Que1- What is ythe difference between a function and a method in python?

Ans- Functions are independent blocks of code that can be called from anywhere, wheras methods are tied to objects or classes and need an object or class instance to be invoked. Example of functions: # defining a function def add_num(a,b) #calling a functon add_num(3.5) # output 8

Exampe of method:

Que2- Explain the concept of function arguments and parameters in python.

Ans Function arguments are values that you provide to a function when you call it. These arguments can be used by the function to perform specific tasks. A parameter is the variable defined within the parentheses when we declare a function. Example - # function to find square of a number def square(a): # where 'a' is parameter return a**2 square(6) # where 6 is an argument * #output* 36

Que3- What are the different ways to define and call a function in python ?

Ans The differnt ways to didine and call a function in python are-

1. Basic function definition and call:- This is the most common way to define and call a function. Example- # Function definition def greets(): print("Hello") # Function call greets()

- #Output* Hello

2. Function with arguments:- Example - #Function call def sum (a,b): return a+b * # Function call* sum(4,6)

#output 10

3. Function with variable length arguments:- Variable length arguments refer to the ability of function to accept variable number of parameters. Example- #Function definition def 1 sum1(*args): sum = 0 for i in args: s = s+i return s # Function call sum1(1,4,7) # output 12
4. Lambda function :- These are small anonymous function defined by using lambda keyword. Example- square_lambda = lambda x: x**2 square_lambda(3) * #output* 9

Que4- What is the purpose of the 'return' statement in a python function ?

Ans The purpose of the 'return' statement in a python function to give the output of a function by ending the execution of the function call. Example - def func(): calc = 2+3+65 return calc func() # output* 35

Que5- What are iterators in python and how they differ from iterables ?

Ans Iterators in python are an object representing a stream of data and return the data one by one. They are differ from iterables as an iterable is basically an object that any user can iterate over. Every iterator is basically iterable but not vice-versa. Example- s ="book" # string is iterable a = iter(s) # iterator object next(a) 'b' next(a) 'o' next(a) 'o' next(a) 'k'

Que6- Explain the concept of generators in python and how they are defined.

Ans Generator functions in python are a way to create iterators in a more concise and memomryefficient manner. These functions allow you to iterate over a potentially large sequence of data without loading the entire sequence into memory. Here , instead of using return to send a value back to the caller, generator function uses 'yield'. def pw_generator(): yield 1 yield 2

using the generator generator = pw_generator()

print(next(generator)) *# output1* print(next(generator)) *#output2* Example- def countdown(n): while n>0: yield n n -= 1

using the generator for a countdown

countdown_gen = countdown(5) for number in countdown_gen: print(number)

output : 5,4,3,2,1

Que7- What are advantages of using generators over regular functions ?

Ans The advantages of using generators over regular functions is that the generators use 'yield' keyword to produce a series of values over time , one at a time , rather than returning a single result which helps in usage of memory will be less. Example - # Execution by regular

function def square_number(n): result =[] for i in range(n): result.append(i**2) return result square_number(10) # output: [1,4,9,16,25,36,49,64,81] *# Execution by generator function * def square_number_generators(n): for i in range(n): yield i**2 gen = square_number_generators(10) next(gen) # output:1 next(gen # output:4 2 ... next(gen) # output :81

que8- What is lambda function in python and when is it typically used ?

Ans lambda function in python is a concise way to create anonymous functions, also known as lambda expressions. Lambda functions are often used for short- term operations and are defined in a single line. Example - add = lambda x,y:x+y result = add(5,6) print(result)

outpu:11

Que9- Explain the purpose and usage of the ’map()’function in python.

Ans The purpose and usage of the ‘map()’ function in python is to execute a specified function for each item in an ‘iterable’ and the item is sent to the function as a parameter. It starts by applying the function to the first items , then second items and so on. Example- num = [1,2,3,4] square_num = list(map(lambda x:x**2,num)) print(square_num) #output : [1,4,9,15]

Que10- What is the difference between ‘map()’ , ‘reduce()’ and ’filter()’functions in python ?

Ans- The differnce between ‘map()’ , ‘reduce()’ and ‘filter()’ functions in python are- • Map() - Executes a specified function for each of item of an iterable. Example: num_as_strings = [“2”,“4”,“6”,“8”] num_as_integers = list(map(int,num_as_strings)) print(num_as_integers) #output: [2,4,6,8] • Reduce() - It implements a mathematical technique commonly known as ‘folding’ or ‘reduction’. It is a process where you reduce a list of items to a single value(cumulative value). Example: from functools import reduce num = [1,2,3,4] total_sum = reduce(lambda(x,y:x+y, num)) print(total_sum) #output: 10 • Filter() - It is used to filter elements from an iterable based on some condition. Example: l1 =[-1,-2,3,4,5] list(filter(lambda x: x<0,l1))

output: [-1,-2]

```
A# write a python function that takes a list of numbers as input and returns the sum of all even numbers in the list.
def sum_even_numbers(num):
  sum = 0
  for numbers in num:
    if numbers % 2 == 0:

      sum += numbers
  return sum


sum_even_numbers([1, 5, 6, 7, 8, 9, 10, 11, 12])
```

```
36
```

```
# create a python function that accepts a string and returns the reverse of that string.
def reverse_string(input_string):
    return input_string[::-1]

result = reverse_string("Hello, World!")
print(result)
```

```
!dlroW ,olleH
```

```
#Implement a python that takes a list of integrs and return a new list containing the squares of each numbers.
def square_numbers(numbers):
    squared_numbers = []
    for num in numbers:
        squared_numbers.append(num ** 2)
    return squared_numbers


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

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

```
# write a python function that checks if a given number is prime or not from 1 to 200.

def is_prime(n):
    for num in range(1, 201):
      if n <= 1:
        return False
      for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
          return False
      return True


is_prime(197)
```

```
True
```

```
# creata a iterator class in python that generate the fibonacci sequence up to a specified number of terms.
```

```
# write a generator function in python that yields the powers of up 2 to given exponent.
def power_of_two(n):
    power = 1
    for _ in range(n):
        yield power
        power *= 2

for power in power_of_two(10):
    print(power)
```

```
1
2
4
8
16
32
64
128
256
512
```

```
# Implement a generator function that reads a file line by line and yields each line as a string
def read_file_line_by_line(file_path):
    """Generator function to read a file line by line."""
    with open(file_path, 'r') as file:
        for line in file:
            yield line.strip()

# Example usage:
for line in read_file_line_by_line('example.txt'):
    print(line)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-34-2446629247.py in <cell line: 0>()
      7
      8 # Example usage:
----> 9 for line in read_file_line_by_line('example.txt'):
     10     print(line)

/tmp/ipython-input-34-2446629247.py in read_file_line_by_line(file_path)
      2 def read_file_line_by_line(file_path):
      3     """Generator function to read a file line by line."""
----> 4     with open(file_path, 'r') as file:
      5         for line in file:
      6             yield line.strip()

FileNotFoundError: [Errno 2] No such file or directory: 'example.txt'
```

```
# use a lambda function in python to sort a list of tuples based on the second element of each tuple.
# Sample list of tuples
data = [(1, 'banana'), (2, 'apple'), (3, 'cherry')]

# Sorting the list of tuples based on the second element
sorted_data = sorted(data, key=lambda x: x[1])

print(sorted_data)
```

```
[(2, 'apple'), (1, 'banana'), (3, 'cherry')]
```

```
# write a python program that uses 'map()' to convert a list of temperatures from celesius to fahrenheit
 # Function to convert Celsius to Fahrenheit
def celsius_to_fahrenheit(celsius):
    return (9/5) * celsius + 32

# List of temperatures in Celsius
celsius_temps = [0, 20, 37, 100]
```

```
# Using map() to convert Celsius to Fahrenheit
fahrenheit_temps = list(map(celsius_to_fahrenheit, celsius_temps))

# Print the results
print("Celsius temperatures:", celsius_temps)
print("Fahrenheit temperatures:", fahrenheit_temps)
```

```
Celsius temperatures: [0, 20, 37, 100]
Fahrenheit temperatures: [32.0, 68.0, 98.60000000000001, 212.0]
```

```
# create a python program that uses 'filter()' to remove all the vowels from a given string.
def remove_vowels(input_string):
    vowels = 'aeiouAEIOU'
    return ''.join(filter(lambda x: x not in vowels, input_string))

# Example usage:
test_string = "Hello World"
string_without_vowels = remove_vowels(test_string)
print(f"Original string: {test_string}")
print(f"String without vowels: {string_without_vowels}")
```

```
Original string: Hello World
String without vowels: Hll Wrld
```

```
# imagine an accounting routine used in a book shop. It works on a list with sublists , which look like this:
```