In [ ]:

```
# simple function
def my_func():
  print("Hello! Hope you're doing well")
my_func()
```

```
Hello! Hope you're doing well
```

In [ ]:

```
#function call can be before function defintion
my_func()
def my_func():
  print("Hello! Hope you're doing well")
```

```
Hello! Hope you're doing well
```

In [ ]:

```
#function with argumets
def my_func(name,place):
  print(f"Hello {name}! Are you from {place}?")
my_func("john","chennai")
```

```
Hello john! Are you from chennai?
```

In [ ]:

```
#types of functions
#defining function no return value and no parameter
def sum():
  a=int(input("number1"))
  b=int(input("number2"))
  print(a + b)
sum()
```

```
number12
number23
5
```

In [ ]:

```
#defining function with parameter and  no return value
def sum(x,y):
    print(x + y)
a=int(input("number1"))
b=int(input("number2"))
sum(a,b)
```

```
number13
number22
5
```

In [ ]:

```
#defining function with  no parameter but with return value
def sum():
    a=int(input("number1="))
    b=int(input("number2="))
    return (a + b)

c=sum()
print("sum is {}".format(c))
```

```
number1=3
number2=2
sum is 5
```

In [ ]:

```

```

In [ ]:

```
#defining function with  parameter and return value
def sum(a, b):
  return a + b
result = sum(1, 2)
print(result)
```

```
3
```

In [ ]:

```
#Pass by Reference vs Value
#mutable data objects are passed by reference its value will change
def funct1(a):
  a[0]=100
a=[1,2,3]
funct1(a)
print(a)

#strings are immutable.so its value wont change
def f(s):
    s='cek'
    print (s)
s='mec'
f(s)
print(s)
```

```
[100, 2, 3]
cek
mec
```

In [ ]:

```
#function to find maximum of 2 numbers
def maximum(a,b):
  if(a>b):
    print("largest is ",a)
  else:
    print("largest is ",b)

x=int(input("number1="))
y=int(input("number2="))
maximum(x,y)
```

```
number1=4
number2=3
largest is  4
```

In [ ]:

```
#function to find  the absolute value of a number
def absval(a):
  if(a>0):
    print(a)
  else:
    print(a*-1)

x=int(input("number1="))
absval(x)
```

```
number1=-3
3
```

In [ ]:

```
#different types of formal arguments -Variable-length arguments"""
#1)Required arguments are the arguments passed to a function in correct positional orde
r.
#Here, the number of arguments in the function call should match exactly with the funct
ion definition.
def hello(s,msg):
  print( "the person is {} and message is{}" .format(s,msg))
hello("mark","how are you")
#arguments should be in correct positional order.

def hello2(msg,s):
  print("the person is {} and message is{}" .format(s,msg))
hello2("mark","how are you")
```

```
the person is mark and message ishow are you
the person is how are you and message ismark
```

In [ ]:

```
#positional arguments shows error if one value is missing
def hello3(s,msg):
  print( s,msg)
hello3("mark")
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-22-7b6427e53925> in <module>()
      2 def hello3(s,msg):
      3   print( s,msg)
----> 4 hello3("mark")

TypeError: hello3() missing 1 required positional argument: 'msg'
```

In [ ]:

```
#2)Default arguments
def hello(name="mark",msg="how are you"):
  print("the person is {} and message is {}" .format(name,msg))
hello("john","welcome")
hello('peter')
hello()
```

```
the person is john and message is welcome
the person is peter and message is how are you
the person is mark and message is how are you
```

In [ ]:

```
#3)Keyword arguments
def hello(name,msg):
  print(name,msg)
hello(name="john",msg="hai")
hello(msg="hello",name="edwin")
```

```
john hai
edwin hello
```

In [ ]:

```
#4)arbitary arguments
def hello(*names):
     for name in names:
         print("Hello", name)

hello("adam","mark","john")
```

```
Hello adam
Hello mark
Hello john
```

In [ ]:

```
#add sum of n numbers passed
def add(*x):
  sum=0
  for i in x:
    sum=sum+i
  print(sum)
add(1,2,3)
add(1,4,3,2,4,6,7)
```

```
6
27
```

In [ ]:

```
#map function
integer=["23","12",'11']
newlist=map(int,integer)
print(newlist)
print(list(newlist))
```

```
<map object at 0x7f6f8a198e10>
[23, 12, 11]
```

In [ ]:

```
#reduce function
from functools import reduce
def add(x,y):
  return(x+y)
def mul(x,y):
  return(x*y)
data=[8,4,2]
addval=reduce(add,data)
mulval=reduce(mul,data)
print(addval,"summation of the list")
print (mulval,"product of the list")
```

```
14 summation of the list
64 product of the list
```

In [ ]:

```
#lambda function
z=lambda x,y:x+y
print ((z(2,3)))
```

```
5
```

In [ ]:

```
#filter function:a function(predicate) is applied to each value in the list.
#if the value returns true its added  to the filter object.otherwised dropped
def odd(n):
  return n%2==1
f1=filter(odd,range(10))
print(list(f1))
```

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