1 What is the difference between a function and a method in Python?

```
#A function is a block of code or statements that performs a specific task and runs only if it is called .
# There are mainly two types of functions:
#(a).built-in functions: these are predefined functions in python that can be directly used.
#(b).User-defined functions: They are not predefined functions. Here the user nedds to call the function inorder to run the specific nee

#for examples:
def func():
  return "hello! welcome on board "
func()+"Praveen"
```

```
'hello! welcome on board Praveen'
```

```
def sqr(a):
  sqr=a*a
  return sqr
sqr(5)
```

```
25
```

```
a=int(input("entera number over here: "))
def sqr(a):
  sqr=a*a
  return sqr
sqr(a)
```

```
entera number over here: 12
144
```

```
#METHOD:
#Python methods has various uses:

# Methods in Python are used to define the behaviour of the Python objects.
# Methods are used to improve the readability and maintainability of code.
# They help in breaking down complex tasks into smaller, more manageable tasks
```

2. Explain the concept of function arguments and parameters in Python.

```
# In Python, an argument is the value passed to a function when it's called. Fundamentally, parameters are the variables inside a functi
#Function Arguments: In programming we utilize functions to organize our code and make the code easy and simpler to use as well understa

#so, let's take an example and undersstand it briefly:
```

```
def my_fun(fname,lname):
  print(fname + " " + lname)
my_fun("praveen","kumar")
```

```
praveen kumar
```

3. What are the different ways to define and call a function in Python?

```
#To define a function, you use the def keyword followed by the name of the function and parentheses (). If the function takes any argume


#example:
def greet(name):
  print("Hello, " + name + "! how are you ?")
#In this function we defined a function called greet that takes one argument called name . the function then prints out a greeting messa
```

```
# To call a function:
#Once you have defined a function, you can call it in your code as many times as you need.

# To call a function in Python, you simply type the name of the function followed by parentheses (). If the function takes any arguments
```

```
#example:
greet("mandy")
```

```
Hello, mandy! how are you ?
```

```
greet("pollard")
```

```
Hello, pollard! how are you ?
```

```
greet("ramandino")
```

```
Hello, ramandino! how are you ?
```

4. What is the purpose of the `return` statement in a Python function?

```
# A return statement is used to end the execution of the function which is called.
#The statements afteer return statements are not executed.If the return
# statement is written without any expression,then special value None is returned .
```

```
#"Return statement cannot be used outside the function"
```

```
#example:
def fun(x,y):
  mul=x*y
  return mul
fun(2,3)
```

```
6
```

5. What are iterators in Python and how do they differ from iterables?

```
#iterable: An iterable is basically an object that any user can iterate over.
#method used: We can generate an iterator when we pass the object to the iter()method.
#inter-relation:Every iterator is basically iterable.

#Iterator: an iterator is also an object that helps a user in iterating over that helps a user in iterating over another object (that is
#method used:we use the next()method for iterating.This method helps iterators return the next item available from the object.
#Inter-relation: not every iterable is an object.
```

```
iter_list=iter([1,2,3,4,5])
print(next(iter_list))
```

```
1
```

```
print(next(iter_list))
```

```
2
```

```
print(next(iter_list))
#like this,you have to manually do and it will go till the last.
```

```
3
```

6. Explain the concept of generators in Python and how they are defined.

```
#  A generator is a function that returns an iterable set of values. Generators avoid creating a list or other data structure in memory
```

```
def countdown(a):
  while a>0:
    yield a
    a-=1
gen=countdown(10)
for i in gen:
  print(i)
```

```
10
9
8
7
6
5
4
3
2
1
```

7. What are the advantages of using generators over regular functions?

```
#efficient: Python Generators are more efficient than loops for large datasets, as they produce values one by one instead of storing the
#memory management: As generators produce values one by one rather than storing them in a list or other data structure, they also requir
#time saving:Generators in Python can also save time as they do not need to wait for the entire sequence to be generated before returnir
#infinite sequencesGenerators can produce infinite sequences, which is helpful for tasks such as stream processing or other activities r
#flexibility:As generators can produce a sequence of values over time, they are incredibly versatile and can be used in various applicat
```

```
#example:
def generator_function():
  for i in range (10):
    yield i
gen=generator_function()
for j in gen:
  print(j)
```

```
0
1
2
3
4
5
6
7
8
9
```

8. What is a lambda function in Python and when is it typically used?

```
#A lambda function is a anonymous function which is used when the code is to be written in a single line.
#it is used to make a bulky code to a single line code and it reduces the clumsyness in a particular code
# below i'm mentioning two codes which can be written in both paras:
```

```
#WAP code to check ehether enterred no. is even or not:
#normal code using coditional:
n=int(input("enter a number : "))
if n%2==0:
  print("even")
else:
  print("odd")
```

```
enter a number : 24
even
```

```
even=lambda x:x%2==0
even(2)
```

```
True
```

```
odd=lambda x:x%2!=0
odd(3)
```

```
True
```

```
even(3)
```

```
False
```

```
odd(4)
```

```
False
```

```
#Inorder to sort a function based on length of the characters entered in it.
y=["python ","java ","c ","c++"]
sorted(y,key=lambda y:len(y))
```

```
['c ', 'c++', 'java ', 'python ']
```

9. Explain the purpose and usage of the `map()` function in Python.

```
# Map in Python is a function that works as an iterator to return a result after applying a function to every item of an iterable (tuple
#the map() function executes a specified function fo r each item in an iterable.
```

```
n=[1,2,3,4,5,6]
sq=map(lambda x:x*x,n)
print(list(sq))
```

```
[1, 4, 9, 16, 25, 36]
```

10. What is the difference between `map()`, `reduce()`, and `filter()` functions in Python?

```
# These three operations are paradigms of functional programming. They allow one to write simpler, shorter code without needing to bother
# The map () function returns a map object(which is an iterator) of the results after applying the given function to each item of a given
#syntax:
#map(function,iterable)
#fun: it is a function to which map passes each element of given iterable
#iter:iterable object to be mapped.
```

```
#example:
n=[5,10,15,20]
double=map(lambda x:x*2,n)
print(list(double))
```

```
[10, 20, 30, 40]
```

```
m=[2,4,6,8,10]
list(map(lambda x:x+10,m))
```

```
[12, 14, 16, 18, 20]
```

```
#reduce
#It is not a python function you need to import it from function tools>>It is a basic function which is used to store data in small memo
#basically,it is widely used in the mathemeatical functions inorder to return only values and store the data which is useful to us.

#syntax:
reduce(func,*iterables)
```

```
from functools import reduce
n=[1,2,3,4,5]
reduce (lambda x,y:x+y,n)
```

```
15
```

```
m=[6,7,8,9,10]
reduce(lambda x,y:x*y,m)
```

```
30240
```

```
reduce(lambda x,y:x if x>y else y,m)
```

```
10
```

```
#Filter():
#syntax of filter:
filter(func,*iterables)
#it is used to filter the components whivh are present on a list.As the name itself suggest filter .
#it is used to filter large data sets in order to complete our tasks with short span of time.
```

```
o=[1,2,3,4,5,6,7,8,9,10,11,12,13,14]
list(filter(lambda x:x%2==0,o))
```

```
[2, 4, 6, 8, 10, 12, 14]
```

```
list(filter (lambda x:x>10,o))
```

```
[11, 12, 13, 14]
```

Double-click (or enter) to edit

PRACTICAL QUESTIONS

```
#1. Write a Python function that takes a list of numbers as input and returns the sum of all even numbers in
the list
```

```
def iseven(a):
  sum=0
  for i in a:
    if i%2==0:
      sum=sum+i
  print(sum)
```

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

```
2
6
12
16
18
24
```

```
#2. Create a Python function that accepts a string and returns the reverse of that string.
```

```
def reverse (s):
  return s[::-1]

reverse("pwskills")
```

```
'sllikswp'
```

```
# 3. Implement a Python function that takes a list of integers and returns a new list containing the squares of each number.
```

```
def sqr(l):
  s=[]
  for i in l:
    s.append(i*i)
  return s
```

```
sqr([1,252,224,1,3,5,41,2,55,556])
```

```
[1, 63504, 50176, 1, 9, 25, 1681, 4, 3025, 309136]
```

```
# 4. Write a Python function that checks if a given number is prime or not from 1 to 200.
```

```
def prim(num):
  if num<=1 and num>=202:
    return false
  if num==2 or num==3:
    return true
  for i in range(2,int(num**0.5)+1):
    if num%i==0:
      return False
  return True
num=int(input("enter a number between(1-200)"))
if prim (num):
  print(f"{num} is a prime number")
else:
  peint(f"{num} is not a prime numbner")
```

```
enter a number between(1-200)23
23 is a prime number
```

```
# 5.Create an iterator class in Python that generates the Fibonacci sequence up to a specified number of terms.
def sq_num_generator(n):
  a=0
  b=1
  c=0
  for i in range (n):
    yield a
    c=a+b
    a=b
    b=c
gen=sq_num_generator(10)
```

```
next(gen)
```

```
0
```

```
next(gen)
```

```
0
```

```
next(gen)
```

```
1
```

```
next(gen)
```

```
1
```

```
next(gen)
```

```
2
```

```
next(gen)
```

```
3
```

```
next(gen)
```

```
5
```

```
next(gen)
```

```
8
```

```
next(gen)
```

```
13
```

```
next(gen)
```

```
21
```

```
next(gen)
```

```
34
```

```
#see at last it gives error because the entry you provided is out of the given input you provided to it .
next(gen)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-87-6e72e47198db> in <cell line: 1>()
----> 1 next(gen)

StopIteration:
```

```
#7. Implement a generator function that reads a file line by line and yields each line as a string


def read_file(filename):
  with open(filename) as f:
    for line in f:
      yield line
read=read_file("praveen.txt")
```

```
# 8. Use a lambda function in Python to sort a list of tuples based on the second element of each tuple


tuples_list = [(1, 3), (4, 1), (2, 2), (5, 0)]

# Sort using a lambda function
tuples_list.sort(key=lambda x: x[1])

print(tuples_list)
```

```
[(5, 0), (4, 1), (2, 2), (1, 3)]
```

```
# 9. Write a Python program that uses `map()` to convert a list of temperatures from Celsius to Fahrenheit


l=[1,2,3,4,5]
list(map(lambda x:(x*9/5)+32,l))
```

```
[33.8, 35.6, 37.4, 39.2, 41.0]
```

```
# 10. Create a Python program that uses `filter()` to remove all the vowels from a given string.


def test(a):
  vowels="aeiouAEIOU"
  ann = filter(lambda x:x not in vowels ,a)
  for i in ann:
    print(i)
```

```
test("praveen")
```

```
p
r
v
n
```

```
test("pw skills")
```

```
p
w

s
k
l
l
s
```

```
test("to become a data analyst is my dream and i'm sur that i will become the expert of this domain very soon")
```

```
t

b
c
m


d
t
```

```
n
l
y
s
t

s

m
y

d
r
m

n
d

'
m

s
r

t
h
t


w
l
l

b
c
m

t
h

x
p
r
t

f

t
```

```
# 11) Imagine an accounting routine used in a book shop. It works on a list with sublists, which look like this:
# Write a Python program, which returns a list with 2-tuples. Each tuple consists of the order number and the product of the price per i
# Write a Python program using lambda and map
```

```
# Sample data
orders = [
    [34587, "Learning Python, Mark Lutz", 4, 40.95],
    [98762, "Programming Python, Mark Lutz", 5, 56.80],
    [77226, "Head First Python, Paul Barry", 3, 32.95],
    [88112, "Einführung in Python3, Bernd Klein", 3, 24.99]
]

# Lambda function to calculate the order total and apply the condition
calculate_total = lambda order: (order[0], order[2] * order[3] + 10 if order[2] * order[3] < 100 else order[2] * order[3])

# Apply the function to each order using map
result = list(map(calculate_total, orders))

# Print the result
print(result)
```

```
[(34587, 163.8), (98762, 284.0), (77226, 108.85000000000001), (88112, 84.97)]
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

l = [47, 11, 42, 13]

Output using reduce function

list (reduce (lambda x, y : x+y, l))

output :→ 113

Internal Mechanism:

$$\begin{tikzcd}
l = [\, 47 \arrow[d] & 11 \arrow[d] & 42 \arrow[ddd, "" , to=4-2] & 13\,] \arrow[dddd] \\
x + & y & & \\
58 \arrow[d] & & & \\
x + & y \Rightarrow & 100 \arrow[d] & \\
& & x + & y = 113
\end{tikzcd}$$