Advanced concepts in Function

In [6]:
```
#function as arg

def show(x):
    print("this is show")
    print(x)

def disp():
    print("this is disp")

show(10)
show(10.5)
show(disp)
```

```
this is show
10
this is show
10.5
this is show
<function disp at 0x0000019FBCB66660>
```

In [9]:
```
#function as arg

def show(x):
    print("this is show")
    x()

def disp():
    print("this is disp")

show(disp)
#show(10)
```

```
this is show
this is disp
```

In [10]:
```
p=print
print('hi')
p('hello')
```

```
hi
hello
```

In [11]:
```
p=10
p()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 2
      1 p=10
----> 2 p()

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

In [ ]:
```
x=fun    # x is simply refering to fun
x=fun()  #x is refereing to return value of fun
```

In [12]:
```
#fun as return stmt

def show():
    print("this is show")
```

```
def disp():
    print("this is disp")
    return show

x=disp()
```

```
this is disp
```

In [13]:
```
x()
```

```
this is show
```

In [19]:
```
#fun as return stmt

def show():
    print("this is show")

def disp():
    print("this is disp")
    return show

x=disp()
x()
x()
```

```
this is disp
this is show
this is show
```

In [17]:
```
#Function chaining

def show():
    print("this is show")

def disp():
    print("this is disp")
    return show

disp()() #Function chaining
```

```
this is disp
this is show
```

In [18]:
```
disp()()()
```

```
this is disp
this is show
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 disp()()()

TypeError: 'NoneType' object is not callable
```

In [3]:
```
#Nested Function
def show():                  #outer/enclosing/top level fun
    print("this is show")
    def disp():              #inner/nested/local fun
        print("this is disp")
    print("end of show")

show()
disp()
```

```
this is show
end of show
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 9
      6     print("end of show")
      8 show()
----> 9 disp()

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

In [5]:
```
#Nested Function
def show():                 #outer/enclosing/top level fun
    print("this is show")
    def disp():             #inner/nested/local fun
        print("this is disp")
    print("end of show")
    disp()
    disp()
show()
```

```
this is show
end of show
this is disp
this is disp
```

In [8]:
```
#Nested Function
def show():                 #outer/enclosing/top level fun
    print("this is show")
    def disp():             #inner/nested/local fun
        print("this is disp")
    print("end of show")
    return disp

f=show()
f()
f()
```

```
this is show
end of show
this is disp
this is disp
```

In [9]:
```
#Closure Function
def show():
    print("this is show")
    x=10
    def disp():         #closure function
        print("this is disp",x)
    print("end of show")
    return disp

f=show()
f()
```

```
this is show
end of show
this is disp 10
```

In [10]:
```
def show():
    x=10
    print("this is show",x)
    def disp():
        x=20
        print("this is disp",x)
    disp()
    print("end of show",x)
```

```
show()
```

```
this is show 10
this is disp 20
end of show 10
```

In [12]:
```
#nonlocal keyword
def show():
    x=10
    print("this is show",x)
    def disp():
        nonlocal x
        x=20
        print("this is disp",x)
    disp()
    print("end of show",x)

show()
print(x)
```

```
this is show 10
this is disp 20
end of show 20
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 12
      9     print("end of show",x)
     11 show()
---> 12 print(x)

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

In [13]:
```
def show():
    x=10
    print("this is show",x)
    def disp():
        global x
        x=20
        print("this is disp",x)
    disp()
    print("end of show",x)

show()
print(x)
```

```
this is show 10
this is disp 20
end of show 10
20
```

In [14]:
```
#Generator fun
def mygen():          #definition of generator
    print('hi')
    yield 5
    print('hello')
    yield 7.5
    yield 8

gen=mygen()          #returns generator type

print(next(gen))
```

```
hi
5
```

```
In [15]: print(next(gen))
```

```
hello
7.5
```

```
In [16]: print(next(gen))
```

```
8
```

```
In [17]: print(next(gen))
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
Cell In[17], line 1
----> 1 print(next(gen))

StopIteration: 
```

```
In [18]: def mygen():
             start=1
             while(start<=3):
                 yield start
                 start+=1

         gen=mygen()
         print(next(gen))
         print(next(gen))
         print(next(gen))
```

```
1
2
3
```

```
In [19]: def mygen():
             start=1
             while(start<=3):
                 yield start
                 start+=1

         gen=mygen()
         for i in gen:
             print(i)
```

```
1
2
3
```

```
In [20]: for i in range(1,4):
             print(i)
```

```
1
2
3
```

```
In [24]: def mygen(start,stop,step):
             while(start<=stop):
                 yield start
                 start+=step

         gen=mygen(10,15,1)
         for i in gen:
             print(i)
```

```
10
11
12
13
```

```
14
15
```

In [28]:
```python
def mygen(start,stop,step=1):
    while(start<stop):
        yield start
        start+=step

gen=mygen(10,15,.5)
for i in gen:
    print(i)
```

```
10
10.5
11.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5
```

In [31]:
```python
def mygen(start,stop,step=1):
    while(start<stop):
        yield start
        start+=step

for i in mygen(1,5,.5):
    print(i)
```

```
1
1.5
2.0
2.5
3.0
3.5
4.0
4.5
```

In [34]:
```python
#Decorator fun
def datedecorator(fun):
    def adddate():
        fun()
        import time
        print(time.ctime())
    return adddate

def login():
    print("this is login")

login()
login1=datedecorator(login)
login1()
```

```
this is login
this is login
Sat Aug 12 09:31:56 2023
```

In [35]:
```python
login1()
```

```
this is login
Sat Aug 12 09:32:03 2023
```

In [36]:
```python
login()
```

```
this is login
```

In [37]:
```
#Decorator fun
def datedecorator(fun):
    def adddate():
        fun()
        import time
        print(time.ctime())
    return adddate

def login():
    print("this is login")

login()
login=datedecorator(login)
login()
```

```
this is login
this is login
Sat Aug 12 09:33:01 2023
```

In [38]:
```
login()
```

```
this is login
Sat Aug 12 09:33:06 2023
```

In [41]:
```
#Decorator fun
def datedecorator(fun):
    def adddate():
        fun()
        import time
        print(time.ctime())
    return adddate

@datedecorator
def login():
    print("this is login")

login()
```

```
this is login
Sat Aug 12 09:35:44 2023
```

In [45]:
```
@datedecorator
def register():
    print("this is register")

register()
```

```
this is register
Sat Aug 12 09:43:52 2023
```

In [ ]: