PW Assignment – 19th Sept 2025

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

Ans >

Function	**Method**
Block of reusable code which will do specific task.	Method is function which is associated with a Class or Objects
Defined using def or lambda outside a class	Defined using def inside class
Called using func()	Called using obj.method()
def greet(name): return "Hi " + name greet("Ankur")	def greet(self, name): return "Hi " + name (in a class) obj.greet("Ankur")
No access to object attributes	Can access and modify object's attributes

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

Ans> Parameter : A variable in the function definition that receives a value.

def sum(a ,b) -> a , b is parameter

return a +b

Argument : The actual value passed to the function when it is called .

Sum(9,8) -> 9,8 are the arguments for sum function.

Parameters are placeholders.

Arguments are actual values.

Rules

Required arguments first

Then *args (positional variable-length)

Then default arguments

Finally **kwargs (keyword variable-length)

E.g

```
def show_info(name, *hobbies, country="India", **details):
   print(f"Name: {name}")
   print(f"Hobbies: {hobbies}")
   print(f"Country: {country}")
   print(f"Other Details: {details}")
show_info(
   "Ankur",
   "Reading", "Yoga",
   country="USA",
   age=25,
   city="Mumbai"
)
```

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

Ans> function in python is define using def keyword.

1.def sum(a, b)
    Return a +b

Sum(9,8) #O/P – 17

2.Function with default parameters
    def sum(a, b = 5)
    Return a +b

    Sum(10) #O/P – 10 + 5 = 15
    Sum(10,9) #O/p – 19

3.Function with variable length

    def sum_all(*args)

        return sum(args)

    sum_all(1,2,3,4) #O/P – 10

4.Keyword argument

    def full_info(**kwargs):

        return kwargs

```
full_info(name = ‘Ankur’, city = "Kolkata", age = 33)

#O/P -  { name : ‘Ankur’, city : " Kolkata ", age : 33}
```

5. Define and call

```
square = lambda x: x ** 2

square(5)  O/P – 25
```

6. positional keyword

```
def info(name, age):
        return f"{name} is {age} years old"


info("John", 25)         # Positional

info(age=25, name="John")  # Keyword
```

7. Functions can be passed around like variables.

```
l =[1,2,3,4,4,5,6]

Reduce(lambda x,y : x+y ,l)                  O/P - 25
```

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

Ans> The return statement in Python is used to exit a function and send a result back to the caller. When a function reaches a return statement, it stops executing and returns the specified value. If no value is provided, the function returns None by default. Additionally, return can be used to exit a function early or to return multiple values at once as a tuple.

```
def square(x):
        return x * x

result = square(4)

print(result)  # Output: 16
```

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

Ans> Iterables : An Iterable is any python Objects/sequential Str/data that is capable of return number one at a time .

E,g. Lists ([1, 2, 3]),Tuples ((1, 2)),Strings ("hello").Dictionaries ,Sets.

Iterators : An iterators is an objects representing a stream of data return the date one by one .

Iter(Object) – return an object is iterator or not .

```
nums = [10, 20, 30]   # This is an **iterable**
it = iter(nums)      # Convert iterable to an **iterator**
print(next(it))      # Output: 10
print(next(it))      # Output: 20
print(next(it))      # Output: 30
print(next(it))      # Raises StopIteration
```

next() – will give element of and object until value is present.

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

Ans> A generator is a special type of iterator that lets you generate values on the fly, instead of storing them all in memory at once.

Generators are useful when you're working with large data sets or streams of data and want to save memory and processing time.

Yield function uses return statement but a generator function use "Yield" state .

```
def square_num(n):
    for i in range(1,n):
        yield i**2
a = square_num(4)
```

next(a) = 1

next(a) = 4

next(a) = 9

next(a) = StopIteration

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

Ans>

Generators	Regular Functions
Use less memory by yielding one item at a time	Store all results in memory at once
Evaluate values only when needed	Compute all values immediately
Faster for large or infinite data streams	Slower for large datasets due to full computation
Can represent infinite sequences safely	Not suitable for infinite sequences (causes memory issues)
Cleaner and easier to manage state with yield	Requires complex state management and data structures
Can pause (yield) and resume execution	Executes all at once and exits
Directly iterable (can be used in for loops)	Must return iterable objects explicitly
Quick startup (no full list creation)	Slower start due to full list or data preparation
Ideal for large files, streams, or pipelines	Not memory-efficient for large or streaming data

def gen_nums(n):
for i in range(n):
yield i # yields one value at a time
	def list_nums(n):
	return [i for i in range(n)] # builds entire list in memory

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

Ans> A lambda function in Python is a small, anonymous (unnamed) function defined using the lambda keyword instead of def. It's typically used when you need a quick function for a short period of time — often as an argument to another function.

Syntax : lambda arguments : epression

e.g., square = lambda x: x**2

square(2) = 4

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

Ans> Map function execute a specific function for each of items of an iterables.

Syntax: map(func ,*iterables)

```
def square(x):
    return x**2
lst = [1,2,3,4]
```

map(square,lst) – O/P = [1,4,9,16]

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

Ans> map() – Transforms each item

- Purpose: Applies a function to every element in an iterable and returns a new iterable with the results.
- Returns: A map object (you can convert it to list, tuple, etc.)

```
nums = [1, 2, 3, 4]
squared = list(map(lambda x: x**2, nums))
print(squared)  # Output: [1, 4, 9, 16]
```

reduce() – Reduces all items to a single value

- **Purpose:** Repeatedly applies a function to **accumulate** a result from all items.
- **Returns:** A single result (not a list)

You must import it from functools

```
from functools import reduce

nums = [1, 2, 3, 4]
product = reduce(lambda x, y: x * y, nums)
print(product)  # Output: 24
```

filter() – Filters items based on condition

- **Purpose:** Applies a function that **returns True/False** to each item and **keeps only the items where the function returns True**.
- **Returns:** A filter object (convert to list() to see results)

```
nums = [1, 2, 3, 4]

evens = list(filter(lambda x: x % 2 == 0, nums))

print(evens)  # Output: [2, 4]
```

11. Using pen & Paper write the internal mechanism for sum operation using reduce function on this given list:[47,11,42,13];