- python basic syntax
  - basic codes
  - data types
  - print statements
- concept realted to some packages
  - how to read the package
  - how to explore the package
- conditional statements

Functions

it is block of code have set of instructions, and we can recall multiple time

In [2]:
```
a=20
b=30
print(a+b)
```

```
50
```

With out arguments

In [ ]:
```
def <function_name>():
    # statement-1
    # statement-2
    #
    #
```

In [3]:
```
# i want to create a function with name add
def add():
    a=20
    b=30
    print(a+b)

# if you run the function it will not return any output
# untill unless you call the function
```

In [4]:
```
add()
```

```
50
```

In [5]:
```
# i want to do an operation of adding a base salary and DA amount
# total salary

# write the normal code
base_salary=eval(input("enter base salary:"))
DA_amount=eval(input("enter DA amount:"))
total_salary=base_salary+DA_amount
print("The total salary is:",total_salary)
```

```
enter base salary:50000
enter DA amount:10000
The total salary is: 60000
```

In [9]:
```
def salary():
    base_salary=eval(input("enter base salary:"))
    DA_amount=eval(input("enter DA amount:"))
    total_salary=base_salary+DA_amount
    print("The total salary is:",total_salary)

# it will not show any error
# it will not give any output also
```

In [10]:
```
salary()
```

```
enter base salary:60000
enter DA amount:15000
The total salary is: 75000
```

In [12]:
```
# Wap ask the user enter 3 numbers and calculate average (with out function)
# implement function on this code
num1=eval(input("enter number1:"))
num2=eval(input("enter number2:"))
num3=eval(input("enter number3:"))
avg=(num1+num2+num3)/3
print("Average is:",avg)
```

```
enter number1:10
enter number2:20
enter number3:30
Average is: 20.0
```

In [15]:
```
def avg():
    num1=eval(input("enter the number"))
    num2=eval(input("enter the number"))
    num3=eval(input("enter the number"))
    avg=(num1+num2+num3)/3
    print("avg of numbers are",avg)
```

In [16]:
```
avg()
```

```
enter the number10
enter the number20
enter the number30
avg of numbers are 20.0
```

In [ ]:
```
def avg():
    try:
        num1=eval(input("enter the number"))
        num2=eval(input("enter the number"))
        num3=eval(input("enter the number"))
        avg=(num1+num2+num3)/3
        print("avg of numbers are",avg)

    except Exception as e:
        print(e)
```

In [17]:
```
# wap ask the user enter bill amount
#                   enter how much tip you ant to pay
# then calculate total bill

# implement the function

# Method-1: Normal way
bill_amount=eval(input("enter the bill amount"))
tip_amount=eval(input("enter the tip amount"))
total_amount=bill_amount+tip_amount
print("total amount to pay:",total_amount)
```

```
enter the bill amount1000
enter the tip amount50
total amount to pay: 1050
```

In [18]:
```
# Method-2:
def bill():
    bill_amount=eval(input("enter the bill amount"))
    tip_amount=eval(input("enter the tip amount"))
    total_amount=bill_amount+tip_amount
    print("total amount to pay:",total_amount)

bill()
```

```
enter the bill amount1000
enter the tip amount40
total amount to pay: 1040
```

In [19]:
```
# Method-3:
def bill():
    try:
        bill_amount=eval(input("enter the bill amount"))
        tip_amount=eval(input("enter the tip amount"))
        total_amount=bill_amount+tip_amount
        print("total amount to pay:",total_amount)

    except Exception as e:
        print(e)

bill()
```

```
enter the bill amount2000
enter the tip amount20
total amount to pay: 2020
```

In [ ]:
```
# wap ask the user enter a number
# print if it is an even number or odd number

# implement a function on this

num=eval(input("enter a num"))
if num%2==0:
    print("it is a even number")
else:
    print("it is odd")
```

In [22]:
```
def even_odd():
    num=eval(input("enter a num"))
    if num%2==0:
        print("it is a even number")
    else:
        print("it is odd")

even_odd()
```

```
enter a num91
it is odd
```

In [ ]:
```
1) add()   # not provided anything inside the bracket (with out arguments)
2) bill()  # not provided anything inside the bracket (with out argumnets)
3) avg()
4) salary()
5) even_odd()

# If we provide any thing inside the brackets
# that are called arguments/ parameters
```

In [23]:
```
import random
random.randint()
```

With arguments

In [24]:
```
def add():
    a=20
    b=30
    print(a+b)

add()
```

```
50
```

In [26]:
```
def add(a):
    b=30
    print(a+b)

add(100)  # a=100
```

```
130
```

In [31]:
```
def add(a):
    b=eval(input("enter number b"))
    print(a+b)

add(100)
```

```
enter number be400
500
```

In [27]:
```
def add(a,b):  # while intializing you given two arguments
    print(a+b)

add(100)  # a=100  but while calling me , you provided only one arguments
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[27], line 4
      1 def add(a,b):
      2     print(a+b)
----> 4 add(100)

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

In [29]:
```
def add(b,a):  # while intializing you given two arguments
    print(a+b)

add(100)  #
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[29], line 4
      1 def add(b,a):  # while intializing you given two arguments
      2     print(a+b)
----> 4 add(100)

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

In [30]:
```
def add(a,b):
    print(a)
    print(b)
    print(a+b)

add(100,300)  # a=100 b=300

# how many variables you provide
```

```
100
300
400
```

In [32]:
```
def add(a,b):
    print(a+b)

add()

# error:
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[32], line 4
      1 def add(a,b):
      2     print(a+b)
----> 4 add()

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

In [34]:
```
def add():
    print(x+y)

add()
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[34], line 4
      1 def add():
      2     print(x+y)
----> 4 add()

Cell In[34], line 2, in add()
      1 def add():
----> 2     print(x+y)

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

In [35]:
```
def add(x, y):
    print(x + y)

add(20,50)
```

```
70
```

In [36]:
```
def add(x, y):
    x = 50
    x =150
    print(x + y)

add(20)  # 20 is x   first in first out

# 20 is x
# what is the final value of x ?
```

```
550
```

In [ ]:
```
# With out arguments
def salary():
    base_salary=eval(input("enter base salary:"))
    DA_amount=eval(input("enter DA amount:"))
    total_salary=base_salary+DA_amount
    print("The total salary is:",total_salary)
```

In [37]:
```
def salary(base_salary,DA_amount):
    total_salary=base_salary+DA_amount
    print("The total salary is:",total_salary)

salary(50000,10000)
```

```
The total salary is: 60000
```

In [ ]:
```
def avg():
    try:
        num1=eval(input("enter the number"))
        num2=eval(input("enter the number"))
        num3=eval(input("enter the number"))
        avg=(num1+num2+num3)/3
        print("avg of numbers are",avg)

    except Exception as e:
        print(e)
```

In [ ]:
```
Question: Sir what is the difference between With and Without Arguments?
Which one is good to use in real time?
```

In [38]:
```
def avg(n1,n2,n3):
    avg=(n1+n2+n3)/3
    print("avg is:",avg)

avg(10,20,30)
```

```
avg is: 20.0
```

- With out arguments
- with arguments
- default arguments

In [42]:
```
def avg(n1,n2,n3=500):
    avg=(n1+n2+n3)/3
    print("avg is:",avg)

avg(10,20)

# fixed means default
# which argument is fixed that is called default argumnets
# here n3 is the default argument
```

```
avg is: 176.66666666666666
```

In [43]:
```
# 1) verification

def avg(n1,n2,n3=500):
    print("n1:",n1)
    print("n2:",n2)
    print("n3:",n3) # 30
    avg=(n1+n2+n3)/3
    print("avg is:",avg)

avg(10,20,30)

# intialization is first   ============ n3=500
# or calling is first      ============ n3=30
# latset value n3=30
```

```
n1: 10
n2: 20
n3: 30
avg is: 20.0
```

In [49]:
```
def avg(n1,n2,n3=500):
    n3=1000
    print("n1:",n1)  # 10
    print("n2:",n2)  # 20
    print("n3:",n3)  # 1000
    avg=(n1+n2+n3)/3
    print("avg is:",avg)



# intialization is first   ============ n3=500
# or calling is first      ============ n3=30
# latset value n3=30
#
```

In [50]:
```
avg(10,20)
```

```
n1: 10
n2: 20
n3: 1000
avg is: 343.3333333333333
```

In [45]:
```
def avg(n1,n2,n3=500):
    n3=1000
    print("n1:",n1)  # 10
    print("n2:",n2)  # 20
    print("n3:",n3)  # ?
    avg=(n1+n2+n3)/3
    print("avg is:",avg)

avg(10,20,30)

# step-1:  what intilaztion
# step-2: calling
# step-3: inside the function
```

```
n1: 10
n2: 20
n3: 1000
avg is: 343.3333333333333
```

In [46]:
```
def avg(n1,n2=1000,n3):
    print("n1:",n1)
    print("n2:",n2)
    print("n3:",n3)
    avg=(n1+n2+n3)/3
    print("avg is:",avg)

avg(10,20)

# always default parameters at last
```

```
  Cell In[46], line 1
    def avg(n1,n2=1000,n3):
                       ^
SyntaxError: non-default argument follows default argument
```

In [47]:
```
def avg(n1,n3,n2=1000):
    print("n1:",n1)   # 10
    print("n2:",n2)   # 1000
    print("n3:",n3)   # 20
    avg=(n1+n2+n3)/3
    print("avg is:",avg)

avg(10,20)
```

```
n1: 10
n2: 1000
n3: 20
avg is: 343.3333333333333
```

In [ ]:
```
def add(x, y):
    x = 50
    x =150
    print(x + y)

add(50)  # 20 is x   first in first out
```

In [48]:
```
def avg(n1=2000,n2,n3):
    print("n1:",n1)
    print("n2:",n2)
    print("n3:",n3)
    avg=(n1+n2+n3)/3
    print("avg is:",avg)

avg(20,30)
```

```
  Cell In[48], line 1
    def avg(n1=2000,n2,n3):
                    ^
SyntaxError: non-default argument follows default argument
```

In [ ]:
```
# tax payer
# tax=10%
```

In [ ]:
```
- with out arguments

- with arguments

- default arguments
```

In [ ]:
```
# avg problem
# WAP ask the user enter 3 number and find the avergae
# 1) Normal code
# 2) with out arguments
# 3) with arguments
# 4) default arguments
```

In [1]:
```
# Method-1:
n1=eval(input("enter number1:"))
n2=eval(input("enter number2:"))
n3=eval(input("enter number3:"))
add=n1+n2+n3
avg=round(add/3,2)
print("The addition of {} {} and {} is: {}".format(n1,n2,n3,add))
print("The average of {} {} and {} is: {}".format(n1,n2,n3,avg))
```

```
enter number1:10
enter number2:20
enter number3:30
The addition of 10 20 and 30 is: 60
The average of 10 20 and 30 is: 20.0
```

In [4]:
```
# Method-2: Implement function with out arguments
def avg_with_out():
    n1=eval(input("enter number1:"))
    n2=eval(input("enter number2:"))
    n3=eval(input("enter number3:"))
    add=n1+n2+n3
    avg=round(add/3,2)
    print("The addition of {} {} and {} is: {}".format(n1,n2,n3,add))
    print("The average of {} {} and {} is: {}".format(n1,n2,n3,avg))

# above function is defined
# It will not give any output untill unless call the function
```

In [5]:
```
avg_with_out()
```

```
enter number1:20
enter number2:50
enter number3:100
The addition of 20 50 and 100 is: 170
The average of 20 50 and 100 is: 56.67
```

In [6]:
```
# Method-3: with argumnets
#In method we are asking user to enter 3 numbers
# we use these three numbers as argumnets

def avg_with_arg(n1,n2,n3):
    add=n1+n2+n3
    avg=round(add/3,2)
    print("The addition of {} {} and {} is: {}".format(n1,n2,n3,add))
    print("The average of {} {} and {} is: {}".format(n1,n2,n3,avg))
```

In [7]:
```
avg_with_arg(100,200,300) # n1=100 n2=200 n3=300
```

```
The addition of 100 200 and 300 is: 600
The average of 100 200 and 300 is: 200.0
```

In [8]:
```
# Mrthod-4: Default argumnets  n3=1000
def avg_with_default(n1,n2,n3=1000):
    add=n1+n2+n3
    avg=round(add/3,2)
    print("The addition of {} {} and {} is: {}".format(n1,n2,n3,add))
    print("The average of {} {} and {} is: {}".format(n1,n2,n3,avg))
```

In [9]:
```
avg_with_default(200,300) # n1=200 n2=300
```

```
The addition of 200 300 and 1000 is: 1500
The average of 200 300 and 1000 is: 500.0
```

In [15]:
```
def avg_with_out():
    n1=eval(input("enter number1:"))
    n2=eval(input("enter number2:"))
    n3=eval(input("enter number3:"))
    ADD=n1+n2+n3
    AVG=round(add/3,2)
    print("The addition of {} {} and {} is: {}".format(n1,n2,n3,add))
    print("The average of {} {} and {} is: {}".format(n1,n2,n3,AVG))

avg_with_out()
```

```
enter number1:20
enter number2:30
enter number3:40
The addition of 20 30 and 40 is: 60
The average of 20 30 and 40 is: 20.0
```

In [16]:
```
# I want use the avg value in future reference
# Im printing avg value outside the function
print(AVG)

# the name :  AVG is not defined before
# sir i defined inside the function
# that function not handover the value to you
# handover ======= return
# he is not return , he is just printing the value


inside defined fun will not work for outside the fun
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[16], line 3
      1 # I want use the avg value in future reference
      2 # Im printing avg value outside the function
----> 3 print(AVG)

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

In [17]:
```
print(ADD)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[17], line 1
----> 1 print(ADD)

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

In [34]:
```
def avg_with_out():
    n1=eval(input("enter number1:"))
    n2=eval(input("enter number2:"))
    n3=eval(input("enter number3:"))
    ADD11=n1+n2+n3
    AVG11=round(ADD11/3,2)
    print("The addition of {} {} and {} is: {}".format(n1,n2,n3,ADD11))
    print("The average of {} {} and {} is: {}".format(n1,n2,n3,AVG11))
    return(AVG11)


#step-1: no return syntax in your code
# step-2: change the varible name
# step-3: this function not return anything
#        but you want to use AVG11 value outside the function call
```

In [35]:
```
avg_value=avg_with_out() # this function is return one value, you need ti s
```

```
enter number1:20
enter number2:30
enter number3:40
The addition of 20 30 and 40 is: 90
The average of 20 30 and 40 is: 30.0
```

In [36]:
```
avg_value
```

Out[36]:
```
30.0
```

In [25]:
```
output1=avg_with_out()  # this function is return only one value
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[25], line 1
----> 1 output1=avg_with_out()

Cell In[24], line 2, in avg_with_out()
      1 def avg_with_out():
----> 2     n1=eval(input("enter number1:"))
      3     n2=eval(input("enter number2:"))
      4     n3=eval(input("enter number3:"))

File ~\anaconda3\Lib\site-packages\ipykernel\kernelbase.py:1202, in Kernel.raw_input(self, prompt)
   1200     msg = "raw_input was called, but this frontend does not support input requests."
   1201     raise StdinNotImplementedError(msg)
-> 1202 return self._input_request(
   1203     str(prompt),
   1204     self._parent_ident["shell"],
   1205     self.get_parent("shell"),
   1206     password=False,
   1207 )

File ~\anaconda3\Lib\site-packages\ipykernel\kernelbase.py:1245, in Kernel._input_request(self, prompt, ident, parent, password)
   1242 except KeyboardInterrupt:
   1243     # re-raise KeyboardInterrupt, to truncate traceback
   1244     msg = "Interrupted by user"
-> 1245     raise KeyboardInterrupt(msg) from None
   1246 except Exception:
   1247     self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

In [23]:
```
print(output1)
```

```
20.0
```

In [37]:
```
def avg_with_out():
    n1=eval(input("enter number1:"))
    n2=eval(input("enter number2:"))
    n3=eval(input("enter number3:"))
    ADD11=n1+n2+n3
    AVG11=round(ADD11/3,2)
    print("The addition of {} {} and {} is: {}".format(n1,n2,n3,ADD11))
    print("The average of {} {} and {} is: {}".format(n1,n2,n3,AVG11))
    return(AVG11,ADD11)
```

In [38]:
```
avg,add=avg_with_out()

# Most of the time developers use same name
```

```
enter number1:300
enter number2:300
enter number3:300
The addition of 300 300 and 300 is: 900
The average of 300 300 and 300 is: 300.0
```

In [39]:
```
print(avg,add)
```

```
300.0 900
```

In [42]:
```
def bill():
    try:
        bill_amount=eval(input("Enter the Bill amount: "))
        tip=eval(input("Enter the Tip: "))
        total_amount=bill_amount+tip
        print("The total bill amount is : ",total_amount)
        return(bill_amount,tip)
    except Exception as e:
            print(e)

return_bill_amount=bill()
return_bill_amount


# in bill function:
# total how many variables : 3
# 2 are user taken values
# 1 is calculated by python
# how many values you want return , that is your wish
```

```
Enter the Bill amount: 1000
Enter the Tip: 200
The total bill amount is :  1200
```

Out[42]:
```
(1000, 200)
```

In [ ]:
```
Sir, how we know function has two values.
So that we can write and add and avg
```

In [ ]:
```
# create a function with deafault parameter as tax_percentage=10
# enter your salary
# calaculate totl tax_amount you want to pay:  (salary*tax_per)/100
# and return tax_amount

def tax_amount(): # inside provide tax_percentage=10
    salary=eval(input("enter your salary:")) # this correct
    tax_amount=10*salary/100 #default argument tax_percentage
    print("tax amout is:",tax_amount)
    # return
```

In [ ]:
```
# wap with two a and b arguments
# return add sub  mul  divsion
```

In [43]:
```
def tax_percentage(tax_percentage=10):
    salary=eval(input("enter your salary:"))
    tax_amount=tax_percentage*salary/100
    return(tax_amount)
```

In [44]:
```
def tax_percentage(tax_percentage=10):
    salary=eval(input("enter your salary:"))
    tax_amount=tax_percentage*salary/100
    return(tax_amount)

tax_amount=tax_percentage()
print(tax_amount)
```

```
enter your salary:10000
1000.0
```

In [45]:
```
def math(a,b):
    add=a+b
    sub=a-b
    mul=a*b
    div=a/b
    return(add,sub,mul,div)
```

In [46]:
```
add,sub,mul,div=math(10,20)
print(add,sub,mul,div)
```

```
30 -10 200 0.5
```

Local Variables

In [ ]:
```
def avg_with_out():
    n1=eval(input("enter number1:"))
    n2=eval(input("enter number2:"))
    n3=eval(input("enter number3:"))
    ADD_11=n1+n2+n3
    AVG_11=round(ADD_11/3,2)
    print("The addition of {} {} and {} is: {}".format(n1,n2,n3,ADD_11))
    print("The average of {} {} and {} is: {}".format(n1,n2,n3,AVG_11))
    return(AVG11,ADD11)

# How many variables are there:5
# n1 n2 n3 are values provided by user
# ADD_11  and AVG_11 is getting by python cde

# q) n1 n2 n3 are inside the function:  local variables
# these local variables you cant use outside the function, until unless retu
```

In [ ]:
```

```

In [51]:
```
def tax_per(tax_percentage11=10):
    salary=eval(input("enter your salary:"))
    tax_amount=tax_percentage*salary/100
    return(tax_amount)

# what are the local variables: tax_per,salary,tax
```

In [52]:
```
tax_percentage11
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
Cell In[52], line 1
----> 1 tax_percentage11

NameError: name 'tax_percentage11' is not defined
```

In [55]:
```
n1=eval(input("enter number1:")) # GV
n2=eval(input("enter number2:")) # GV
n3=eval(input("enter number3:")) # GV

def avg_with_out():
    ADD_11=n1+n2+n3   #LV
    AVG_11=round(ADD_11/3,2) #LV
    print("The addition of {} {} and {} is: {}".format(n1,n2,n3,ADD_11))
    print("The average of {} {} and {} is: {}".format(n1,n2,n3,AVG_11))
    return(AVG_11,ADD_11)
```

```
enter number1:100
enter number2:200
enter number3:300
```

In [56]:
```
avg,add=avg_with_out()
```

```
The addition of 100 200 and 300 is: 600
The average of 100 200 and 300 is: 200.0
```

- Local varaibles means inside the function
  - you cant use out side the function untill unless you return the variables
- Global variable are out side the function, it can use inside the function also

- define the functions
- with out arguments
- with arguments
- default arguments
- return
- local variable
- global variable

In [ ]:
```
# take three numbers find the greatest number
# n1=50  n2=60  n3=70
if num1>num2 and num1>num3 # num1 is
elif num2>num3  # num2
else: num3
```

In [ ]:
```
def math(x,y):
    add=x+y
    sub=x-y
    mul=x*y
    div=x/y
    return(add,sub,mul,div) # 4 items take ===== 4 bags

ans1,ans2,ans3,ans4=math(20,30)
```

In [61]:
```
n1 = eval(input("Enter the first number: "))
n2 = eval(input("Enter the Second number: "))
n3 = eval(input("Enter the third number: "))

def avg_cal():
    sum_3 = n1+n2+n3
    avg = sum_3/3
    return(sum_3,avg)


add,avg=avg_cal()  # dont give any thing inside
```

```
Enter the first number: 20
Enter the Second number: 30
Enter the third number: 40
```

In [65]:
```
n1 = eval(input("Enter the first number: "))
n2 = eval(input("Enter the Second number: "))
n3 = eval(input("Enter the third number: "))

if n1>n2 and n1>n3:
    print("{} is greatest".format(n1))
elif n2>n3:
    print("{} is greatest".format(n2))
else:
    print("{} is greatest".format(n3))



# define the functions

# with out arguments

# with arguments

# default arguments

# return

# local variable

# global variable
```

```
Enter the first number: 300
Enter the Second number: 100
Enter the third number: 200
300 is greatest
```

- Basic function creation
- With out arguments
- With arguments
- Default arguments
- Return concepts
- Local variable
- Global variable

In [1]:
```
a=20

def val():
    a=30
    print('the value of a:',a)
```

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

```
the value of a: 30
```

In [3]:
```
print(a)

# my main aim is I want to use a=30 value with out return
# I want to use lastest a value i.e 30 outside the functution
# loacl ======= pan
# you need to convert local variable to global variable
```

```
20
```

In [ ]:
```
keyword:  global
```

In [ ]:
```
# local variable  a=30
# global variable is a=20
```

In [8]:
```
a1=20

def val():
    global a1
    a1=30
    print('the value of a1:',a1)
```

In [9]:
```
val()
```

```
the value of a1: 30
```

In [10]:
```
print(a1)
```

```
30
```

In [13]:
```
a=20  #  global variable

def val():
    a=30    #  local variable
    print('the value of a:',a)

val()
print(a)

#=============================================================================
# use case: we want to use latest value of a
#           we want to use local variable outside function call with out re

#=============================================================================
a=20  #  global variable

def val():
    global a
    a=30    #  local variable
    print('the value of a:',a)

val()
print(a)
```

```
the value of a: 30
20
the value of a: 30
30
```

In [20]:
```
num1=30  # global varaible
def add(num2,num3):
    global num1
    num1=num2+num3  #  30,40   30+40  =70
    # local variable global varible


add(30,40)
print(num1)
```

```
70
```

In [ ]:
```
wht is the difference betwn return and global
the parameters
```

In [21]:
```
num1=30  # global varaible
def add(num2,num3):
    global num1
    num1=num2+num3
    return(num2)



num2=add(30,40)
print(num1)     # 70
print(num2)     # 30

# i want to use num2 also out side the function call: return
```

```
70
30
```

In [22]:
```
# Can I use global variable on num2?
num1=30  # global varaible
def add(num2,num3): # with argument num2 is a parameter
    global num1,num2 # num2 also gloabl
    num1=num2+num3



num2=add(30,40)
print(num1)     # 70
print(num2)     # 30

# parameters never become a global variable
# that are always inside the function only
```

```
  Cell In[22], line 4
    global num1,num2
    ^
SyntaxError: name 'num2' is parameter and global
```

In [25]:
```
# Can I use global variable on num2?
# here num2 is a local variable
# it is not a parameter
num1=30  # global varaible
def add(num3):
    global num1,num2
    num2=eval(input('enter number:'))
    num1=num2+num3
    # num1=100+40 =140
    # num2=100



add(40)
print(num1)
print(num2)

# two solutions
# num2 using return statement
# num2 using global statement : global num2  is working or not
```

```
enter number:100
140
100
```

- if you want convert local varaible to global varaible : global keyword
- parameters can not convert into global variable

In [ ]:
```
# tip and bill
# step-1: take bill as global variable : bill=1000
# step-2: create a function , inside the function provide default parameter
# step-3: create global keyword as total amount
# step-4: claculate total amount
# step-5: print total amount out side the function

bill=1000 # step-1  correct
def totalamt(tip=30): # step-2: correct
    # provide global keyword
    totalamt=bill+tip
print("totalamount",totalamt)

bill=1000
def totalamt(tip=30):
    global totalamt
    totalamt=bill+tip
print("totalamount",totalamt)
```

In [26]:
```
Bill=eval(input("Please enter Bill AMount:"))
def Total_Bill(Tip=30):
    try:
        global total_bill
        total_bill = Bill + Tip
    except Exception as e:
        print(e)
Total_Bill()
print ("Total Bill is:",total_bill)
```

```
Please enter Bill AMount:1000
Total Bill is: 1030
```

In [27]:
```
bill=1000
def t_bill(tip):
    global t_bill
    t_bill=bill+((bill*tip)/100)

t_bill(10)
print(t_bill)
```

```
1100.0
```

In [28]:
```
bill_amount=eval(input("enter the bill: "))
def bill1(tip=50):
    global total_amount
    total_amount=bill_amount+tip

bill1()
print(total_amount)
```

```
enter the bill: 1000
1050
```

functions in functions

In [29]:
```
def hey():
    print("vamsi")
    print("hello good morning") # second line I want to replace other funct
    print("how do you do")

hey()
```

```
vamsi
hello good morning
how do you do
```

In [30]:
```
def greet():
    print("hello good morning")

greet()
```

```
hello good morning
```

In [31]:
```
def greet():
    print("hello good morning")

def hey():
    print("vamsi")
    greet()
    print("how do you do")


hey()
```

```
vamsi
hello good morning
how do you do
```

In [32]:
```
def greet ():
    print ('Apoorva')
    print ('hello good morning')
    print ('hello good morning')

greet()
```

```
Apoorva
hello good morning
hello good morning
```

In [33]:
```
def greet1():
    print("hello")

def greet2():
    print('good morning')

def greet3():
    print("how much 3+5?")

def greet4():
    print("i think i dont know")

def solve():
    greet3() #
    greet2()
    greet1()
    greet4()
    print("end the story")

solve()
```

```
how much 3+5?
good morning
hello
i think i dont know
end the story
```

In [42]:
```
def greet1(name1):
    print("hello:",name1)

def greet2(name2):
    print('good morning',name2)

def greet3(val):
    print("how much 3+5?",val)

def greet4():
    print("i think i dont know")

def solve(val,name1,name2):
    greet3(val) #
    greet2(name2)
    greet1(name1)
    greet4()
    print("end the story")

solve(8,'python','anil')
```

```
how much 3+5? 8
good morning anil
hello: python
i think i dont know
end the story
```

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

add(3,7)
```

In [ ]:
```
# simple calaculator program
# define 4 functions individually
# first func: add(a,b) ====== two arguments
#              print("the addition of {} and {} is:",a+b)
# second    : sub(a,b) ====== two arguments
#             print("the subtraction of {} and {} is:",a-b)
# thirs     : mul(a,b) ===== two arguments
#           print("the multiplication of {} and {} is:",a*b)
# fourth    : div(a,b) ====== two arguments
#             print("the division of {} and {} is:",a/b)

# define main function as calaculator(a,b):
# print("if you enter 1 it gives addition")
# print("if you enter 2 it gives sub")
# print("if you enter 3 it gives mul")
# print("if you enter 4 it gives div")
# inside number: input("enter a number between 1 to 4")
# if that number =='1' : the call add(a,b)
# if that number=='2'  : then call sub(a,b)
# if that number=='3'  : then call mul(a,b)
# if that number=='4'  : then call div(a,b)


#calculator(10,20)
```

In [44]:
```
def add(a,b):
    print("Total:",a+b)
def sub(a,b):
    print("SUbstract:", a-b)
def mul(a,b):
    print("Multiplication:", a*b)
def div(a,b):
    print("Division:",a/b)

def calculator(a,b):
    num = input("Please enter a number between 1 to 4:")
    if num == '1':
        add(a,b)
    elif num == '2':
        sub(a,b)
    elif num == '3':
        mul(a,b)
    elif num == '4':
        div(a,b)
    else:
        print("Enter a valid number")

calculator(4,3)
```

```
Please enter a number between 1 to 4:3
Multiplication: 12
```

In [45]:
```
def add(a,b):
    print(a+b)
def sub(a,b):
    print(a-b)
def mul(a,b):
    print(a*b)
def div(a,b):
    print(a/b)

def cal(a,b):
    num=eval(input("Enter a number between 1 to 4:"))
    if num==1:
        add(a,b)
    elif num==2:
        sub(a,b)
    elif num==3:
        mul(a,b)
    elif num==4:
        div(a,b)

cal(10,20)
```

```
Enter a number between 1 to 4:3
200
```

In [51]:
```
def add(a,b):
    print("Addition is",a+b)
def minus(a,b):
    print("Subtractions is", a-b)
def mul(a,b):
    print("Multiplication is ", a*b)
def div(a,b):
    print("Divison is",a/b)
def calc(a,b):
    num = input("enter the numberbetween 1 and 4 : ") #  string
    if num =='1':
        add(a,b)
    elif(num =='2'):
        minus(a,b)
    elif(num =='3'):
        mul(a,b)
    else:
        div(a,b)
calc(5,9)
```

```
enter the numberbetween 1 and 4 : 3
Multiplication is  45
```

In [53]:
```
try:
    def add(a,b):
        print("add:",a+b)
    def sub(a,b):
         print("sub:",a-b)
    def mul(a,b):
         print("mul:",a*b)
    def div(a,b):
         print("div:",a/b)
    def calculator(a,b):
        number=eval(input("enter number between 1 to 4"))
        if number==1:
            add(a,b)
        if number==2:
            sub(a,b)
        if number==3:
            mul(a,b)
        if number==4:
            div(a,b)
    calculator(2,3)
except exception as e:
    print(e)
```

```
enter number between 1 to 42
sub: -1
```

In [ ]:
```
def add(a,b):
    return a+b
def multiplication(a,b):
    return a*b
def division(a,b):
    return a/b
def substract(a,b):
    return a-b
try:
    option = eval(input("enter a number"))
    a,b = 10, 20
    if(option ==1):
         print ("addition:",add(a,b))
    elif(option ==2):
         print ("substraction:",substract(a,b))
    elif(option ==3):
         print ("multiplication:",multiplication(a,b))
    elif(option ==4):
         print ("division:",division(a,b))
    else:
        print("wrong choice")

except Exception as ex:
    print("Exception", ex)
```

In [55]:
```
def add(a,b):
    print("the addition of {} and {} is: {}".format(a,b,a+b))

def sub(a,b):
    print("the substraction of {} and {} is: {}".format(a,b,a-b))

def mul(a,b):
    print("the multiplication of {} and {} is: {}".format(a,b,a*b))

def div(a,b):
    print("the division of {} and {} is: {}".format(a,b,a/b))

def calculator(a,b):
    print('+  then add')
    num=input("Enter the operation +,-,*,/: ")
    if num=='+':
        add(a,b)
    elif num=='-':
        sub(a,b)
    elif num=='*':
        mul(a,b)
    elif num=='/':
        div(a,b)

calculator(10,20)
```

```
Enter the operation +,-,*,/: +
the addition of 10 and 20 is: 30
```

In [56]:
```
def add(a,b):
    print ("addition is",a+b)
def sub(a,b):
    print("substraction is:",a-b)
def mul(a,b):
    print("multiplication:",a*b)
def div(a,b):
    print("division is:",a/b)
def calculator(a,b):
    number=eval(input("enter the number 1 to 4"))
    if number==1:
        add(a,b)
    elif number==2:
        sub(a,b)
    elif number==3:
        mul(a,b)
    else:
        div(a,b)

calculator(30,40)
```

```
  Cell In[56], line 1
    def add(30,40):
            ^
SyntaxError: invalid syntax
```

In [ ]:
```
def add(a,b):
    print('the addition of {} and {} is :'.format(a,b),a+b)
def sub(a,b):
    print('the substraction of {} and {} is :'.format(a,b),a-b)
def mul(a,b):
    print('the multiplication of {} and {} is :'.format(a,b),a*b)
def div(a,b):
    print('the division of {} and {} is :'.format(a,b),a/b)

def cal(a,b):
    num=eval(input("enter a num:"))
    if num==1:
        add(a,b)
    elif num==2:
        sub(a,b)
    elif num==3:
        mul(a,b)
    else:
        div(a,b)
```

In [57]:
```
def add(a,b):
    print("add",a+b)
def sub(a,b):
     print("sub",a-b)
def mul(a,b):
     print("mul",a*b)
def sep(a,b):
     print("sep",a/b)

def calculation(a,b):


    n1=eval(input("enter a number between 1 to 4:"))
    if n1==1:
        add(a,b)
    elif n1==2:
        sub(a,b)
    elif n1==3:
        mul(a,b)
    elif n1==4:
        sep(a,b)
    else:
        print("the correct one")

calculation(5,9)
```

```
enter a number between 1 to 4:2
sub <function sub at 0x000002E2DCF4DB20>
add 14
sub -4
mul 45
sep 0.5555555555555556
```

In [ ]: