In [ ]:
```
- function without arguments
- function with arguments
- Defalut
- local vs global
- return
- function in function
```

In [ ]:
```
#Practice every day
#Attend the classes
```

In [3]:
```
#Functions with out arguments
def even_odd():
    num = eval(input("Enter a number:"))
    if num%2==0:
        print('Even')
    else:
        print('Odd')

even_odd()
```

```
Enter a number:78
Even
```

In [4]:
```
#Functions with  arguments
def even_odd(num):

    if num%2==0:
        print('Even')
    else:
        print('Odd')

even_odd(78)
```

```
Even
```

In [6]:
```
#Default Arguments:this means we can fix the argument values or the argument is
#Defalut argument should always be at last i.e second parameter.
def summ(num1,num2=100):
    print("num1:", num1)
    print("num2:", num2)
    add=num1+num2
    print(add)
summ(20)
#num1 = 20 num2=100
```

```
num1: 20
num2: 100
120
```

In [8]:
```
def summ(num1=200,num2=100):
    print("num1:", num1)
    print("num2:", num2)
    add=num1+num2
    print(add)
summ()
```

```
num1: 200
num2: 100
300
```

- Note : Deafult arguments is always at last

In [ ]:
```
avg(n1,n2,n3=100) #Works
avg(n1,n2=100,n3=100)#Works
avg(n1=100,n2=100,n3=100) #Works
avg(n1=100,n2,n3)#Fails
avg(n1=100,n2=100,n3) #Fails
avg(n1=100,n2,n3=100) #Fails
avg(n1,n2=100,n3=100)#Works
```

In [10]:
```
#Case-1:
def avg(n1,n2,n3=100):
    averge=(n1+n2+n3)/3
    print(averge)
avg(200,300)  #n1=200 n2=300 n3=100
```

```
200.0
```

In [11]:
```
#Case-2:
def avg(n1,n2=100,n3=100):
    averge=(n1+n2+n3)/3
    print(averge)
avg(300)
```

```
166.66666666666666
```

In [13]:
```
#Case-3:
def avg(n1=100,n2=100,n3=100):
    averge=(n1+n2+n3)/3
    print(averge)
avg()
```

```
100.0
```

In [14]:
```
#Case-4:
def avg(n1=100,n2,n3):
    averge=(n1+n2+n3)/3
    print(averge)
avg(100,200)
```

```
  Cell In[14], line 1
    def avg(n1=100,n2,n3):
                    ^
SyntaxError: non-default argument follows default argument
```

In [17]:
```
#Case-5:
def avg(n1=100,n2=100,n3):
    averge=(n1+n2+n3)/3
    print(averge)
avg(200)
```

```
  Cell In[17], line 1
    def avg(n1=100,n2=100,n3):
                          ^
SyntaxError: non-default argument follows default argument
```

In [18]:
```
#Case-6:
def avg(n1=100,n2,n3=100):
    averge=(n1+n2+n3)/3
    print(averge)
avg(100)
```

```
  Cell In[18], line 1
    def avg(n1=100,n2,n3=100):
                    ^
SyntaxError: non-default argument follows default argument
```

In [19]:
```
#Case-7:
def summ(n1,n2=100):
    averge=n1+n2
    print(averge)
summ(100)
```

```
100.0
```

In [ ]:
```
def avg(n1,n2=100 ):
    averge=(n1+n2+n3)/3
    print(averge)
avg(100)
# have you provided any value before define the function
# what are you provided while defineing the function
# what are you provided while calling the function num2= 100
# what are you provided while calling the function num2=500
```

In [ ]:
```
#define call then kaam
```

In [20]:
```
#Create the default argument for tip_percentage
def totalbill(bill, tip=10):
#     bill = eval(input("Enter the bill:"))
#     tip =eval(input("Enter the tip percentage :"))
    tip_percent =(bill*tip)/100
    total_bill=bill+tip_percent
    print(f"The total bill is {total_bill} for the tip percent {tip}")

totalbill(1000)
```

```
The total bill is 1100.0 for the tip percent 10
```

In [ ]:
```
#wap ask the user enter salary :100000
#ask the user enter tax percentage : 10
#calculate tax
#implement fnction without argument
#function with argument
#function default argument :tax_per = 10
```

In [23]:
```
#implement fnction without argument
def salary():
    sal=eval(input('Enter the salary:'))
    tax=eval(input('Enter the tax percentage:'))
    tax_per=(sal*tax)/100
    print(f'The salary is {sal} and tax percent is {tax_per}')
salary()
```

```
Enter the salary:100000
Enter the tax percentage:10
The salary is 100000 and tax percent is 10000.0
```

In [25]:
```
#function with argument
def salary1(sal,tax):

    tax_per=(sal*tax)/100
    print(f'The salary is {sal} and tax percent is {tax_per}')
salary1(100000,10)
```

```
The salary is 100000 and tax percent is 10000.0
```

In [26]:
```
#function default argument :tax_per = 10

def salary1(sal,tax=10):

    tax_per=(sal*tax)/100
    print(f'The salary is {sal} and tax percent is {tax_per}')
salary1(100000)
```

```
The salary is 100000 and tax percent is 10000.0
```

- Return Statements

In [ ]:
```
# tax_per we re getting inside the function whereas,
# if i want to use it outside I am supposed to use
# retun function
```

In [32]:
```
def salary1(sal,tax=10):
    tax_per=(sal*tax)/100
    return(tax_per)
#we are asking function to return a value
tax_per=salary1(100000)
print(tax_per)
print(sal)
```

```
10000.0

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[32], line 7
      5 tax_per=salary1(100000)
      6 print(tax_per)
----> 7 print(sal)

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

In [ ]:
```

```

In [35]:
```
#WAP take 3 numbers do the sum seperately and average seperately and return sum
def mathe(n1,n2,n3):
    summ=n1+n2+n3
    avr=(n1+n2+n3)/3
    return(avr,summ)
avr,summ=mathe(10,100,10000)
print(summ)
print(avr)
```

```
10110
3370.0
```

In [ ]:
```

```