```
# function as parameter

def f1(f):
  return 'hello ' +f
def f2():
  return 'ravi'



f1(f2())
```

```
'helloravi'
```

```
# inner function

def outer ():
  def inner():
    print('hello')
  return inner()
outer()
```

```
hello
```

```
inner()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-bc10f1654870> in <module>
----> 1 inner()

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

SEARCH STACK OVERFLOW

```
def outer(x):
  def inner():
    print('i am inner func')
    y=x+100
    return y
  return inner()

outer(10)
```

```
i am inner func
110
```

```
# decorator

def div (x,y):
  return x/y
```

```
div(10,2)
```

```
5.0
```

```
div(20,5)
```

```
4.0
```

```
div(5,10)
```

```
0.5
```

```
def div(x,y):
  if x<y:
    x,y=y,x
    return x/y

div(2,10)
```

```
5.0
```

```
def div_modify(func):
  def inner(x,y):
    if x<y:
      x,y=y,x
    return  func(x,y)
  return inner

def div (x,y):
  return x/y


d=div_modify(div)
d(10,2)
```

```
5.0
```

```
d(2,10)
```

```
5.0
```

```
def div_modify(func):
  def inner(x,y):
    if x<y:
      x,y=y,x
    return  func(x,y)
  return inner
@div_modify
def div (x,y):
  return x/y
```

```
div(4,10)
```

```
2.5
```

```
def outer_decor(func):
  def inner():
    val=func()
    s=val+100
    d=val-50
    return s,d
  return inner
@outer_decor
def show():
  x=10
  return x
@outer_decor
def func2():
  return 20

def func3():
  return 40
```

```
show()
```

```
(110, -40)
```

```
func2()
```

```
(120, -30)
```

```
def show():
  x=20
  s=x+100
  d=x-50
  return s,d
show()
```

```
(120, -30)
```

```
func3()
```

```
40
```

```
# iterable

l=[1,2,3,4]
for i in l:
  print(i)
```

```
1
```

```
2
3
4
```

```
s='sai ravi'
for i in s:
  print(i)
```

```
s
a
i

r
a
v
i
```

```
x=45
for i in x:
  print(i)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-eff6014ecc17> in <module>
      1 x=45
----> 2 for i in x:
      3   print(i)

TypeError: 'int' object is not iterable
```

SEARCH STACK OVERFLOW

```
l=[10,20,30,40]
for i in l:
  print(i)
```

```
10
20
30
40
```

```
next(l)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-29-cdc8a39da60d> in <module>
----> 1 next(l)

TypeError: 'list' object is not an iterator
```

SEARCH STACK OVERFLOW

```
# to convert iterable into iterator ...use iter()
b=iter(l)
b
```

```
<list_iterator at 0x7f3a65342090>
```

```
next(b)
```

```
10
```

```
next(b)
```

```
20
```

```
next(b)
```

```
30
```

```
next(b)
```

```
40
```

```
next(b)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-35-adb3e17b0219> in <module>
----> 1 next(b)

StopIteration:
```

SEARCH STACK OVERFLOW

```
t=(1,2,3,4) # iterable

x=iter(t)    # iterator
next(x)
```

```
1
```

```
next(x)
```

```
2
```

```
next(x)
```

```
3
```

```
next(x)
```

```
4
```

```
next(x)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-41-92de4e9f6b1e> in <module>
----> 1 next(x)

StopIteration:
```

SEARCH STACK OVERFLOW

```
z=10
y=iter(z)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-42-449c9cdb63cc> in <module>
      1 z=10
----> 2 y=iter(z)

TypeError: 'int' object is not iterable
```

SEARCH STACK OVERFLOW

```
# yield #   generator
range(4)
```

```
range(0, 4)
```

```
list(range(4))
```

```
[0, 1, 2, 3]
```

```
for i in range(4):
  print(i)
```

```
0
1
2
3
```

```
def func2(n):
  v=[v for v in range(n)]
  return v
func2(4)
```

```
[0, 1, 2, 3]
```

```
def func2(n):
  v=[v for v in range(n)]
  yield v
d=func2(4)
d
```

```
<generator object func2 at 0x7f3a652af1d0>
```

```
list(d)
```

```
[[0, 1, 2, 3]]
```

```
for i in d:
  print(i)
```

```
[0, 1, 2, 3]
```

```
def fib(n):
  a=1
  b=1
  l=[]
  for i in range(n):
    l.append(a)
    a,b=b,a+b
  return l
fib(1000000)
```

```
def fib_gen(n):
  a=1
  b=1
  for i in range(n):
    yield a
    a,b=b,a+b

fb=fib_gen(1000000)
fb
```

```
<generator object fib_gen at 0x7f7b45c6e050>
```

```
list(fb)
```

```
def f(*args,a,b,c):
  return args,a,b,c
f(1,2,3,a=4,b=10,c=0)
```

```
((1, 2, 3), 4, 10, 0)
```

```
def f(a,b,c,*args):
  return args,a,b,c
f(1,2,3,4,5,6)
```

```
((4, 5, 6), 1, 2, 3)
```

```
def f(**kwargs):
  return kwargs
f(1,2,3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-57badc479b1e> in <module>
      1 def f(**kwargs):
      2   return kwargs
----> 3 f(1,2,3)

TypeError: f() takes 0 positional arguments but 3 were given
```

SEARCH STACK OVERFLOW

```
f(x=1,y=2,z=3)
```

```
{'x': 1, 'y': 2, 'z': 3}
```

```
def display(f):
  return 'hello'+f
def show():
  return ' prabhat'

display(show())
```

```
'hello prabhat'
```

```
def decor_(fun):
  def inner():
    val=fun()
    val+=100
    return val
  return inner
@decor_
def fn1():
  return 10
fn1()
```

```
110
```

```
##

def f1(func):
```

```
  def wrapper():
    print('started')
    func()
    print('ended')
  return wrapper
@f1
def f2():
  print('hello')
f2()
```

```
started
hello
ended
```

```
# #
def sales(no_of_items,price_per_item):
  p=no_of_items*price_per_item
  if p>=500:
    p=p-p*0.1
    return p
  else:
    return p
sales(30,20)
```

```
540.0
```

```
## discount decorator
def discount(func):
  def inner(x,y):
    p=func(x,y)
    if p>=500:
      p=p-p*0.1
      return p
    else:
      return p
  return inner
@discount
def sales(no_of_items,price_per_item):
  return no_of_items*price_per_item

sales(5,20)
```

```
100
```

```
def discount(func):
  def inner(*args):
    p=func(*args)
    if p>=500:
      p=p-p*0.1
      return p
    else:
      return p
  return inner
@discount
```

```
def sales(no_of_items,price_per_item,length):
  return no_of_items*price_per_item*length

sales(10,20,3)
```

```
def discount(func):
  def inner():
    p=func()
    if p>=500:
      p=p-p*0.1
      return p
    else:
      return p
  return inner
@discount
def sales():
  return 500

sales()
```

```
450.0
```

```
# decorator can be used in class also

def before_after(func):
  def wrapper(*args):   # wrapper(x)
    print('before')
    func(*args)         # func(x)
    print('after')
  return wrapper

class Test:
  @before_after
  def decor_method(self):
    print('Run')
t=Test()
t.decor_method()
```

```
before
Run
after
```

```
# time decorator
import time
def timer(f):
  def wrapper():
    before=time.time()
    #print(before)
    f()
    #print(time.time())
    print('function took :',time.time()-before," seconds")
  return wrapper
@timer
```

```
def run():
     time.sleep(3)
run()
```

```
     function took : 3.0031447410583496  seconds
```

```
time.sleep(3)
```

```
# decor for calculating execution time for a fun
import time
def timer(f):
  def wrapper(*args):  # x
    before=time.time()
    #print(before)
    f(*args)  # *args--> x
    #print(time.time())
    print('function took :',time.time()-before," seconds")
    #return f(x)
  return wrapper
@timer
def fib(n):
  a=1
  b=1
  l=[]
  for i in range(n):
    l.append(a)
    a,b=b,a+b
  return l
fib(99000)
```

```
     function took : 0.692270040512085  seconds
```

```
# Two decorator in a function
def dadecor(func):
  def inner():
    sal=func()
    sal+=sal*0.10
    return sal
  return inner()
def hradecor(func):
  def inner():
    sal=func()
    sal+=sal*0.20
    return sal
  return inner

@dadecor
@hradecor
def emp_sal():
  return 10000
emp_sal
```

```
13200.0
```

```
##
def main_func(fun):
  def inner_func(*args):
    print('Function that is running : ',fun.__name__)
    print(fun.__name__,fun(*args))
    op=fun(*args)+10
    print('final op= ',op)
  return inner_func
@main_func
def add(*args):
  s=0
  for i in args:
    s+=i
  return s
@main_func
def mul(*args):
  m=1
  for i in args:
    m*=i
  return m


add(1,2,3,4)
```

```
Function that is running :  add
add 10
final op=  20
```

```
mul(1,2,3,4)
```

```
Function that is running :  mul
mul 24
final op=  34
```

```
###
import time
def main_func(fun):
  def inner_func(*args):
    print('Function that is running : ',fun.__name__)
    before=time.time()
    print(fun.__name__,fun(*args))
    op=fun(*args)+10
    print('function took :',time.time()-before, " seconds")
    print('final op= ',op)
  return inner_func
@main_func
def add(*args):
  s=0
  for i in args:
    s+=i
  return s
@main_func
```

```
def mul(*args):
  m=1
  for i in args:
    m*=i
  return m

add(1,2,3,4)
```

```
     Function that is running :  add
     add 10
     function took : 6.079673767089844e-05  seconds
     final op=  20
```

```
mul(1,2,3,4)
```

```
     Function that is running :  mul
     mul 24
     function took : 5.030632019042969e-05  seconds
     final op=  34
```

```
## decorator for writing log file
import datetime
import time

def log(func):
  def wrapper(*args):
    val=func(*args)
    with open('log_file.txt','a') as f:
      f.write('The running function is : '+func.__name__+' of '+
              ' '.join([str(arg) for arg in args])+
              ' at '+str(datetime.datetime.now())+ ' and value= '+str(val)+
              '\n')

    return val
  return wrapper
@log
def sum(a,b,c):
  return a+b+c
sum(10,20,30)
```

```
60
```

```
time.time()
```

```
     1668499189.9967787
```

Colab paid products - Cancel contracts here

✓ 0s completed at 3:55 PM