Variables

python variable is used to store values.

In [3]:
```
a = 10
```

In [5]:
```
print(a)
```

```
10
```

In [8]:
```
b = 9.8
print(B)
```

```
---------------------------------------------------------------------
NameError                                 Traceback (most recent c
all last)
<ipython-input-8-cd07d2818810> in <module>
      1 b = 9.8
----> 2 print(B)

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

In [9]:
```
print(b)
```

```
9.8
```

In [10]:
```
c = 'coder'
```

In [11]:
```
print(c)
```

```
coder
```

In [12]:
```
a = 56
```

In [16]:
```
#printing value of a
print(a)
```

```
56
```

Data types-

 # Numeric [# integer # complex number # Float]

 # Dictionary

 # Boolean

 # Set

 # String

 # List

 # Tuple

In [20]:
```
# integer
a = 9
```

In [21]:
```
# finding data type of stored value
type(a)
```

Out[21]:
```
int
```

In [22]:
```
# float
weight = 45.8
```

In [23]:
```
type(weight)
```

Out[23]:
```
float
```

In [40]:
```
# string
name = 'avengers'
type(name)
```

Out[40]:
```
str
```

In [45]:
```
# boolean

6>5
```

Out[45]:
```
True
```

In [46]:
```
1>100
```

Out[46]:
```
False
```

In [47]:
```
type(8>67)
```

Out[47]:
```
bool
```

In [52]:
```
# complex number  ( combination of real and imaginary number)

var1 = 5 + 7j

type(var1)
```

Out[52]:
```
complex
```

In [63]:
```
# list

l = [5,6,7,8,7.4,'python']
```

In [64]:
```
type(l)
```

Out[64]:
```
list
```

In [65]:
```
# Tuple

tup = (4,6,2,7.4,'python')
```

In [66]:
```
type(tup)
```

Out[66]:
```
tuple
```

In [74]:
```
# Dictionary  :{keys :values} pair

d = {"python" :3.8,'b':10 ,8:'programming'}
```

In [78]:
```
print(d)
```

```
{'python': 3.8, 'b': 10, 8: 'programming'}
```

In [75]:
```
d.keys()
```

Out[75]:
```
dict_keys(['python', 'b', 8])
```

In [76]:
```
d.values()
```

Out[76]:
```
dict_values([3.8, 10, 'programming'])
```

In [68]:
```
type(d)
```

Out[68]:
```
dict
```

In [79]:
```
# set  : no (keys :values) pair present

s = {4,8,10}
```

In [62]:
```
type(s)
```

Out[62]:
```
set
```

User input()

 # used to get input from the user .

 # we input the text we want to display to the user.

In [86]:
```
# input() : used to take input from user

name = input("Hiii whats your name ? ")
```

```
Hiii whats your name ? python
```

In [88]:
```
type(name)
```

Out[88]:
```
str
```

In [95]:
```
age = input("can you tell me your age : ")
```

```
can you tell me your age : 20
```

In [106]:
```
print(age)
```

```
21
```

In [96]:
```
# input gives output in terms of strings
type(age)
```

Out[96]:
```
str
```

Typecasting

In [97]:
```
# converting one data type to other datatype is called typecasting
```

In [98]:
```
age = int(input("can you tell me your age : "))
```

```
can you tell me your age : 21
```

In [99]:
```
type(age)
```

Out[99]:
```
int
```

In [101]:
```
print(3.14,int(3.14))
```

```
3.14 3
```

In [102]:
```
print(9,float(9))
```

```
9 9.0
```

In [ ]:
```

```

Operators

Arithmatic operators

In [46]:
```
x = 15
y = 4
```

In [ ]:
```
# Addition

ad = x+y
print("Addition of two numbers is: ",ad)
```

In [ ]:
```
# Subtraction

sub = x-y
print("Subtraction of two numbers is: ",sub)
```

In [ ]:
```
# Multiplication

mul = x*y

print("Multiplication of two numbers is: ",mul)
```

In [47]:
```
# Division ( / )

div = x/y
print("Division of two numbers is: ",div)
```

```
Division of two numbers is:  3.75
```

In [48]:
```
# Floor division

floor_div = x//y
print("Floor Division of two numbers is: ",floor_div)
```

```
Floor Division of two numbers is:  3
```

In [49]:
```
# Modulous division (Remainder)

mod_div = x%y
print("Mod Division of two numbers is: ",mod_div)
```

```
Mod Division of two numbers is:  3
```

In [ ]:
```
# power
a = 3
b = 2

sq = a**b
print(" value of 3^2 is :",sq)
```

In [ ]:
```

```

Identity Operators

In [1]:
```
x1 = 5
y1 = 5
x2 = "hello world"
y2 = "hello india"
```

In [2]:
```
# is : returs true if both the operands are similar

x1 is y1
```

Out[2]: True

In [4]:
```
# is not : returs true if both the operands are not similar
x2 is not y2
```

Out[4]: True

Membership Operator

In [5]:
```
x = "i am a coder"
```

In [6]:
```
# in

'H' in x
```

Out[6]: False

In [8]:
```
'i' in x
```

Out[8]: True

In [10]:
```
d = {1:'a',2:'b'}
```

In [11]:
```
1 in d
```

Out[11]: True

In [16]:
```
# not in

l = [1,2,3]
```

In [17]:
```
3 not in l
```

Out[17]: False

In [18]:
```
6 not in l
```

Out[18]: True

In [ ]:
```

```

Relational Operator

In [30]:
```
# equal to
1 == 2
```

Out[30]: False

In [28]:
```
2 ==2
```

Out[28]: True

In [31]:
```
# Not equal to

5!=6
```

Out[31]: True

In [32]:
```
5!=5
```

Out[32]: False

In [34]:
```
# less than
6<7
```

Out[34]: True

In [35]:
```
# greater than

18>400
```

Out[35]: False

In [36]:
```
# less than equal to : check for two conditions (less, equal)
# if any one is true, then my final answer is true

8<=10
```

Out[36]: True

In [40]:
```
# Greater than equal to : check for two conditions (greater, eq
# if any one is true, then my final answer is true


800>=500
```

Out[40]: True

In [38]:
```
# T T = T
# T F = T
# F T = T
# F F = F
```

In [43]:
```
7>=16
```

Out[43]: False

In [ ]:
```

```

Assignment Operators

In [52]:
```
# assign add   syntax : +=
a = 4
a+=2 # internally it performs a = a+2
```

In [53]:
```
a
```

Out[53]: 6

In [54]:
```
# assign sub   syntax : -=
b = 20
b-=5 # internally it performs b = b-5
print(b)
```

15

In [55]:
```
# assign Mul   syntax : *=
c = 8
c*=4 # internally it performs c = c*4
print(c)
```

32

In [56]:
```
# assign float division syntax : /=
d = 15
d/=4 # internally it performs d = d/4
print(d)
```

```
3.75
```

In [57]:
```
# assign floor division syntax : //=
e = 15
e//=6 # internally it performs e = e//6
print(e)
```

```
2
```

In [59]:
```
# assign Modulous division syntax : %=
f = 15
f%=6 # internally it performs f = f%6
print(f)
```

```
3
```

logical Operator

In [60]:
```
m = True
n = False
```

In [ ]:
```
# AND operator  syntax: and
```

In [ ]:
```
# T T = T
# T F = F
# F T = F
# F F = F
```

In [65]:
```
print(m and n)
```

```
False
```

In [66]:
```
# OR Operator  syntax: or
```

In [ ]:
```
# T T = T
# T F = T
# F T = T
# F F = F
```

In [71]:
```
print(m or n)
```

```
True
```

In [ ]:
```
# NOT operator  syntax: not
```

In [76]:
```
print(not m)
```

```
False
```

Bitwise operators

In [79]:
```
# Bitwise AND  syntax: &
```

In [77]:
```
a = 4  # binary  = 0100
b = 10 # binary = 1010
```

In [78]:
```
# 0     1    0    0
# |     |    |    |
# 2^3   2^2 2^1 2^0
```

In [81]:
```
b & a

# internal calculation : multiplication bit by bit

# 1 0 1 0
# 0 1 0 0
```

Out[81]: 0

In [82]:
```
p = 3  # binary  = 0011
q = 10 # binary = 1010
```

In [83]:
```
# internal calculation : multiplication bit by bit
p & q
# 0 0 1 1
# 1 0 1 0
# after multiply bit by bit output = 0 0 1 0 (binary)
# 0      0    1     0
# |      |    |     |
# 2^3   2^2  2^1   2^0


# decimal of 0010 = 2
```

Out[83]: 2

In [ ]:
```
# Bitwise OR  syntax: |
```

In [84]:
```
x = 4  # binary  o 0100
y = 10 # binary  o 1010
```

In [85]:
```
# internal calculation : addition bit by bit
x | y

# 0 1 0 0
# 1 0 1 0

# after adding bit by bit output = 1 1 1 0 (binary)
# 8+4+2+0
```

Out[85]: 14

In [89]:
```
# Bitwise not syntax : ~

s = 10 # 1010

# -1010
# -(1010+1)= -(1011) = -(11)

~s

```

Out[89]: -11

In [90]:
```
~4 # 0100
# one's complement = (0100+1) = -(0101)
```

Out[90]: -5

In [ ]:
```

```

In [92]:
```
# Bitwise left shift

l = 5 # binary (0000 0101)
```

In [93]:
```
l << 1 # (0000 1010)  = 10
```

Out[93]: 10

In [94]:
```
l<<2 #(0001 0100)

# 0   0    0    1   0   1    0   0
#2^7  2^6  2^5 2^4 2^3 2^2 2^1 2^0

#0+0+0+16+0+4+0+0
```

Out[94]: 20

IF

In [ ]:
```
# if evaluates either true or false
```

In [4]:
```
# Lets check a particular number is even or odd

number = int(input("Enter a number : "))

if number%2 == 0:
    print("The number is even")

```

```
Enter a number : 6
The number is even
```

else

In [3]:
```
number = int(input("Enter a number : "))

if number%2 == 0:
    print("The number is even")
else:
    print("The number is odd")
```

```
Enter a number : 13
The number is odd
```

if elif else

In [5]:
```
# ask two numbers from user, find which number is greater,small
```

In [9]:
```
number_1 = int(input("Enter 1st number : "))
number_2 = int(input("Enter 2nd number : "))

if number_1 == number_2:
    print("Both the numbers are equal")
elif number_1>number_2:
    print("your 1st number is greater")
else:
    print("Your 2nd number is greater")
```

```
Enter 1st number : 30
Enter 2nd number : 40
Your 2nd number is greater
```

In [10]:
```
# if-elif ladder
```

In [21]:
```
marks = int(input("enter yout marks"))

if marks<0:
    print("entered marks are invalid")
elif marks>100:
    print("entered marks are invalid")
elif marks<35:
    print("Try again")
elif marks>=35 and marks<50:
    print("you got 2nd class")
elif marks>=50 and marks<70:
    print("you got 1st class")
elif marks>=70 and marks<90:
    print("you got distinction")
elif marks>=90 and marks<=100:
    print("Meritttttttt. ..... ")
```

```
enter yout marks67
you got 1st class
```

In [22]:
```
# Nested if
```

In [ ]:
```
# within one if we may have another if
```

In [29]:
```
num = int(input("enter a number : "))

if num>= 0:

    if num == 0:
        print("Its a zero")
    else:
        print("Its a positive number")
else:
    print("its a negative number")
```

```
enter a number : -10
its a negative number
```

In [31]:
```
# Account balance check algorithm
```

In [ ]:
```
# card number
# pin
# print balance
```

In [34]:

```
card = int(input("enter your card number: "))

balance = 100000

if card == 1234:
    pin = int(input("enter your pin number: "))
    if pin == 7777:
        print("Your account balance is : ",balance)
    else:
        print("You have entered wrong PIN")
else:
    print("You have entered wrong Card number ")
```

```
enter your card number: 1234
enter your pin number: 7777
Your account balance is :  100000
```

For loop

In [4]:
```
for i in range(0,6):
    print(i)
```

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

In [ ]:
```
# range(0,6) = 0,1,2,3,4,5
```

In [2]:
```
# range(a,b,c)
# a = starting point
# b = ending point (its always consider b-1 )
# c = step size
```

In [6]:
```
for i in range(0,6,2):
    print(i)
```

```
0
2
4
```

In [7]:
```
for i in "python":
    print(i)
```

```
p
y
t
h
o
n
```

In [10]:
```
l = [1,2,3,4,5]
for x in l:
    sq = x*x
    print(sq)
```

```
1
4
9
16
25
```

if else in for loop

In [11]:
```
lst = [6,7,8,9,10]
for i in lst:
    if i == 9:
        print("yessss i got it..... ")
    else:
        print("keep moving")
```

```
keep moving
keep moving
keep moving
yessss i got it.....
keep moving
```

Nested for loop

In [15]:
```
fruits = ["apple","cherry","orange"]
adj = ["red","big","tasty"]

for i in adj:
    for j in fruits:
        print(i,j)
```

```
red apple
red cherry
red orange
big apple
big cherry
big orange
tasty apple
tasty cherry
tasty orange
```

While loop

In [27]:
```
x = 45

while x!=0:
    print("condition satisfied, you are in while loop")
    x = x//10
    print(x)

print("----------------------- ")
print("condition breaked , you are out of loop")
```

```
condition satisfied, you are in while loop
4
condition satisfied, you are in while loop
0
-----------------------
condition breaked , you are out of loop
```

Break, continue and Pass

In [28]:
```
# break : it stops the execution of code if condition satisfied
```

In [31]:
```
for i in range(1,10):
    if i==5:
        break
    print(i)
```

```
1
2
3
4
```

In [ ]:
```
# continue : it continues the execution of code except for cond
```

In [32]:
```
for i in range(1,10):
    if i==5:
        continue
    print(i)
```

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

In [33]:
```
# pass
```

In [39]:
```
for i in range(1,10):
    if i==5:
        pass
    print(i)
```

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

In [ ]:
```

```

In [ ]:
```

```

List

In [6]:
```
course = ['python','java','R','c++']
```

In [7]:
```
type(course)
```

Out[7]:
```
list
```

In [8]:
```
l = len(course)
print("lenght of my list is : ",l)
```

```
lenght of my list is :  4
```

In [9]:
```
l1 = ["python",5,89,4.5]
type(l1)
```

Out[9]:
```
list
```

In [10]:
```
# access an element out of list
```

In [19]:
```
course = ['python','java','R','c++']
```

In [12]:
```
course[2]
```

Out[12]:
```
'R'
```

In [13]:
```
# list[a:b:c]  a = starting index, b = ending index (b-1), c =
course[0:3]
```

Out[13]:
```
['python', 'java', 'R']
```

In [18]:
```
course[0:3:2]
```

Out[18]:
```
['python', 'R']
```

In [20]:
```
course[0:4:3]
```

Out[20]:
```
['python', 'c++']
```

In [ ]:
```

```

In [21]:
```
fruits = ['apple','grapes','mango']
```

In [22]:
```
# append() : Adds an item to end of the list

fruits.append('orange')
fruits
```

Out[22]:
```
['apple', 'grapes', 'mango', 'orange']
```

In [23]:
```
print(fruits)
```

```
['apple', 'grapes', 'mango', 'orange']
```

In [24]:
```
# insert(i,x): insert an item at given position (i = index, x =
fruits.insert(1,"papaya")
print(fruits)
```

```
['apple', 'papaya', 'grapes', 'mango', 'orange']
```

In [25]:
```
# extend() : we can append another list to main list using exte
citrus = ['orange','lime','grapes']
berries = ['strawberry','raspberry','blueberry']
citrus.extend(berries)
print(citrus)
```

```
['orange', 'lime', 'grapes', 'strawberry', 'raspberry', 'blueberry
']
```

In [26]:
```
# + operator
citrus = ['orange','lime','grapes']
berries = ['strawberry','raspberry','blueberry']
fruits = citrus + berries
print(fruits)
```

```
['orange', 'lime', 'grapes', 'strawberry', 'raspberry', 'blueberry
']
```

In [ ]:
```
# remove(x) :removes an element which is defined within functio
```

In [27]:
```
fruits.remove('lime')
print(fruits)
```

```
['orange', 'grapes', 'strawberry', 'raspberry', 'blueberry']
```

In [ ]:
```
# pop(i) : removes an element of which index is defined
# but if nothing defined within pop, it removes last element
```

In [28]:
```
fruits.pop()
print(fruits)
```

```
['orange', 'grapes', 'strawberry', 'raspberry']
```

In [29]:
```
# index() : returns location of particular element
# index(x,y,z) ----- x = element, y = starting index, z = endin
fruits.index('strawberry')
```

Out[29]: 2

In [32]:
```
fruits.index('strawberry',0,4)
```

Out[32]: 2

In [33]:
```
fruits.append('strawberry')
fruits
```

Out[33]: ['orange', 'grapes', 'strawberry', 'raspberry', 'strawberry']

In [36]:
```
# count(x) : gives count of a particular element

fruits.count('strawberry')
```

Out[36]: 2

In [ ]:
```
# sort : sorting of given list
```

In [37]:
```
fruits.sort()
print(fruits)
```

['grapes', 'orange', 'raspberry', 'strawberry', 'strawberry']

In [38]:
```
# reverse() : it converts list to current reverse version of it

fruits.reverse()
print(fruits)
```

['strawberry', 'strawberry', 'raspberry', 'orange', 'grapes']

In [39]:
```
# copy() = returns copy of the list

fruits_1 = fruits.copy()
print(fruits_1)
```

['strawberry', 'strawberry', 'raspberry', 'orange', 'grapes']

In [40]:
```
# clear() = removes all element from the list

fruits_1.clear()
print(fruits_1)
```

[]

In [41]:
```
fruits
```

Out[41]: ['strawberry', 'strawberry', 'raspberry', 'orange', 'grapes']

In [ ]:
```

```

List comprehension

In [52]:
```
# its a one liner solution on list
```

In [54]:
```
# normal method
new_list = []
for i in range(0,10):
    new_list.append(i)
print(new_list)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [55]:
```
new_list_1 = [i for i in range(0,10)]
print(new_list_1)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [56]:
```
lst = [1,2,3,4,5,6,7,8,9]

# i want to find numbers greater than 5
```

In [57]:
```
#normal method
blank_list = []
for i in lst:
    if i>5:
        blank_list.append(i)
print(blank_list)
```

```
[6, 7, 8, 9]
```

In [58]:
```
# using list comprehension

abc = [i for i in lst if i > 5]
print(abc)
```

```
[6, 7, 8, 9]
```

In [60]:
```
# using list comprehension

xyz = [i if i%2==0 else 0 for i in lst]
print(xyz)
```

```
[0, 2, 0, 4, 0, 6, 0, 8, 0]
```

In [65]:
```

b = [i if i>5 else 0 for i in lst]
print(b)
```

```
[0, 0, 0, 0, 0, 6, 7, 8, 9]
```

In [ ]:
```
empdata = ('rudra',20,'IT',50000)
```

In [ ]:
```
type(empdata)
```

In [ ]:
```
empdata[3] # slicing ------ always use []
```

In [ ]:
```
# len() : number of elements in my tuple

l = len(empdata)
print(l)
```

In [ ]:
```
empdata[0:2] # 0,1
```

In [ ]:
```
empdata[0:4:2]
```

In [ ]:
```
# for single element tuple "," is must
tup = ('avengers',)
```

In [ ]:
```
type(tup)
```

In [ ]:
```
tup1 = tuple('car')
type(tup1)
```

In [ ]:
```
# count : counts occurences of particular element

empdata.count('rudra')
```

In [ ]:
```
# index : gives location of particular element

empdata.index(50000)
```

In [1]:
```
 fruits = ('orange','lime','grapes')
```

In [2]:
```
fruits.remove('orange') # tuple is immutalbe
```

```
---------------------------------------------------------------------
----------
AttributeError                            Traceback (most recent c
all last)
<ipython-input-2-7ba8c4dcc0eb> in <module>
----> 1 fruits.remove('orange') # tuple is immutalbe

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

In [ ]:
```
# to change tuple value, convert it to list and then do it
```

In [3]:
```
fruits_lst = list(fruits)
print(fruits_lst)
```

```
['orange', 'lime', 'grapes']
```

In [4]:
```
fruits_lst.remove('orange')
print(fruits_lst)
```

```
['lime', 'grapes']
```

In [5]:
```
fruits_tup = tuple(fruits_lst)
print(fruits_tup)
```

```
('lime', 'grapes')
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```
# add
```

In [1]:
```
fruits = {'apple','grapes','mango'}
```

In [2]:
```
type(fruits)
```

Out[2]: set

In [6]:
```
fruits.add('orange')
print(fruits)
```

```
{'orange', 'apple', 'mango', 'grapes'}
```

In [7]:
```
# discard()
fruits.discard("mango")
print(fruits)
```

```
{'orange', 'apple', 'grapes'}
```

In [13]:
```
# union
fruits = {'apple','grapes','mango'}
color = {'red','green','yellow','mango'}

final = fruits.union(color)
print(final)
```

```
{'apple', 'red', 'yellow', 'mango', 'grapes', 'green'}
```

In [21]:
```
# intersection
fruits = {'apple','grapes','mango'}
color = {'red','green','yellow','mango'}

final1 = fruits.intersection(color)
print(final1)
```

```
{'mango'}
```

In [23]:
```
# difference
fruits = {'apple','grapes','mango'}
color = {'red','green','yellow','mango'}

final2 = fruits.difference(color)
print(final2)

```

```
{'apple', 'grapes'}
```

In [39]:
```
# comparision

set1 = {1,2}
set2 = {1,2,3}

set1<set2
```

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

In [27]:
```
# sorted

color = {'red','green','yellow','mango'}
sorted_color = sorted(color)
print(sorted_color)
```

```
['green', 'mango', 'red', 'yellow']
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [1]:
```
dept_head = {"HR":'rajeev','AC':'ramesh','IT':'raman'}
type(dept_head)
```

Out[1]:
```
dict
```

In [2]:
```
# accessing an elements from dictionary
```

In [6]:
```
dept_head['HR'] # call dictionary elemnts using key name
```

Out[6]:
```
'rajeev'
```

In [ ]:
```

```

In [7]:
```
numbers = {1:'one',2:'two',3:'three',4:'four',5:'five'}
```

In [8]:
```
# keys() = returns list of all keys present in my dictionary
numbers.keys()
```

Out[8]:
```
dict_keys([1, 2, 3, 4, 5])
```

In [9]:
```
# values() = returns list of all values present in my dictionar
numbers.values()
```

Out[9]:
```
dict_values(['one', 'two', 'three', 'four', 'five'])
```

In [10]:
```
# items() = returns list of tuple of key value pairs present in


numbers.items()
```

Out[10]:
```
dict_items([(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four'), (5,
'five')])
```

In [11]:
```
# update() = we can add new key value pair to our existing dict
# ata time we can add multiple elements
numbers.update({6:'six'})
numbers
```

Out[11]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'six'}
```

In [14]:
```
# we can add a new element to dictionary by using new index[new
# one by one
numbers[7] = 'seven'
numbers
```

Out[14]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'six', 7
: 'seven'}
```

In [17]:
```
# pop(n) = remove any dictionary item by specifying its key(n)

numbers.pop(1)
```

Out[17]:
```
'one'
```

In [18]:
```
numbers
```

Out[18]:
```
{2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'six', 7: 'seven'}
```

In [19]:
```
# popitem = removes last item from dictionary

numbers.popitem()
numbers
```

Out[19]:
```
{2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'six'}
```

In [44]:
```
# get() = accessing elements

numbers.get(2)
```

Out[44]:
```
'two'
```

In [32]:
```
# adding multiple elements in dictionary
```

In [33]:
```
flowers = {}
```

In [34]:
```
flowers[2] = 'rose'
flowers[1] = 'sunflower'
```

In [35]:
```
flowers
```

Out[35]:
```
{2: 'rose', 1: 'sunflower'}
```

In [36]:
```
flowers[3] = 'lily'
flowers[4] = 'lotus'
```

In [37]:
```
flowers
```

Out[37]:
```
{2: 'rose', 1: 'sunflower', 3: 'lily', 4: 'lotus'}
```

In [38]:
```
flowers[3] = 'jasmin'
```

In [39]:
```
flowers
```

Out[39]:
```
{2: 'rose', 1: 'sunflower', 3: 'jasmin', 4: 'lotus'}
```

In [ ]:
```

```

In [45]:
```
# dictionary comprehension
```

In [47]:
```
# Normal method

sq_dict = dict()
for i in range(1,11):
    sq_dict[i] = i*i

print(sq_dict)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 1
00}
```

In [51]:
```
# dictionary comprehension
square_dict = {i:i*i for i in range(1,11)}
square_dict
```

Out[51]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 1
00}
```

In [ ]:
```

```

In [72]:
```
keys = [1,2,3,4,5]
values = ['abc','def','ghi','jkl','mno']
```

In [73]:
```
combined = {x:y for (x,y) in zip(keys,values)}
combined
```

Out[73]:
```
{1: 'abc', 2: 'def', 3: 'ghi', 4: 'jkl', 5: 'mno'}
```

In [ ]:
```

```

In [76]:
```
l1 = [1,2,3]
l2 = [4,5,6]
l3 = []
for i,j in zip(l1,l2):
    l3.append(i*j)
l3
```

Out[76]:
```
[4, 10, 18]
```

In [81]:
```
l4 = []
for i in range(len(l1)):
    a = l1[i]*l2[i]
    l4.append(a)

l4
```

Out[81]:
```
[4, 10, 18]
```

functions

In [1]:
```
# definition of function

def sayhi():
    print("hiiii hello everyone welcome to my session ")
```

In [3]:
```
# calling a function
sayhi()
```

```
hiiii hello everyone welcome to my session
```

In [13]:
```
# generalised function with user input
def addition(n1,n2):
a = n1+n2
    return a
```

In [14]:
```
# calling a function

result = addition(20,250)
print(result)
```

```
270
```

In [16]:
```
# function parameter order

def  areaofcircle(radius,pi):
    return  (pi*radius**2)
```

In [17]:
```
# calling a function  using proper order
areaofcircle(10,3.14)
```

Out[17]:
```
314.0
```

In [18]:
```
# calling a function  using wrong order
areaofcircle(3.14,10)
```

Out[18]:
```
98.596
```

In [19]:
```
# using default arguements

def areacircle(radius,pi = 3.14):
    return  (pi*radius**2)
```

In [20]:
```
# calling
areacircle(10)
```

Out[20]:
```
314.0
```

In [32]:
```
# parameters vs arguments :
# parameters are used while defining function
# arguments are used while calling function
```

In [30]:
```
# odd even

def iseven(num):
    if num%2==0:
        return True
    else:
        return False
```

In [31]:
```
iseven(10)
```

Out[31]:
```
True
```

In [ ]:
```
# num : parameter
# 10 : argument
```

In [33]:
```
# docstring : add comments / documentations to your function
# it is written in triple quotes

def iseven(num):
    """Return true if number is even else return false."""
    if num%2==0:
        return True
    else:
        return False
```

In [34]:
```
# calling doc string from defined function

iseven.__doc__
```

Out[34]:
```
'Return true if number is even else return false.'
```

In [ ]:
```

```

In [35]:
```
# variable lenght of arguments
```

In [36]:
```
def add(*args):
    return sum(args)
```

In [38]:
```
add(1,3,5,10)
```

Out[38]:
```
19
```

In [50]:
```
def getmax(*args):
    return max(args)
```

In [52]:
```
getmax(1,7,89,45)
```

Out[52]: 89

In [53]:
```
# global Vs local variable
```

In [54]:
```
var = 1 # global varibale

def ad(*nums):
    # total = local variable
    total = 0
    for i in nums:
        total = total + i
    return total
```

In [58]:
```
ad(1,2,3,4,5,6)
```

Out[58]: 21

In [56]:
```
print(var) # global
```

```
1
```

In [57]:
```
print(total) # local
```

```
---------------------------------------------------------------------
----------
NameError                                 Traceback (most recent c
all last)
<ipython-input-57-517c68ee957b> in <module>
----> 1 print(total)

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

In [ ]:
```

```

lambda

In [59]:
```
# find cube of number using normal def(function)

def cube(n):
    return (n*n*n)
```

In [60]:
```
cube(2)
```

Out[60]: 8

In [65]:
```
# find cube of number using lamda
g = lambda x:x*x*x
```

In [68]:
```
g(3)
```

Out[68]: 27

In [70]:
```
# square rot of number using lambda
(lambda a:a**0.5)(9)
```

Out[70]: 3.0

In [ ]:
```

```

In [71]:
```
def agecal(yob):
    return (2022-yob)
```

In [72]:
```
agecal(1990)
```

Out[72]: 32

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

functions

In [ ]:
```
# definition of function

def sayhi():
    print("hiiii hello everyone welcome to my session ")
```

In [ ]:
```
# calling a function
sayhi()
```

In [ ]:
```
# generalised function with user input
def addition(n1,n2):
a = n1+n2
    return a
```

In [ ]:
```
# calling a function

result = addition(20,250)
print(result)
```

In [ ]:
```
# function parameter order

def areaofcircle(radius,pi):
    return (pi*radius**2)
```

In [ ]:
```
# calling a function  using proper order
areaofcircle(10,3.14)
```

In [ ]:
```
# calling a function  using wrong order
areaofcircle(3.14,10)
```

In [ ]:
```
# using default arguements

def areacircle(radius,pi = 3.14):
    return (pi*radius**2)
```

In [ ]:
```
# calling
areacircle(10)
```

In [ ]:
```
# parameters vs arguments :
# parameters are used while defining function
# arguments are used while calling function
```

In [ ]:
```python
# odd_even

def iseven(num):
    if num%2==0:
        return True
    else:
        return False
```

In [ ]:
```python
iseven(10)
```

In [ ]:
```python
# num : parameter
# 10 : argument
```

In [ ]:
```python
# docstring : add comments / documentations to your function
# it is written in triple quotes

def iseven(num):
    """Return true if number is even else return false."""
    if num%2==0:
        return True
    else:
        return False
```

In [ ]:
```python
# calling doc string from defined function

iseven.__doc__
```

In [ ]:
```python

```

In [ ]:
```python
# variable lenght of arguments
```

In [ ]:
```python
def add(*args):
    return sum(args)
```

In [ ]:
```python
add(1,3,5,10)
```

In [ ]:
```python
def getmax(*args):
    return max(args)
```

In [ ]:
```python
getmax(1,7,89,45)
```

In [ ]:
```python
# global Vs local variable
```

In [ ]:
```
var = 1 # global varibale

def ad(*nums):
    # total = local variable
    total = 0
    for i in nums:
        total = total + i
    return total
```

In [ ]:
```
ad(1,2,3,4,5,6)
```

In [ ]:
```
print(var) # global
```

In [ ]:
```
print(total) # local
```

In [ ]:
```

```

lambda

In [ ]:
```
# find cube of number using normal def(function)

def cube(n):
    return (n*n*n)
```

In [ ]:
```
cube(2)
```

In [ ]:
```
# find cube of number ı ing lamda
g = lambda x:x*x*x
```

In [ ]:
```
g(3)
```

In [ ]:
```
# square rot of number using lambda
(lambda a:a**0.5)(9)
```

In [ ]:
```

```

In [ ]:
```
def agecal(yob):
    return (2022-yob)
```

In [ ]:
```
agecal(1990)
```

In [ ]:
```

```

map()

In [ ]:
```
# give new alist of numbers, return new list with squares of ea
# normal method
nums = [1,2,3,4,5,6,7,8,9]
sq_lst = []

for i in nums:
    a = i**2
    sq_lst.append(a)

print(sq_lst)
```

In [ ]:
```
# map() : takes function and list as an input

nums = [1,2,3,4,5,6,7,8,9]
sq = list(map(lambda x:x**2,nums))
```

In [ ]:
```
sq
```

In [ ]:
```

```

reduce()

In [3]:
```
#reduce :it reduces input list using our logic , it takes funct

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

import functools
addition = functools.reduce((lambda x,y:x+y),nums)
print(addition)
```

```
45
```

filter()

In [12]:
```
# filter : it filters from list using defined condition

nums = [1,2,3,4,5,6,7,8,9]
evens = list(filter((lambda x:x%2==0),nums))
print(evens)
```

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

In [ ]:
```

```

In [13]:
```
names = ['ramesh','suresh','smith','john','nasar']
names_up = list(map(str.upper,names))
print(names_up)
```

```
['RAMESH', 'SURESH', 'SMITH', 'JOHN', 'NASAR']
```

In [13]:
```
names = ['ramesh','suresh','smith','john','nasar']
names_up = list(map(str.upper,names))
print(names_up)
```

```
['RAMESH', 'SURESH', 'SMITH', 'JOHN', 'NASAR']
```

In [ ]:
```

```

In [ ]:
```

```

Decorator ¶

In [17]:
```
def welcome(msg):
    print(msg)
```

In [19]:
```
welcome("Hiiii how are u")
```

```
Hiiii how are u
```

In [20]:
```
greeting = welcome
greeting("hello hiiii")
```

```
hello hiiii
```

In [ ]:
```

```

In [21]:
```
# calling function within function

def inc5(x):
    return (x+5)

def dec5(x):
    return (x-5)

def process(func,x):
    res = func(x)
    return res
```

In [22]:
```
process(inc5,100)
```

Out[22]:
```
105
```

In [23]:
```
process(dec5,37)
```

Out[23]:
```
32
```

In [ ]:
```

```

```
i am normal function
```

In [ ]:
```
```

In [42]:
```
def divide(x,y):
    return x/y
```

In [43]:
```
divide(10,5)
```

Out[43]:
```
2.0
```

In [44]:
```
divide(10,0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call
<ipython-input-44-101aa8dc6e2a> in <module>
----> 1 divide(10,0)

<ipython-input-42-cde09cc5b8ef> in divide(x, y)
      1 def divide(x,y):
----> 2     return x/y

ZeroDivisionError: division by zero
```

In [47]:
```
def divide_smartly(func):
    def condition(a,b):
        print("i am performing division")
        if b == 0:
            print("you are trying to devide by zero !!!!")
            return
        return func(a,b)
    return condition
```

In [48]:
```
@divide_smartly
def divide(a,b):
    return a/b
```

In [49]:
```
divide(10,5)
```

```
i am performing division
```

Out[49]:
```
2.0
```

In [50]:
```
divide(10,0)
```

```
i am performing division
you are trying to devide by zero !!!!
```

In [ ]:
```
```

In [ ]:
```
```

error handling

In [1]:
```
# try: (try block lets you test block of code)

# except: (lets you handle error)

# else: (optional)

# finally: (lets you execute code,it always runs irrespective o
```

In [2]:
```
try:
    print(x)
except:
    print("something went wrong")
finally:
    print("try except block is finished, its done")
```

```
something went wrong
try except block is finished, its done
```

In [10]:
```
y = 10

try:
    print(y)
except:
    print("something went wrong")
finally:
    print("try except block is finished, its done")
```

```
10
try except block is finished, its done
```

In [13]:
```
try:
    value = int(input("type a number between 1 and 10 : "))
except:
    print("you must enter a digit !!!!!")
else:
    if (value>0) and (value<=10):
        print("you entered : ", value)
    else:
        print("please eneter number within given range ........
```

```
type a number between 1 and 10 : 7
you entered : 7
```

In [ ]:
```

```

In [17]:
```
def div(a,b):
    try:
        output = a/b
        print("division output is :",output)
    except ZeroDivisionError:
        print("you are trying to divide by 0")
```

In [19]:
```
div(10,'b')
```

```
you entered wrong type
```

In [16]:
```
div(10,0)
```

```
you are trying to divide by 0
```

In [ ]:
```

```

In [ ]:
```

```

In [1]:
```
# write to file

file = open("test.txt",'w')
file.write("Hello everyone, welcome to python coding.....")
file.close()
```

In [2]:
```
# read from file

file = open("test.txt",'r')
file_data = file.read()
print(file_data)
```

```
Hello everyone, welcome to python coding.....
```

In [3]:
```
file.close() # close the file
```

In [4]:
```
# append to file

file = open("test.txt",'a')
file.write("python has many application and it is widely used !
file.close()
```

In [6]:
```
# writelines

file = open("test.txt",'a')
file.writelines(['\ni have python version 3.8','\ni am using ju
file.close()
```

In [7]:
```
file = open("test.txt",'r')
print(file.read())
```

```
Hello everyone, welcome to python coding.....python has many appli
cation and it is widely used !!
i have python version 3.8
i am using jupyter notebook
```

In [ ]:
```

```

 # Created by sakshi sunil bhingarde