FUNCTION

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

→ \*Function- A function is used while using the ‘def’ function and is not tied to any object.

Called independently, often with arguments.

Example: def greet(name):

return f”Hello, {name}”

print(greet(“Laxmi”))

\*Method- A method is a function that is associated with an object (usually a class instance).

Called on an object using dot notation (object.method()).

Example: def greet(self, name):

Return f”Hello, {name}”

greeter = Greeter()

print(greeter.greet(“Bob”))

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

→ \*Functions Arguments in Python

a)Positional Arguments

-Matched to parameters by position.

Ex- def add(a, b):

return a + b

add(2, 3)

a = 2, b = 3

b)Keyword Arguments

-Passed with parameter names, allowing order to change.

Ex- add(b = 3, a =2)

a = 2, b = 3

c)Default Arguments

-Parameters can have default values.

Ex- def greet(name=”Guest”):

print(f”Hello, {name}”)

greet(“Laxmi”)

d)Variable-length Arguments

-Allows passing an arbitrary number of arguments.

args for non-keyword (positional) arguments:

def total(\*args):

return sum(args)

total(1, 2, 3)

#Returns 6

Kwargs for keyword arguments:

def show\_info(\*\*kwargs):

for key, value in kwargs.items():

print(f”{key}: {value}”)

show\_info name(name = “Laxmi”, age = 25)

\*Functions Parameters in Python

a)Positional(Required) Parameters

-Values are passed based on position.

Ex- def greet(name, age):

print(f”{name} is {age} years old.”)

greet(“Laxmi”, 25)

b)Default Parameters

-Parameters that have a default value.

Ex- def greet(name=”Guest”):

print(f”Hello, {name}”)

greet(“Laxmi”)

c)Keyword Parameters

-Order doesn’t matter when using keywords.

Ex- def describe(name, job):

print(f”{name} is a {job}”)

d)Variable - length Positional Parameters (\*args)

-Used to accept any number of positional arguments as a tuple.

Ex- def add(\*numbers):

return sum(numbers)

add(1, 2, 3, 4)

# Returns 10

e)Variable -length keyword Parameters(\*\*kwargs)

-Used to accept any number of keyword arguments as a dictionary.

Ex- def show\_info(\*\*details):

for key, value in details.items():

print(f”{key}: {value}”)

show\_info(name = “Alice”, age =30)

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

→ Defined using a ‘def’ keyword

def greet(name):

print(f”Hello, {name}”)

\*Function with default parameters

def greet(name=”Guest”)

print(f”Hello, {name}”)

\*Variable- length arguments

positional(\*args)

def add(\*numbers):

return sum(numbers)

keyword(\*\*kwargs)

def show\_info(\*\*info):

for key, value in info.items():

print(f”{key}: {value}”)

\*Lambda Function

square = lambda x: x \* x

Calling Functions in Python

\*Positional Arguments

def greet(name, age):

print(f”{name} is {age} years old.”)

greet(“Alice”, 30)

\*Keyword Arguments

greet(age=30, name=”laxmi”)

\*Mixed Arguments

greet(“charlie”, age=25)

\*Using \*args and kwargs

def greet(\*args, \*\*kwargs):

print(“Args:”, args)

print(“Kwargs:”, kwargs)

greet(“Hello”, “world”, name=”Laxmi”, age=30)

\*Calling Lambda Function

add = lambda x, y: x + y

print(add(2, 3))

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

→ \*Send Output Back to Caller

It ends the function and gives back a value.

def add(a, b):

Return a + b

result = add(2, 3)

print(result)

#output: 5

\*Exit a Function Early

It can be used to stop the function before reaching the end.

def check\_even(n):

If n % 2 == 0:

return True

return False

\*Return Multiple Values

Python allows returning multiple values as a tuple.

def get\_name\_age():

return “Laxmi”, 25

Name, age = get\_name\_age()

\*Default Return Value is None

If no return is used, the function returns None.

def greet():

print(“Hello”)

result= greet()

print(result)

#Output: None