**Assignments:**

**1.What is a lambda function in Python?**

**A.** A lambda function is a small function which can take any arguments and one expression.

a = lambda x,y:x+y

a(3,4) #return 3+4 = 7

**2.How is the 'map' function used in Python? Provide an example.**

**A.** map() function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

syntax : map(function,iterable)

example,

def square(number):

return number\*\*2

map\_obj = map(square, [1,2,3])

list(map\_obj)

# return [1,4,9]

**3.Explain the 'reduce' function and provide an example of its usage in Python.**

A. The reduce function in Python is a built-in function that takes a function and an iterable as arguments and returns a single value by applying the function to the elements of the iterable. The function must take two arguments and return one value. The reduce function is defined in the functools module.

from functools import reduce

examples,

def add(x, y):

return x + y

numbers = [1, 2, 3, 4, 5]

sum = reduce(add, numbers)

print(sum) # 15

The reduce function can also take an optional third argument, which is the initial value to be used as the first argument to the first function call. For example, if you want to calculate the product of a list of numbers, you can use the reduce function with an initial value of 1:

def multiply(x,y):

return x\*y

numbers = [1, 2, 3]

product = reduce(multiply, numbers, 1)

print(product) # 6

**4.Write a Python function to filter even numbers from a list using the 'filter' function.**

**A.** A python function to filter numbers from a list using the “filter” function-

list1 = [1,2,3,4,5,6,7,8,9]

evn\_num = filter(lambda x: x%2 == 0,list1)

for i in evn\_num:

print(i)

# 2

# 4

# 6

# 8

**5.What is the purpose of the 'yield' keyword in Python? Provide an example of a generator function using 'yield'.**

**A.** the purpose of "yield" keyword is to create generator function which is a special type of function that returns a iterator object.

when a generator function is called, it does not execute the body of the function immediately, but only create a generator object.

when the generator object is iterated over, the function body is executed until "yield" keyword is encountered.

A generator function uses the ‘yield’ keyword instead of the ‘return’ keyword to return a value from the function. The ‘yield’ keyword also pauses the execution of the function and saves its state, so that it can resume from where it left off when the next value is requested.

example,

def fibonacci():

# initialize the first two numbers of the sequence

a = 0

b = 1

# loop indefinitely

while True:

# yield the current value of a

yield a

# update a and b to the next two numbers of the sequence

a, b = b, a + b

# create a generator object by calling the generator function

fib = fibonacci()

# iterate over the generator object using a for loop

for i in range(10):

# print the next value generated by the generator object

print(next(fib))

**6.How is recursion used in Python functions? Provide an example to illustrate its use.**

**A.** Recursion is the process of defining something in terms of itself. In Python, recursion is a technique where a function calls itself directly or indirectly to solve a problem by breaking it down into smaller sub-problems.

def my\_recur(x):

    if x>0:

## my\_recur() func is calling itself inside the function

        result = x+ my\_recur(x-1)

        print(result)

    else: result = 0

    return

my\_recur(5)

# 1

# 3

# 6

# 10

# 15

**7.Explain the concept of a higher-order function with respect to Python functions.**

**A.** A higher-order function is a function that meets one or more of the following criteria:

* It accepts other functions as arguments.
* It returns a function as a result.
* It creates a new function within its body.

Higher-order functions are useful for abstraction and code reuse in Python. They allow us to modify the behavior of other functions, or create new functions based on existing ones. Some examples of higher-order functions in Python are:

* map, filter, and reduce, which apply a function to a sequence of elements and return a new sequence.
* sorted, min, and max, which take a function as a key parameter and use it to compare elements.
* decorator, which is a function that takes another function as an argument and returns a modified version of it.

def create\_adder(x):

def adder(y):

return x + y

return adder

add\_10 = create\_adder(10) # create a new function that adds 10

print(add\_10(5)) # 15

print(add\_10(20)) # 30

**8.Compare and contrast the 'map' and 'filter' functions in Python.**

**A.** The ‘map’ and ‘filter’ functions are both examples of higher-order functions in Python, which means they can take other functions as arguments and return functions as results. They are also both used to apply a function to each element of an iterable, such as a list or a tuple, and produce a new iterable. However, they have some differences in how they work and what they return.

The ‘map’ function is used to apply a function to each element of an iterable and return a new iterable with the transformed elements, while the ‘filter’ function is used to apply a function to each element of an iterable and return a new iterable with only the elements that satisfy the function.

For example:

# Using the 'map' function to convert a list of strings to uppercase

names = ["Alice", "Bob", "Charlie"]

upper\_names = map(str.upper, names) # returns a map object

print(list(upper\_names)) # converts the map object to a list

# Output: ['ALICE', 'BOB', 'CHARLIE']

filter() function:

# Using the 'filter' function to filter out odd numbers from a list

numbers = [1, 2, 3, 4, 5]

def is\_even(n):

return n % 2 == 0 # returns True if n is even, False otherwise

even\_numbers = filter(is\_even, numbers) # returns a filter object

print(list(even\_numbers)) # converts the filter object to a list

# Output: [2, 4]

**9.What are some advantages of using lambda functions in Python? Provide examples to support your answer.**

**A.** lambda function is a small function that can be defined in a line of code without using "def" keyword. So, it is useful to create a simple function that will not reuse elsewhere.

x = lambda a: a+2

print(x(8)) # 10

some advantages are -

* They are concise and easy to read, as they avoid unnecessary names and indentation.
* They can be used to create higher-order functions, such as map, filter, and reduce, that apply a function to a sequence of elements and return a new sequence.
* They can be used to create decorators, which are functions that modify the behavior of other functions.
* it takes less time to write rather than a general function.

Using a lambda function to square each element of a list and return a new list:

numbers = [1, 2, 3, 4, 5]

squared\_numbers = list(map(lambda x: x\*\*2, numbers))

print(squared\_numbers)

# Output: [1, 4, 9, 16, 25]

**10.How does the 'any' and 'all' functions work in Python? Provide examples to demonstrate their usage.**

**A.** The "any" and "all" functions are built-in functions in python that used to check the Truth values of iterables.

syntax:

any(iterable) - return True if any element is true

False otherwise

all(iterable) - return True if all elements are true of if the iterable is empty

False otherwise

examples,

# Using the 'any' function to check if a list contains any odd numbers

numbers = [2, 4, 6, 8, 10]

odd = any(x % 2 == 1 for x in numbers) # returns False

print(odd)

# Using the 'all' function to check if a list contains only positive numbers

numbers = [1, 2, 3, 4, 5]

positive = all(x > 0 for x in numbers) # returns True

print(positive)