Functions

August 26, 2024

Q1)What is the difference between a function and a method in python.

A)Functions: 1)A python functions are self contained chunk code designed to perform a specific task. 2)It is like tiny independent program within a main program 3)Functions are defined using "def" keyword and they can take arguments and return a value 4)They are independent blocks of code, can tie down to the specific object and can use again and again throughout your program ex:imagine a function like a handy tool in toolbox.you can wip it out whenever you needed,use it to perform a task and then put it back,ready to be used again. 5)you call functions independently

[ ]:
```
# simple syntax:
def my_function():
                #my_code
my_function()
```

[10]:
```
#ex:
def greetings():
    print("welcome to the office")
greetings()
```

```
welcome to the office
```

B)Methods: 1)Like functions methods are defined to carry out specific tasks 2)They are bound intrinsically to specific objects 3)They are defined within a class and they describe the behaviours and actions of an object created from that class. 4)When you are defining a method,it automatically takes the object it belongs to as its first argument referred to as "self" This allows the method to access and modify the object's internal state.in short,you call methods on objects,allowing them to interact with object's internal state. ex:personal assistant to object knows all about the object,it can access its private information,can perform task specifically designed for that object Hence,methods are functions declared within a class Three types of methods: instance method,class method,static method

[ ]:
```
#syntax
class ClassName(object):
    def instance_method(self,passed_params): #instance method takes self as␣
 ↪first parameter
        #code to execute

    def __init__(self,passed_params):  #The constructor is an instance method
    #assignments
```

```
    @classmethod
    def class_method(cls,passed_params): #class method always take class␣
 ↪instance as a first parameter
        #code to execute
    @staticmethod
    def static_method(passed_params):  #only parameter passed
        #code to execute

class_instance = ClassName()
                                  #calling instance and class methods
class_instance.instance_method()

ClassName.static_method()  #calling static methods
```

[4]:
```
#ex: instance method
class Person:
    def __init__(self,name,age): #self parameter is reference to the current␣
 ↪instance of the class
                                  #and is used to access variables that belongs␣
 ↪to that class
        self.name = name
        self.age = age
    def myfunc(self):
        print("Hello my name is "+ self.name)
p1 = Person("John",36)
p1.myfunc()
```

```
Hello my name is John
```

[ ]:

Q2)Explain the concept of function arguments and parameters in python.

ANS:Function arguments are values passed to a function when it is called. Parameters are the variables inside the function's parantheses

[12]:
```
#ex
def add(x,y):  #x,y are arguments
    return x + y
result = add(5,3) # 5,3 are values called parameters
result
```

[12]:
```
8
```

[ ]:

Q3)What are different ways to define and call functions in python.

1)Positional arguments:The order in which we provide arguments uses the same order in function.

[1]:
```
#ex:
def add(a,b): #a,b positional arguments
    return a+b
add("pw","skills") #value and a and b
```

[1]:
```
'pwskills'
```

[4]:
```
add("skills","pw") #here though we change the values assigned to a and b but␣
 ↪a,b dont change its position.
                #we see at output whatever the values assigned to a,b may be␣
 ↪in order or not in order.
                #in short,values may change but positon of argument is fixed␣
 ↪it maintains the order.
```

[4]:
```
'skillspw'
```

2)Variable length argument:used when we don't know no. of arguments. it takes any no. of arguments

[6]:
```
#ex
def sum1(*args): #this is variable length argument of any size or length
    # astric(*) indicates length of argument is not fixed
    return args
sum1(1,2,3,4,5,6,7)
```

[6]:
```
(1, 2, 3, 4, 5, 6, 7)
```

3)Keyword argument:uses key value pair to send the arguments key = value

[10]:
```
#ex
def test1(**kargs): #if no of keyword argument is unknown use (**) before␣
 ↪parameter in functiojn
    return kargs
test1(a=100,b=120)
```

[10]:
```
{'a': 100, 'b': 120}
```

4)Default arguments:They have preset values when not to specify when calling function

[11]:
```
#ex
def sum1(b,c,a=0): #default arguments are given at end a=0
    return a+b+c
sum1(b=3,c=5) #no error because default a=0
```

[11]:
```
8
```

[ ]:
```

```

Q4)What is purpose of return statement in python function.

ANS:Return:1)returns value which we want to return from function to output.hence it returns the result at output. 2)It ends the execution of a function and returns control to calling function. 3)It returns result to calling function

[1]:
```
#ex
def add(a,b): #a=5,b=7 values enters in variables when it called from x,y
    c=a+b #values from x,y to a,b result is stored in c
    return c #c=12 it returns value to z
x = int(input("enter first number"))  #x=5 goes to a
y = int(input("enter second number")) #y=7 goes to b .
z = add(x,y) #calls the function  as values of x,y goes to a,b.
print("addition",z) #we get output of z
```

```
enter first number 5
enter second number 7

addition 12
```

[ ]:
```

```

Q5)What are iterators and how they differ from iterables.

ANS:1)iterables: object that can return elements one by one majorly in for loop 2)iteration:process of returning elements one by one 3)iterators:it is intermediate step for creating an abject that can be iterated.

[4]:
```
#ex:
l = iter([1,2,3,4]) #iterables as iter() as given list is iterable
next(l) # iterators as next()
```

[4]:
```
1
```

[5]:
```
next(l) #returns next value from list
```

[5]:
```
2
```

[6]:
```
next(l)
```

[6]:
```
3
```

[7]:
```
next(l)
```

[7]:
```
4
```

[8]:
```
next(l) #after iterator is exhausted stops iteration
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
```

```
Cell In[8], line 1
----> 1 next(l)

StopIteration:
```

[ ]:

Q6)Explain the concept of generators in python and how they are defined.

ANS:Generators:1)It is a function that returns an iterator that produces a sequence of values when iterated over. 2)It generates the result one by one instead of in one go. 3)useful when we want to produce large sequence of values but we don't want to store all of them in memory at once. syntax: def func_name(): #defining generators yield stmt #to generate value it uses yield keyword

[44]:
```
#ex on fibonacci series
def fib(n):
    a=0
    b=1
    for i in range(n):
        yield a
        a,b=b,a+b
```

[47]:
```
f=fib(100-0)
f
```

[47]:
```
<generator object fib at 0x77857c19e0a0>
```

[48]:
```
next(f)
```

[48]:
```
0
```

[49]:
```
next(f)
```

[49]:
```
1
```

[50]:
```
next(f)
```

[50]:
```
1
```

[51]:
```
next(f)
```

[51]:
```
2
```

[52]:
```
next(f) #continues till the end of the iteration
```

[52]:
```
3
```

[ ]:

Q7)What are advantages of using generators over regular functions.

ANS:1)In Regular function for example we take a list and give square of the list.It calculates square of values and returns in one go 2)So,memory usage is more it takes lot of execution time as If there is lot of computation as all the computation will be done first and then calculation will be written to one go.

[3]:
```
#ex on regular function
def sq_num(n):
    result = []
    for i in range (n):
        result.append(i**2)
    return result
sq_num(10)
```

[3]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

So,generators are used to generate result one by one instead in one go.hence,it is function that returns iterator produces a sequence of values when iterated over. yield keyword is used to create generator object.

[25]:
```
#ex on generator function
def sq_num_generator(n):
    for i in range(n):
        yield i**2
```

[26]:
```
sq_num_generators(10)
```

[26]:
```
<generator object sq_num_generators at 0x77857c19cc10>
```

[27]:
```
gen = sq_num_generators(10)
gen
```

[27]:
```
<generator object sq_num_generators at 0x77857c19d3f0>
```

[28]:
```
next(gen)
```

[28]:
```
0
```

[29]:
```
next(gen)
```

[29]:
```
1
```

[30]:
```
next(gen)
```

[30]:
```
4
```

[31]:
```
next(gen)
```

[31]: 9

[32]:
```
next(gen)
```

[32]: 16

[33]:
```
next(gen)
```

[33]: 25

[34]:
```
next(gen)
```

[34]: 36

[35]:
```
next(gen)
```

[35]: 49

[36]:
```
next(gen)
```

[36]: 64

[37]:
```
next(gen)
```

[37]: 81

[38]:
```
next(gen)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
Cell In[38], line 1
----> 1 next(gen)

StopIteration:
```

[ ]:

Q8)What is lambda function in python and when it is typically used.

ANS:1)lambda function is anonymous(because lambda function is defined without any name using keyword lambda), short hand function(Two line of code converted in a single line).

2)syntax: lamba arguments:expression

3)ex: f= lambda a:a*a f=function object accepts and stores result of expression lambda=keyword a=argument(can have multiple arguments) a*a= has only one line expression no more allowed

4)use case:when we don't want to write return statement and def keyword so to define a function in proper and regular way we use short hand function used for short operation/data manipulation.reduces readibility of code

[4]:
```
#ex on def defined function
def sq_num(x):
    return x**2
sq_num(2)
```

[4]: 4

[5]:
```
#ex on lambda function
sq_lambda = lambda x:x**2
sq_lambda(2)
```

[5]: 4

[6]:
```
#ex
add = lambda a,b:a+b
add(2,5)
```

[6]: 7

[ ]:
```

```

Q9)Explain the purpose and usage of map() function in python.

ANS:1)map function executes a specified function for each element in an iterable.The item is sent to function as parameter.

2)syntax: map(function,iterables) function=function to execute/applied for each element in iterable iterables=collection of elements or objects to be iterated.function has one parameter for each iterable ex list tuple etc we can pass more than one iterable to map function.

3)purpose:map in python works as an iterator to return result after applying a to function to every element of an iterable(tuple,list) usage:It is used when you want to apply a singlr transformation function to all the iterable elements.here,the iterable and function are passed as arguments to the map in python.

[10]:
```
#ex
def add(x):
    return x+10
list(map(add,l))
```

[10]: [11, 12, 13, 14, 15]

[ ]:
```

```

Q10)What is the difference between map(),reduce(),filter() functions in python.

ANS:1)map() function:It executes a specified function for each of elements of an iterable syntax: map(function,iterables)

[16]:
```
#ex
l=[1,2,3,4,5]
list(map(lambda x:x**2,l))
```

[16]:
```
[1, 4, 9, 16, 25]
```

2)Reduce() function:It means folding and reduction.It is not python keyword we have to import it from library called functools. syntax: reduce(function,iterables)

[11]:
```
#ex
from functools import reduce
```

[12]:
```
l=[2,1,3,4,4,5,6]
reduce(lambda x,y:x+y,l) #it folds all the given data from given list at ouput␣
 ↪gives a single element not list.
```

[12]:
```
25
```

3)filter() function:It is used to filter elements from an iterable based on same condition. syntax: filter(function,iterables)

[13]:
```
#ex
l=[2,1,3,4,4,5,6]
list(filter(lambda x:x%2==0,l)) #even:at output filters only even numbers
```

[13]:
```
[2, 4, 4, 6]
```

[15]:
```
list(filter(lambda x:x%2!=0,l)) #odd:at ouput filters ony odd numbers
```

[15]:
```
[1, 3, 5]
```

[ ]:
```

```