[42]: ' good good good good good'

In [44]:
```
print('hello\n"good morning"') #\n for new line
```

```
hello
"good morning"
```

Variable / Identifier / Object

In [45]:
```
x=5
```

In [46]:
```
x
```

Out[46]: 5

In [47]:
```
x+5
```

Out[47]: 10

In [48]:
```
y=2
```

In [49]:
```
y
```

Out[49]: 2

In [50]:
```
x+y
```

Out[50]: 7

In [51]:
```
z=x*y
```

In [52]:
```
z
```

Out[52]: 10

In [53]:
```
_ + z # _ understand the previous result
```

Out[53]: 20

In [54]:
```
_ + y
```

Out[54]: 22

In [55]:
```
_ + (x/y)
```

Out[55]: 24.5

String Variable

In [56]:
```
"world"
```

Out[56]: 'world'

In [58]:
```
"world" + ' nature'
```

Out[58]: 'world nature'

In [59]:
```
name="john"
```

In [61]:
```
name + " the topper"
```

Out[61]: 'john the topper'

In [62]:
```
len(name)
```

Out[62]: 4

In [63]:
```
name[0]
```

Out[63]: 'j'

In [64]:
```
name[3]
```

Out[64]: 'n'

In [65]:
```
name[-1]
```

Out[65]: 'n'

Slicing

In [71]:
```
p= 'boy'
```

In [72]:
```
p[0:2]
```

Out[72]: 'bo'

In [73]:
```
p[:]
```

Out[73]: 'boy'

In [74]:
```
p[:-1]
```

Out[74]: 'bo'

In [75]:
```
p[2:]
```

Out[75]: 'y'

In [76]:
```
#change boy to toy that is I have to change first character to 't'
p[0]='t'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[76], line 2
      1 #change boy to toy that is I have to change first character to 't'
----> 2 p[0]='t'

TypeError: 'str' object does not support item assignment
```

In [77]:
```
#strings in python are immutable
```

In [78]:
```
p[1:]
```

Out[78]: 'oy'

In [79]:
```
't' + p[1:] #change boy to toy
```

Out[79]: 'toy'

In [80]:
```
len(p)
```

Out[80]: 3

List

In [81]:
```
a=[10,11,50]
a
```

Out[81]: [10, 11, 50]

In [82]:
```
a[0]
```

Out[82]: 10

In [84]:
```
a[2]
```

Out[84]: 50

In [85]:
```
b=['Hey',' Have a Good Day']
b
```

Out[85]: ['Hey', ' Have a Good Day']

In [86]:
```
c=[1,55,9.2,'hii',1+3j,True]
c
```

Out[86]: [1, 55, 9.2, 'hii', (1+3j), True]

In [88]:
```
d=[a,b]
d
```

Out[88]: [[10, 11, 50], ['Hey', ' Have a Good Day']]

In [90]:
```
d=[a,b,c]
d
```

Out[90]: [[10, 11, 50], ['Hey', ' Have a Good Day'], [1, 55, 9.2, 'hii', (1+3j), True]]

In [92]:
```
c
```

Out[92]: [1, 55, 9.2, 'hii', (1+3j), True, 9]

In [93]:
```
c.append(9)
c
```

Out[93]: [1, 55, 9.2, 'hii', (1+3j), True, 9, 9]

In [94]:
```
c.remove(9)
```

In [95]:
```
c
```

Out[95]: [1, 55, 9.2, 'hii', (1+3j), True, 9]

In [101]:
```
c.pop(1)
c
```

Out[101]: [1, 9.2, 'hii', (1+3j), True, 9]

In [102]:
```
#55 is pop
```

In [103]:
```
c.pop() #if you dont assign the index element then it will consider by default last index
```

Out[103]: 9

In [104]:
```
c
```

Out[104]:
```
[1, 9.2, 'hii', (1+3j), True]
```

In [105]:
```
c.insert(3,'weldone') #insert the value as per index values i.e 3rd index we are assigning 'weldone'
c
```

Out[105]:
```
[1, 9.2, 'hii', 'weldone', (1+3j), True]
```

In [106]:
```
#if you want to delate multiple value
del c[2:5]
```

In [107]:
```
c
```

Out[107]:
```
[1, 9.2, True]
```

In [108]:
```
# if you need to add multiple values
c.extend(['Hello','good job','great',999])
c
```

Out[108]:
```
[1, 9.2, True, 'Hello', 'good job', 'great', 999]
```

In [109]:
```
a
```

Out[109]:
```
[10, 11, 50]
```

In [110]:
```
min(a) #inbuild function
```

Out[110]:
```
10
```

In [111]:
```
max(a)
```

Out[111]: 50

In [112]:
```
sum(a)
```

Out[112]: 71

In [113]:
```
b
```

Out[113]: ['Hey', ' Have a Good Day']

In [114]:
```
a
```

Out[114]: [10, 11, 50]

In [116]:
```
a.sort() #sort method
a
```

Out[116]: [10, 11, 50]

In [118]:
```
a.sort(reverse=True)
a
```

Out[118]: [50, 11, 10]

Tuple

In [119]:
```
a=(1,2,3)
a
```

Out[119]: (1, 2, 3)

In [120]:
```
type(a)
```

Out[120]: tuple

In [123]:
```
a.append(2) #tuple is immutable
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[123], line 1
----> 1 a.append(2)

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

In [125]:
```
a.count(2)
```

Out[125]: 1

In [126]:
```
a.index(3)
```

Out[126]: 2

In [127]:
```
a[1]
```

Out[127]: 2

SET

In [128]:
```
s={}
```

In [129]:
```
type(s)
```

Out[129]: dict

In [132]:
```
s={1}
```

In [133]:
```
type(s)
```

Out[133]:
```
set
```

In [135]:
```
s={1,66,6.7,'blessings'}
s
```

Out[135]:
```
{1, 6.7, 66, 'blessings'}
```

In [137]:
```
s.add(2)
s
```

Out[137]:
```
{1, 2, 6.7, 66, 'blessings'}
```

In [138]:
```
#set print in ascending order
```

Dictionary

In [139]:
```
d={1:'rose',2:'lotus',3:'hibiscus'}
d
```

Out[139]:
```
{1: 'rose', 2: 'lotus', 3: 'hibiscus'}
```

In [140]:
```
d[1]
```

Out[140]:
```
'rose'
```

In [141]:
```
d[3]
```

Out[141]:
```
'hibiscus'
```

In [142]:
```
d.get(2)
```

Out[142]:
```
'lotus'
```

In [143]:
```
print(d.get(3))
```

```
hibiscus
```

In [144]:
```
d[4]='lilly'
d
```

Out[144]:
```
{1: 'rose', 2: 'lotus', 3: 'hibiscus', 4: 'lilly'}
```

In [145]:
```
del d[4]
d
```

Out[145]:
```
{1: 'rose', 2: 'lotus', 3: 'hibiscus'}
```

In [146]:
```
#List in the dictionary
prog = {'python':['vscode', 'pycharm'], 'machine learning' : 'sklearn', 'datascience':['jupyter','spyder'] }
prog
```

Out[146]:
```
{'python': ['vscode', 'pycharm'],
 'machine learning': 'sklearn',
 'datascience': ['jupyter', 'spyder']}
```

In [148]:
```
prog['python']
```

Out[148]:
```
['vscode', 'pycharm']
```

In [150]:
```
prog['datascience']
```

Out[150]:
```
['jupyter', 'spyder']
```

In [ ]:
```
help() #for any help
```

ID

In [1]:
```
num=10
id(num)
```

Out[1]: 2862548976144

In [2]:
```
name='yogita'
id(name)
```

Out[2]: 2860481710512

In [3]:
```
a=5
id(5)
```

Out[3]: 2862548975984

In [4]:
```
b=a
id(b)  #python is memory efficient
```

Out[4]: 2862548975984

In [5]:
```
k=10
id(k)
```

Out[5]: 2862548976144

In [6]:
```
a = 20  # as we change the value of a then address will change
id(a)
```

Out[6]: 2862548976464

In [7]:
```
id(b)
```

Out[7]: 2862548975984

In [8]:
```
PI = 3.14  #in math this is alway constant but python we can chang
PI
```

Out[8]: 3.14

In [9]:
```
PI=3.15
PI
```

Out[9]: 3.15

In [10]:
```
type(PI)
```

Out[10]: float

Data Types & Data Structures

In [11]:
```
a=5.2
type(a)
```

Out[11]: float

In [14]:
```
c=3-7j
type(c)
```

Out[14]: complex

In [16]:
```
c.imag
```

Out[16]: -7.0

In [17]:
```
c.real
```

Out[17]: 3.0

In [18]:
```
c.conjugate
```

Out[18]: <function complex.conjugate()>

In [19]:
```
c.conjugate()
```

Out[19]: (3+7j)

In [20]:
```
p=4
p
```

Out[20]: 4

In [21]:
```
#convert int to float
q=float(p)
q
```

Out[21]: 4.0

In [22]:
```
#float to complex
k=complex(q)
k
```

Out[22]: (4+0j)

In [23]:
```
print(p)
print(q)
print(k)
```

```
4
4.0
(4+0j)
```

In [24]:
```
print(type(p))
print(type(q))
print(type(k))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

In [27]:
```
k1=complex(p,q)
k1
```

Out[27]: (4+4j)

In [28]:
```
r=9
r
```

Out[28]: 9

In [29]:
```
t=3
t
```

Out[29]: 3

In [30]:
```
t<r
```

Out[30]: True

In [31]:
```
condition=t>r
condition
```

Out[31]: False

In [32]:
```
int(True)
```

Out[32]: 1

In [33]:
```
int(False)
```

Out[33]: 0

In [34]:
```
l=[1,2,3,4,4,5] #list
print(l)
print(type(l))
```

```
[1, 2, 3, 4, 4, 5]
<class 'list'>
```

In [37]:
```
s={3,2,3,4,5,8,2,2,1,9} #set
print(s) #duplicates not allowed,,print in ascending order
print(type(s))
```

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

In [39]:
```
t=(11,11,34,11) #tuple
t
```

Out[39]: (11, 11, 34, 11)

In [40]:
```
type(t)
```

Out[40]: tuple

In [41]:
```
str = 'hello' #we dont have character in python
type(str)
```

Out[41]: str

In [42]:
```
str1='a'
print(str1)
type(str1)
```

```
a
```

Out[42]: str

Range()

In [43]:
```
range(0,10)
```

Out[43]: range(0, 10)

In [44]:
```
range(5)
```

Out[44]: range(0, 5)

In [45]:
```
# if you want to print the range
list(range(0,4))
```

Out[45]: [0, 1, 2, 3]

In [46]:
```
#if you want to print even number
even_number = list(range(2,10,2))
even_number
```

Out[46]: [2, 4, 6, 8]

In [47]:
```
d= {1:'one', 2:'two', 3:'three'} #dictionary
d
```

Out[47]: {1: 'one', 2: 'two', 3: 'three'}

In [48]:
```
type(d)
```

Out[48]: dict

In [50]:
```
d.keys()
```

Out[50]: dict_keys([1, 2, 3])

In [51]:
```
d.values()
```

Out[51]: dict_values(['one', 'two', 'three'])

In [52]:
```
# how to get particular value
d[2]
```

Out[52]: 'two'

In [53]:
```
# other way to get value as
d.get(2)
```

Out[53]: 'two'

Operators

1.Arithmetic Operators

In [54]:
```
x=5
y=10
x,y
```

Out[54]: (5, 10)

In [55]:
```
x+y
```

Out[55]: 15

In [57]:
```
y-x
```

Out[57]: 5

In [58]:
```
y/x
```

Out[58]: 2.0

In [59]:
```
int(y/x)
```

Out[59]: 2

In [60]:
```
x*y
```

Out[60]: 50

In [62]:
```
x1=2
y1=3
x1**y1
```

Out[62]: 8

2.Assignment Operators

In [66]:
```
x3=3
x3
```

Out[66]: 3

In [67]:
```
x3 = x3 + 2 # if you want to increment by 2
x3
```

Out[67]: 5

In [68]:
```
x3 += 2
x3
```

Out[68]: 7

In [69]:
```
x3 += 2
x3
```

Out[69]: 9

In [70]:
```
x3 *= 2
x3
```

Out[70]: 18

In [71]:
```
x3 -=8
x3
```

Out[71]: 10

In [72]:
```
x3/=5
x3
```

Out[72]: 2.0

In [74]:
```
a, b = 5,6 # you can assigned variable in one line as well
print(a)
b
```

```
5
```

Out[74]: 6

3.Unary Operator

In [76]:
```
n = 7 #negation
n
```

Out[76]: 7

In [77]:
```
-(n)
```

Out[77]: -7

In [78]:
```
-n
```

Out[78]: -7

4. Relational Operators

In [79]:
```
a=5
b=7
```

In [80]:
```
a
```

Out[80]: 5

In [81]:
```
b
```

Out[81]: 7

In [82]:
```
a==b
```

Out[82]: False

In [83]:
```
a!=b
```

Out[83]: True

In [84]:
```
a<b
```

Out[84]: True

In [85]:
```
a>b
```

Out[85]: False

In [86]:
```
a<=b
```

Out[86]: True

In [87]:
```
a>=b
```

Out[87]: False

5. Logical Operators

In [90]:
```
a=6
b=8
```

In [91]:
```
a,b
```

Out[91]: (6, 8)

In [92]:
```
a<9 and b>3 #both condtions should True
```

Out[92]: True

In [93]:
```
a<5 and b<9
```

Out[93]: False

In [94]:
```
a<5 or b<9 #atleast one condition should be True
```

Out[94]: True

In [95]:
```
a<5 or b>9 #if both contions are False
```

Out[95]: False

In [96]:
```
x= True
```

In [97]:
```
not x  # you can reverse the operation
```

Out[97]: False

Number System Conversion

binary= base 2|0-1

octal= base 8|0-7

decimal= base 10|0-9

hexadecimal= base 16|0-9 a-f

In [98]:
```
bin(25)
```

Out[98]: '0b11001'

In [99]:
```
0b11001
```

Out[99]: 25

In [100]:
```
int(0b11001)
```

Out[100]: 25

In [101]:
```
oct(3)
```

Out[101]: '0o3'

In [102]:
```
0o3
```

Out[102]: 3

In [103]:
```
hex(4)
```

Out[103]: '0x4'

In [104]:
```
hex(16)
```

Out[104]: '0x10'

In [105]:
```
0xb
```

Out[105]: 11

In [106]:
```
0xf
```

Out[106]: 15

Swapping

In [107]:
```
a=5
b=6
```

In [108]:
```
a=b
b=a
```

In [109]:
```
print(a)
print(b)
```

```
6
6
```

In [110]:
```
# in above scenario we lost the value 5
a1 = 7
b1 = 8
```

In [111]:
```
temp = a1
a1 = b1
b1 = temp
```

In [113]:
```
print(a1)
print(b1)
```

```
8
7
```

In [114]:
```
a2 = 5
b2 = 6
```

In [115]:
```
#swap variables without using 3rd variable
a2 = a2 + b2 # 5+6 = 11
b2 = a2 - b2 # 11-6 = 5
a2 = a2 - b2 # 11-5 = 6
```

In [116]:
```
print(a2)
print(b2)
```

```
6
5
```

In [117]:
```
print(0b101) # 101 is 3 bit
print(0b110) # 110 also 3bit
```

```
5
6
```

In [118]:
```
#but when we use a2 + b2 then we get 11 that means we will get 4 bit which is 1 bit extra
print(bin(11))
print(0b1011)
```

```
0b1011
11
```

swapping using XOR

In [119]:
```
print(a2)
print(b2)
```

```
6
5
```

In [120]:
```
#there is other way to work using swap variable also which is XOR because it will not waste extra bit
a2 = a2 ^ b2
b2 = a2 ^ b2
a2 = a2 ^ b2
```

In [121]:
```
print(a2)
print(b2)
```

```
5
6
```

In [122]:
```
a2 ,b2 = b2, a2 # how it work is b2 6 a2 is 5 first it goes into stack & then it reverse the 2 vlaue
```

In [123]:
```
print(a2)
print(b2)
```

```
6
5
```

6. Bitwise Operators

1.complement(~)

2.and(&)

3.or(|)

4.xor(^)

5.left shift(<<)

6.right shift(>>)

In [1]:
```
print(bin(12))
print(bin(13))
```

```
0b1100
0b1101
```

In [2]:
```
0b1100
```

Out[2]: 12

In [3]:
```
0b1101
```

Out[3]: 13

1.complement(~)

In [4]:
```
~12 # why we get -13 . first we understand what is complment means (reverse of binary format)
```

Out[4]: -13

In [5]:
```
~-1
```

Out[5]: 0

In [6]:
```
~-5
```

Out[6]: 4

In [7]:
```
~5
```

Out[7]: -6

2.and(&)

In [8]:
```
2 & 3
```

Out[8]: 2

In [9]:
```
3 & 3
```

Out[9]: 3

In [10]:
```
20 & 40
```

Out[10]: 0

In [11]:
```
1 & 1
```

Out[11]: 1

In [12]:
```
1 & 0
```

Out[12]: 0

In [13]:
```
0 & 0
```

Out[13]: 0

3.or(|)

In [14]:
```
1 & 0
```

Out[14]: 0

In [15]:
```
5 & 6
```

Out[15]: 4

In [16]:
```
0 & 0
```

Out[16]: 0

In [17]:
```
2 & 2
```

Out[17]: 2

4.xor(^)

In [18]:
```
# in XOR if the both number are different then we will get 1 or else we will get 0

12 ^ 13
```

Out[18]: 1

In [19]:
```
5 ^ 5
```

Out[19]: 0

In [20]:
```
2 ^ 3
```

Out[20]: 1

In [21]:
```
1 ^ 0
```

Out[21]: 1

5.left shift(<<)

In [22]:
```
# BIT WISE LEFT SHIFT OPERATOR
# in left shift what we need to do we need shift in left hand side & need to shift 2 bits
#bit wise left operator bydefault you will take 2 zeros ( )
#10 binary operator is 1010 | also i can say 1010
10<<2
```

Out[22]: 40

In [23]:
```
5<<2
```

Out[23]: 20

In [24]:
```
2<<1
```

Out[24]: 4

In [25]:
```
2<<2
```

Out[25]: 8

In [26]:
```
2<<3
```

Out[26]: 16

In [27]:
```
2<<4
```

Out[27]: 32

6.right shift(>>)

In [28]:
```
10>>1
```

Out[28]: 5

In [29]:
```
10>>2
```

Out[29]: 2

In [30]:
```
30>>3
```

Out[30]: 3

In [31]:
```
100>>4
```

Out[31]: 6

In [32]:
```
5>>5
```

Out[32]: 0

In [33]:
```
5>>4
```

Out[33]: 0

In [34]:
```
5>>2
```

Out[34]: 1

Import math function

In [36]:
```
import math #math is module
```

In [37]:
```
x = sqrt(25) #sqrt is inbuild function
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[37], line 1
----> 1 x = sqrt(25)

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

In [38]:
```
x = math.sqrt(25)
x
```

Out[38]: 5.0

In [39]:
```
x1 = math.sqrt(100)
x1
```

Out[39]: 10.0

In [41]:
```
print(math.floor(2.5)) #floor - minimum or least value
```

```
2
```

In [42]:
```
print(math.ceil(2.5)) #ceil - maximum or highest value
```

```
3
```

In [43]:
```
print(math.floor(3.2))
```

```
3
```

In [44]:
```
print(math.ceil(3.2))
```

```
4
```

In [45]:
```
print(math.floor(2.0))
```

```
2
```

In [46]:
```
print(math.ceil(2.0))
```

```
2
```

In [47]:
```
print(math.ceil(9))
```

```
9
```

In [48]:
```
print(math.pow(2,3))
```

```
8.0
```

In [49]:
```
print(math.pow(2,5))
```

```
32.0
```

In [50]:
```
print(math.pi) #these are constant
```

```
3.141592653589793
```

In [51]:
```
print(math.e) # e - epsilon values
```

```
2.718281828459045
```

In [53]:
```
math.sqrt(625)
```

Out[53]:
```
25.0
```

In [55]:
```
import math as m # we need to use concept aliseing, instead of math we are using as m
m.sqrt(9)
```

Out[55]:
```
3.0
```

In [56]:
```
from math import sqrt,pow # math has many function if you want to import specific function then use import
print(pow(3,4))
print(math.sqrt(64))
```

```
81.0
8.0
```

In [57]:
```
round(pow(3,4))
```

Out[57]: 81

User input function in python

In [1]:
```
x = input()
y = input()
z = x + y
print(z) # console is waiting for user to enter input
```

```
3
5
35
```

In [3]:
```
x1 = input('Enter the 1st number') #whenevery you works in input function it always give you string
y1 = input('Enter the 2nd number') # it wont understand as arithmetic operator
z1 = x1 +  y1
print(z1)
```

```
Enter the 1st number50
Enter the 2nd number40
5040
```

In [5]:
```
print(type(x1))
type(y1)
```

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

Out[5]:
```
str
```

In [6]:
```
x1 = input('Enter the 1st number') #whenevery you works in input function it always give you string
a1 = int(x1)
y1 = input('Enter the 2nd number') # it wont understand as arithmetic operator
b1 = int(y1)
z1 = a1 + b1
print(z1)
```

```
Enter the 1st number3
Enter the 2nd number4
7
```

In [7]:
```
x2 = int(input('Enter the 1st number'))
y2 = int(input('Enter the 2nd number'))
z2 = x2 + y2
z2
```

```
Enter the 1st number50
Enter the 2nd number40
```

Out[7]:
```
90
```

In [9]:
```
ch = input('enter a char')
print(ch)
print(type(ch)) #in python we dont't have character
```

```
enter a chara
a
<class 'str'>
```

In [10]:
```
ch = input('enter a char')
print(ch)
```

```
enter a charyogita
yogita
```

In [11]:
```
print(ch[0])
```

```
y
```

In [12]:
```
print(ch[0:2])
```

```
yo
```

In [13]:
```
print(ch[:])
```

```
yogita
```

In [15]:
```
print(ch[0:3])
```

```
yog
```

In [16]:
```
ch = input('enter a char')[0]
print(ch)
```

```
enter a charyogita
y
```

In [17]:
```
ch = input('enter a char')[1:3]
print(ch)
```

```
enter a charyogita
og
```

In [18]:
```
ch = input('enter a char')
print(ch) # if you enter as 2 + 6 -1 we get output as 2 + 6-1 only cuz 2+6-1 as expression
```

```
enter a char4+6
4+6
```

In [19]:
```
result = eval(input('enter an expr'))
print(result)
```

```
enter an expr4*6
24
```

if statement

if

if else

if elif else

nested if

if

In [2]:
```
if True: # indentiation is always 4 spaces
    print('Data Science')
```

```
Data Science
```

In [3]:
```
if False:
    print('Data Science')
print('bye for now')
```

```
bye for now
```

In [4]:
```
if True: # indentiation is always 4 spaces
    print('Data Science')
print('bye for now')
```

```
Data Science
bye for now
```

In [5]:
```
#to print only even number
x = 4
r = x % 2
if r == 0:
    print('Even number')
```

```
Even number
```

In [6]:
```
x = 5
r = x % 2
if r == 0:
    print('Even number')
print('odd number')
```

```
odd number
```

In [7]:
```
x = 8
r = x % 2
if r == 0:
    print('Even number')
print('odd number')
```

```
Even number
odd number
```

In [8]:
```
x = 8
r = x % 2
if r == 0:
    print('Even number')
if r == 1:
    print('odd number')
```

```
Even number
```

In [9]:
```
x = 7
r = x % 2
if r == 0:
    print('Even number')
if r == 1:
    print('odd number')
```

```
odd number
```

In [10]:
```
x = 15
r = x % 2
if r == 0:
    print('Even number')
if r != 0:
    print('odd number')
```

```
odd number
```

if else

In [11]:
```
x = 5
r = x % 2
if r == 0:
    print('Even number')
else:
    print('Odd Number')
```

```
Odd Number
```

nested if

In [12]:
```
x = 3
r = x % 2
if r == 0:
    print(' Even number')
    if x>5:
        print('greater number')
else:
    print('Odd Number')
```

```
Odd Number
```

In [13]:
```
x = 4
r = x % 2
if r == 0:
    print('Even number')
    if x>5 :
        print('greater number')
    else:
        print('not greater ')
else:
    print('odd number')
```

```
Even number
not greater
```

if elif else

In [14]:
```
#when you use if it will check all condition but if we mention elif then it wont check all condition
# when we use if condition it will check all every block of code  better debug in pycharm
# you can debug with value 1 & d for both if & elif

x = 1

if x == 1:
    print('one')
if x == 2:
    print('Two')
if x == 3:
    print('Three')
if x == 4:
    print('four')
```

```
one
```

In [15]:
```
# elif it wont check till the block once you find the output it wont go to next line
# you can try with multiple parameter 1, 2 & 3 value in x

x = 1

if(x == 1):
    print('one')
elif(x == 2):
    print('Two')
elif(x == 3):
    print('Three')
elif(x == 4):
    print('four')
```

```
one
```

In [16]:
```
x = 7

if(x == 1):
    print('one')
elif(x == 2):
    print('Two')
elif(x == 3):
    print('Three')
elif(x == 4):
    print('four') #nothing is print
```

In [17]:
```
x = 15

if(x == 1):
    print('one')
elif(x == 2):
    print('Two')
elif(x == 3):
    print('Three')
elif(x == 4):
    print('four')
else:
    print('wrong output')
```

```
wrong output
```

In [18]:
```
x = 5

if(x == 1):
    print('one')
elif(x == 2):
    print('Two')
elif(x == 3):
    print('Three')
elif(x == 4):
    print('four')

else:
    print('wrong output')
```

```
wrong output
```

Loops

In [19]:
```
print('data science')
print('data science')
print('data science')
print('data science')
print('data science')
```

```
data science
data science
data science
data science
data science
```

while loop

In [20]:
```
i = 1          # initializing
while i<=5:    # condition
    print('data science')
    i = i + 1  # increment
```

```
data science
data science
data science
data science
data science
```

In [23]:
```
i = 5          # initializing
while i>=1:    # condition
    print('data science')
    i = i - 1  # decrement
```

```
data science
data science
data science
data science
data science
```

In [24]:
```
i = 1          # initializing
while i<=5:    # condition
    print('machine learning',i)
    i = i + 1  # increment
```

```
machine learning 1
machine learning 2
machine learning 3
machine learning 4
machine learning 5
```

In [25]:
```
i = 5          # initializing
while i>=1:    # condition
    print('machine learning',i)
    i = i - 1  # decrement
```

```
machine learning 5
machine learning 4
machine learning 3
machine learning 2
machine learning 1
```

nested while loop

In [27]:

```
i = 1
while i<=5:
    print(' Data Science') # when we mention end then new line will not create
    j = 1
    while j<=4:
        print('Deep Learning')
        j = j + 1

    i = i + 1
    print()

    # the output which we got is very lengty but how to make them one line lets refer to below code
```

```
 Data Science
Deep Learning
Deep Learning
Deep Learning
Deep Learning

 Data Science
Deep Learning
Deep Learning
Deep Learning
Deep Learning

 Data Science
Deep Learning
Deep Learning
Deep Learning
Deep Learning

 Data Science
Deep Learning
Deep Learning
Deep Learning
Deep Learning

 Data Science
Deep Learning
Deep Learning
Deep Learning
Deep Learning
```

In [28]:
```
i = 1
while i<=5:
    print(' Data Science', end = "") # when we mention end then new line will not create
    j = 1
    while j<=4:
        print(' Deep Learning', end="")
        j = j + 1

    i = i + 1
    print()
```

```
 Data Science Deep Learning Deep Learning Deep Learning Deep Learning
 Data Science Deep Learning Deep Learning Deep Learning Deep Learning
 Data Science Deep Learning Deep Learning Deep Learning Deep Learning
 Data Science Deep Learning Deep Learning Deep Learning Deep Learning
 Data Science Deep Learning Deep Learning Deep Learning Deep Learning
```

In [29]:
```
# lets use while loop usig some numbers
i = 1
while i <= 2 :
    j = 0
    while  j <= 2 :
        print(i*j, end=" ")
        j += 1
    print()
    i += 1
```

```
0 1 2
0 2 4
```

In [30]:
```
# lets use while loop usig some numbers
i = 1
while i <= 4 :
    j = 0
    while  j <= 3 :
        print(i*j, end=" ")
        j += 1
    print()
    i += 1
```

```
0 1 2 3
0 2 4 6
0 3 6 9
0 4 8 12
```

for loop

In [31]:
```
name = 'Beautiful'
for i in name:
    print(i)
```

```
B
e
a
u
t
i
f
u
l
```

In [32]:
```
name1 = [1,3.5,'hello'] #i want print the value individualy
for i in name1:
    print(i)
```

```
1
3.5
hello
```

In [33]:
```
for i in [2, 3, 7.8, 'hi']:
    print(i)
```

```
2
3
7.8
hi
```

In [35]:
```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

In [36]:
```
for i in range(1,5):
    print(i)
```

```
1
2
3
4
```

In [37]:
```
for i in range(1,10,3):
    print(i)
```

```
1
4
7
```

In [38]:
```
# print the value which is divisible by 5 i dont want that value
for i in range(1,21):
    if i%5!=0 :
        print(i)
```

```
1
2
3
4
6
7
8
9
11
12
13
14
16
17
18
19
```

In [39]:
```
# print the value which is divisible by 5
for i in range(1,21):
    if i%5==0 :
        print(i)
```

```
5
10
15
20
```

break , continue , pass (keywords)

In [40]:
```
# write the code user ask chocolates from vendor machine write the basic code

x = int(input('How many chocolates you want:?'))

i = 1
while i<=x:
    print('chocolate')
    i += 1
```

```
How many chocolates you want:?10
chocolate
chocolate
chocolate
chocolate
chocolate
chocolate
chocolate
chocolate
chocolate
chocolate
chocolate
```

In [41]:
```
available = 5 # the machine has only 5 chocolate

x = int(input('How many chocolates you want:?'))

i = 1
while i<=x:
    print('chocolate')
    i += 1
# if you check the user wants 10 chocolates  but availabe chocolates are 5 but we got output as 10 choclet
# in this code we just declare but we didn't apply any condition to it
```

```
How many chocolates you want:?5
chocolate
chocolate
chocolate
chocolate
chocolate
```

In [44]:
```
available_chocolate = 5 # the machine has only 10 candis

x = int(input('How many chocolates you want:?'))

i = 1
while i<=x:

    if i>available_chocolate: # we stop the execution but which code execution not entire code , i want to come ou
        break # break is statement | means jump out of the loop
    print('chocolate')
    i += 1

print('bye for now')
```

```
How many chocolates you want:?10
chocolate
chocolate
chocolate
chocolate
chocolate
bye for now
```

In [47]:
```
available_chocolate = 5 # the machine has only 10 candis

x = int(input('How many chocolates you want:?'))

i = 1
while i<=x:

    if i>available_chocolate: # we stop the execution but which code execution not entire code , i want to come ou
        print('out of stock')
        break # break is statement | means jump out of the loop
    print('chocolate')
    i += 1

print('bye for now')
```

```
How many chocolates you want:?7
chocolate
chocolate
chocolate
chocolate
chocolate
chocolate
out of stock
bye for now
```

In [48]:
```
for i in range(1,11):
    print(i)
```

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

In [49]:
```
for i in range(1,11):
    if i == 5:
        break
    print(i)
```

```
1
2
3
4
```

In [50]:
```
for i in range(1,11):
    if i == 3:
        continue
    print(i)
```

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

In [51]:
```
for i in range(1,11):
    if i == 5:
        continue
    print('hello ',i)
```

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

In [52]:
```
for i in range(1,11):
```

```
  Cell In[52], line 1
    for i in range(1,11):
                         ^
SyntaxError: incomplete input
```

In [53]:
```
for i in range(1,11):
    pass
```

you need to print the number from 1 to 50 but dont print the value which is divisible by 3 or 5

In [54]:
```
for i in range(1,50):
    if i%3 == 0:
        print(i)
print('end')
```

```
3
6
9
12
15
18
21
24
27
30
33
36
39
42
45
48
end
```

In [55]:

```
for i in range(1,50):
    if i%3 == 0:
        continue
    print(i)
print('end')
```

```
1
2
4
5
7
8
10
11
13
14
16
17
19
20
22
23
25
26
28
29
31
32
34
35
37
38
40
41
43
44
46
47
49
end
```

In [56]:

```
for i in range(1,50):
    if i%3 == 0 or i%5 == 0:
        continue
    print(i)
print('end')
# it will skip all the value which is divisible by 3 or 5
```

```
1
2
4
7
8
11
13
14
16
17
19
22
23
26
28
29
31
32
34
37
38
41
43
44
46
47
49
end
```

In [57]:

```
for i in range(1,50):
    if i%3 == 0 and i%5 == 0:
        continue
    print(i)
print('end')
# when you apply and you wont get the value which is divisible by both 3 & 5 (15)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
16
17
18
19
20
21
22
23
24
25
26
27
28
29
31
32
33
34
35
36
37
38
39
40
41
42
43
```

```
44
46
47
48
49
end
```

In [59]:

```
for i in range(1,50):
    if i%3 == 0 and i%5 == 0:
        print(i)
print('end')
# when we dont't use continue here, it will print numbers with are divisible by 3&5
```

```
15
30
45
end
```

In [60]:
```
# i dont want to print the values which are odd numbers that means print only even numbers
for i in range(1,50):

    if (i%2 != 0):
        pass
    else:
        print(i)
print('bye')
```

```
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
bye
```

Print Patterns in Python

In [62]:
```
print('# # # #')
print('# # # #')
print('# # # #')
print('# # # #')
```

```
# # # #
# # # #
# # # #
# # # #
```

In [63]:
```
for j in range(4):
    print('#')
```

```
#
#
#
#
```

In [64]:
```
for j in range(4):
    print('#', end=" ")
```

```
# # # #
```

In [65]:
```
for j in range(4):
    print('#', end="  ")

for j in range(4):
    print('#', end="  ")
```

```
#  #  #  #  #  #  #  #
```

In [66]:
```
for j in range(4):
    print('#', end="  ")

print()

for j in range(4):
    print('#', end="  ")
```

```
#  #  #  #
#  #  #  #
```

In [67]:
```
for j in range(4):
    print('#', end="  ")

print()

for j in range(4):
    print('#', end="  ")

print()

for j in range(4):
    print('#', end="  ")

print()

for j in range(4):
    print('#', end="  ")
```

```
#  #  #  #
#  #  #  #
#  #  #  #
#  #  #  #
```

In [68]:
```
for i in range(4):
    for j in range(4):
        print('#', end="  ")
    print()
```

```
#  #  #  #
#  #  #  #
#  #  #  #
#  #  #  #
```

In [69]:
```
for i in range(5):
    for j in range(i):
        print('#', end="  ")
    print()
```

```

#
#  #
#  #  #
#  #  #  #
```

In [70]:
```
for i in range(5):
    for j in range(i+1):
        print('#', end="  ")
    print()
```

```
#
#  #
#  #  #
#  #  #  #
#  #  #  #  #
```

In [71]:
```
for i in range(4):
    for j in range(4-i):
        print('#', end="  ")
    print()
```

```
#  #  #  #
#  #  #
#  #
#
```

In [72]:
```
#pyramid
rows=5
for i in range(1,rows+1):
    for j in range(rows-i):
        print(" ",end="")
    for k in range(2*i-1):
        print("*",end="")
    print()
```

```
    *
   ***
  *****
 *******
*********
```

In [75]:
```
#pyramid
rows=3
for i in range(1,rows+1):
    for j in range(rows-i):
        print(" ",end="")
    for k in range(2*i-1):
        print("*",end="")
    print()
```

```
  *
 ***
*****
```

for else

In [76]:
```
nums = [12,15,18,21,26]
for num in nums:
    if num % 5 == 0:
        print(num)
```

```
15
```

In [77]:
```
nums = [12,14,18,21,25,30,35]
for num in nums:
    if num % 5 == 0:
        print(num)
```

```
25
30
35
```

In [78]:
```
nums = [12,14,18,21,25,20]
for num in nums:
    if num % 5 == 0:
        print(num)
```

```
25
20
```

In [79]:
```
nums = [12,14,18,21,25,20]
for num in nums:
    if num % 5 == 0:
        print(num)
        break
```

```
25
```

In [80]:
```
nums = [12,14,18,21,20,25]
for num in nums:
    if num % 5 == 0:
        print(num)
        break
```

```
20
```

In [81]:
```
nums = [10,14,18,21,5,10]
for num in nums:
    if num % 5 == 0:
        print(num)
        break #it will print only 1 number then it break
```

```
10
```

In [82]:
```
nums = [10,14,18,21,25,20]
for num in nums:
    if num % 5 == 0:
        print(num)
        continue
```

```
10
25
20
```

In [83]:
```
nums = [7,14,18,21,23,27] #hear there is no number which is divisible by 5 we got output as blank
for num in nums:
    if num % 5 == 0:
        print(num)
        break
```

In [84]:
```
nums = [7,14,18,21,23,27]
for num in nums:
    if num % 5 == 0:
        print(num)
        break
    else:
        print('Number Not Found') #every iteration it cheking condition
```

```
Number Not Found
Number Not Found
Number Not Found
Number Not Found
Number Not Found
Number Not Found
```

In [85]:
```
nums = [7,14]
for num in nums:
    if num % 5 == 0:
        print(num)
        break
    else:
        print('Number Not Found') #every iteration it cheking condition
```

```
Number Not Found
Number Not Found
```

In [86]:
```
nums = [7,14,18,21,23,27]
for num in nums:
    if num % 5 == 0:
        print(num)
        break
else:
        print('Number Not Found') # hear else we dont write in if block but we can write in for block only
```

```
Number Not Found
```

In [87]:
```
nums = [10,14,18,21,20,27]
for num in nums:
    if num % 5 == 0:
        print(num)
        #break
else:
        print('Not Found')
```

```
10
20
Not Found
```

In [88]:
```
nums = [10,14,18,21,20,27,30]
for num in nums:
    if num % 5 == 0:
        print(num)
        #break
else:
        print('Not Found')
```

```
10
20
30
Not Found
```

In [90]:
```
nums = [10,14,18,21,20,27,30]
for num in nums:
    if num % 5 == 0:
        print(num)
        break
else:
        print('Not Found')
```

```
10
```

In [92]:
```
nums = [10,14,18,21,20,27,30]
for num in nums:
    if num % 5 == 0:
        print(num)
        continue
```

```
10
20
30
```

In [93]:
```
num = 12
for i in range(2,num):
    if num % i == 0:
        print('Not prime Number')
        break
else:
    print('Prime Number')
```

```
Not prime Number
```

In [94]:
```
num = 13
for i in range(2,num):
    if num % i == 0:
        print('Not prime Number')
        break
else:
    print('Prime Number')
```

```
Prime Number
```

Array in Python

In [95]:
```
from array import *
arr = array('i',[])

n = int(input('Enter the length of the array'))

for i in range(5):
    x = int(input('Enter the next value'))
    arr.append(x)
print(arr)
```

```
Enter the length of the array6
Enter the next value1
Enter the next value2
Enter the next value3
Enter the next value4
Enter the next value5
array('i', [1, 2, 3, 4, 5])
```

In [96]:
```
from numpy import *
arr = array([1,2,3,4,5])
print(arr)
type(arr)
```

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

Out[96]:
```
numpy.ndarray
```

In [97]:
```
print(arr.dtype)
```

```
int32
```

In [98]:
```
arr2 = array([1,2,3,4,5.9],float)
arr2
```

Out[98]:
```
array([1. , 2. , 3. , 4. , 5.9])
```

In [99]:
```
arr3 = array([1,2,3,4,5.6],int)
arr3
```

Out[99]:
```
array([1, 2, 3, 4, 5])
```

In [100]:
```
import numpy as np
```

In [101]:
```
arr4 = np.linspace(0, 16, 10) # break the code between 10 spaces between 0 to 16 but why decimal becuase we break
arr4
```

Out[101]:
```
array([ 0.        ,  1.77777778,  3.55555556,  5.33333333,  7.11111111,
        8.88888889, 10.66666667, 12.44444444, 14.22222222, 16.        ])
```

In [102]:
```
arr5 = np.arange(0,10,2) # arange - as range
arr5
```

Out[102]:
```
array([0, 2, 4, 6, 8])
```

In [103]:
```
ar=range(0,6)
```

In [105]:
```
ar
```

Out[105]:
```
range(0, 6)
```

In [106]:
```
arange(0,6)
```

Out[106]:
```
array([0, 1, 2, 3, 4, 5])
```

In [108]:
```
arr6=np.linspace(5,10,2)
arr6
```

Out[108]:
```
array([ 5., 10.])
```

In [109]:
```
arr7=np.linspace(5,10,3)
arr7
```

Out[109]:
```
array([ 5. ,  7.5, 10. ])
```

In [110]:
```
arr8 = np.zeros(5)
arr8
```

Out[110]:
```
array([0., 0., 0., 0., 0.])
```

In [111]:
```
arr9 = np.ones(5)
arr9
```

Out[111]:
```
array([1., 1., 1., 1., 1.])
```

In [112]:
```
arr6 = np.zeros(5,int)
arr6
```

Out[112]:
```
array([0, 0, 0, 0, 0])
```

In [ ]: