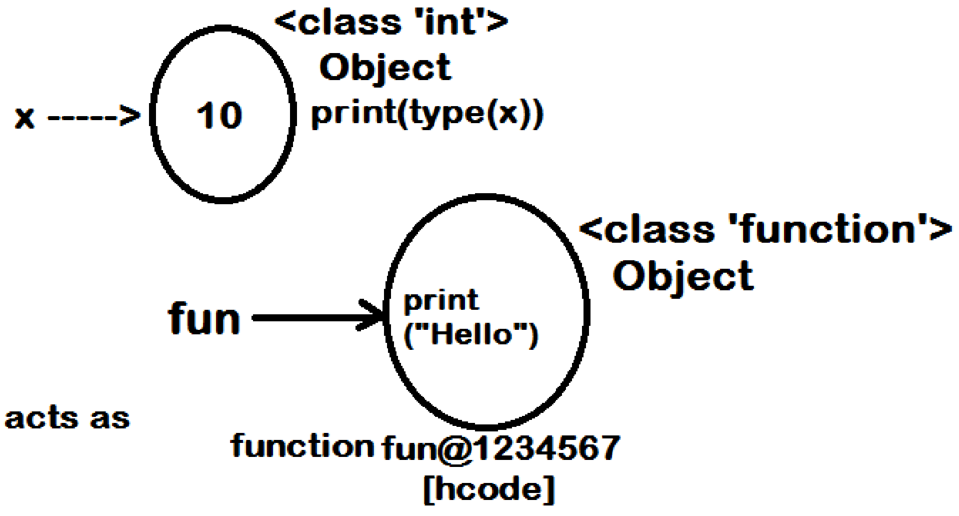
Functions are First Class Object :

- The Python Memory Everything will be stored in the form objects
- For Every Function internally one object will be created of type <class ‘function’>
- However you are working with other objects [ordinary object] the same passion we can also work with the function object

```
x=10
```

```
def fun():
   print("Hello")
```



> Function name itself acts as reference variable

> reference variable is refere to the function object by holding its hashcode

```
print( type(fun) ) #<class 'function'>
print(fun) #hcode of fun object
```

Example:

```
def fun():
   print("Hello")
#calling
print("Type is ",type(fun)) #<class 'function'>
print("Fun is : ",fun)
```

Note:

> We can assign an object[variable] to a variable

```
x=10
y=x
```

1.We can Assign A Function Object to a variable

>We can pass an object [variable] to the function as an argument

2.We can Pass A Function as an Argument to another Function

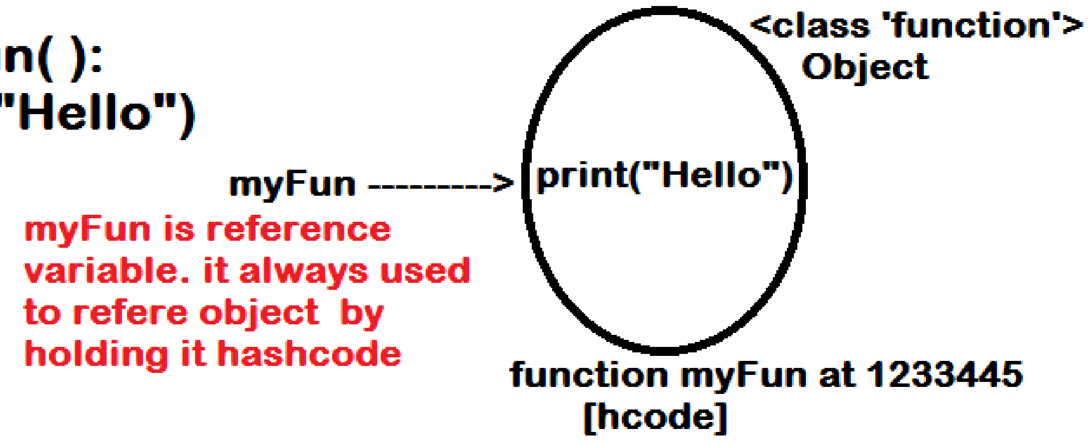
>We Can Use an object as a local variable

3.We can define A function inside another Function

>A Function can return a variable[object] as value

4.A Function can return another function

```
def myFun( ):
    print("Hello")
```


<class 'function'>
Object
myFun --------->
print("Hello")
myFun is reference variable. it always used to refere object by holding it hashcode
function myFun at 1233445
[hcode]

```
print("Type is : ",type(myfun)) #<class 'function'>
print("Value is : ",myfun) #function myFun at 1233445

myFun  # you are getting hcode of function object
myFun( ) #executing myFun( )
```

Example On : We can Assign A Function To A Variable

```
def myFun():  #myFun fun_name acts as ref
   print("Hello")   #variable

print("Type is : ",type(myFun)) #<class 'function'>
print("value is : ",myFun) #<function myFun at 2333445>
myFun( ) #calling myFun( )
print("-----------------------------------------------")
x=myFun  #ref.copy
print("Type is : ",type(x)) #<class 'function'>
print("value is : ",x) #<function myFun at 2333445>
x()  #calling myFun()
```

Example 2: We Can Pass A Function As An Argument To Another Function

```
def greet():
   s="Hello"
   return s

def SGreetings(func): #func is hcode of greet function
   a=func() #calling greet( )-> "Hello"
   b=a+" MyDear...!" # "Hello"+" MyDear .!" -> Hello MyDear.!
   return b


#calling
a=greet()
print("Result of Greet :",a) #Hello

c=SGreetings( greet ) #passing greet hcode to
print("Result of SGreetings : ",c) #Hello MyDear .!
```

Example 3: We Can Define A Function Inside Another Function

```
def myFun():
   def stars():
      for i in range(1,11):
         print('*',end=' ')

   stars()
   print("\n Welcome")
   stars()
   print("\n To")
   stars()

#calling
myFun()
```

Example : A Function Return Another Function

- If you want use the result of inner function inside of OuterFunction then Inner function need return value.

```
def OuterFun():
   def InnerFun():
      a=10
      return a

   r=InnerFun()
   print("Result is ",r)
```

```
#Calling
OuterFun()
```

Case 2:

'''Case 2: If you want use the result of inner function outside of OuterFunction then Outer function need to return the result of innerfun'''

```
def OuterFun():
   def InnerFun():
      a=10
      return a

   r=InnerFun()
   return r

#Calling
b=OuterFun()
print("Result of Inner Function : ",b)
```

Case 3:

```
'''Case 3: If you want use the  inner function
outside of OuterFunction then Outer function need to
return  innerfun function '''

def OuterFun():
   def InnerFun():
      a=10
      return a
   return InnerFun #Hcode of InnerFun
#Calling
hcif=OuterFun() #hashcode of innerFun
print(hcif)
r=hcif()   #calling InnerFun()
print("Result is : ",r)
```

Example On nonlocal keyword:

```
def OuterFun():
   x=30
   def InnerFun():
      nonlocal x
      x=x+20
      print("InnerFun x : ",x)
   InnerFun()

#Calling
OuterFun()
```