QUESTION BANK FOR

PYTHON PROGRAMMING TECHNIQUES

(TIU-UCA-MJ-T22201)

1. What are the Logical operators available in Python state with example?

Ans: - **Python logical operators** allow combining multiple conditions and they always return a **Boolean value** (True or False).

- **and** → True if **both conditions are True**
- **or** → True if **at least one condition is True**
- **not** → Reverses the truth value of a condition

Example:

```
x = 10
y = 5
print(x > 0 and y > 0)
print(x > 0 or y < 0)
print (not(x > 0))
```

2. State with an example, the role of the following operators:
   i) is
   ii) in
   iii)not in
   iv)<<
   v)>>

Ans:-

i) is Operator:-Checks whether two variables refer to the same object in memory, not whether their values are equal.

Example:

```
a = [1, 2, 3]
```

```
b = a
print(a is b)   # True
```

ii) in Operator:-Checks whether a value exists inside a sequence (list, tuple, string, set, etc.).

Example:

```
numbers = [1, 2, 3, 4]
print(3 in numbers)   # True
```

iii) not in Operator:-Checks whether a value does NOT exist inside a sequence.

Example:

```
numbers = [1, 2, 3, 4]
print(5 not in numbers)   # True
```

iv) << (Left Shift Operator):-Shifts the bits of a number to the left, effectively multiplying by powers of 2.

Example:

```
a = 5      # Binary: 0101
print(a << 1)   # 10 (Binary: 1010)
```

v) >> (Right Shift Operator):-Shifts the bits of a number to the right, effectively dividing by powers of 2.

Example:

```
a = 10     # Binary: 1010
print(a >> 1)   # 5 (Binary: 0101)
```

3. Describe if-elif-else structure with an example. (5)

Ans: - The `if-elif-else` structure is a conditional statement used for decision-making. It allows the program to execute different blocks of code based on whether specific conditions are true or false.

- **if** checks the first condition.
- **elif** (else if) checks additional conditions if the previous ones are false.
- **else** executes when none of the conditions are true.

Syntax

```
if condition1:
   statements
elif condition2:
   statements
else:
   statements
```

Example

```
marks = 75

if marks >= 90:
   print("Grade A")
elif marks >= 60:
   print("Grade B")
else:
   print("Grade C")
```

4. Take a age and find he or she is Child or Teenage or adult or Senior Citizen.(5)

Ans:-

Code:-

```
age = int(input("Enter age: "))

if age < 13:
```

```
    print("Child")
elif age >= 13 and age <20:
    print("Teenage")
elif age >=20 and age < 60:
    print("Adult")
else:
    print("Senior Citizen")
```

5. Take a year and find leap year or not. (5)

Ans:-

Code:-

```
year = int(input("Enter a year: "))
if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
    print("Leap Year")
else:
    print("Not a Leap Year")
```

6. The marks obtained by a student in 3 different subjects are input through the keyboard. The student gets a division as per the following rules:

Percentage above or equal to 60 First division
Percentage between 50 and 59 Second division
Percentage between 40 and 49 Third division
Percentage less than 40 --------Fail

Write a program to calculate the division obtained by the student.(5)

Ans:-

Code:-

```
a = int(input("Enter the marks of first subject: "))
b = int(input("Enter the marks of second subject: "))
c = int(input("Enter the marks of third subject: "))

per = ((a + b + c) / 300) * 100

print(f"Your percentage of marks: {per:.2f}%")

if per >= 60:
   print("First Division")
elif per >= 50 and per <= 59:
   print("Second Division")
elif per >= 40 and per <= 49:
   print("Third Division")
else:
   print("Fail")
```

7. Describe range() function with an example.(3)

Ans: - The range () function in Python generates an immutable sequence of integers. It is most commonly used in for loops to iterate a specific number of times and helps manage memory efficiently by generating numbers only when needed.

Syntax

range (start, stop, step)

- start → starting value (default is 0)
- stop → ending value (number is **not included**)
- step → difference between each number (default is 1)

Example

```
for i in range(1, 6):
   print(i)
```

8. Write a program to print all odd non prime numbers between 1 to 50.(5)

Ans:

Code:-

```
for n in range(3, 51, 2):
   for i in range(2, n):
      if n % i == 0:
         print(n)
         break
```

9. Calculates the sum of digits of a number.

Ans:-

Code:-

```
n = int(input("Enter a number: "))
s = 0
while n: s, n = s + n % 10, n // 10
print("Sum of digits:", s)
```

10. Print the following pattern: (5)

```
1
22
333
4444    for n=4
```

Ans:-

Code:-

```
n = 4
for i in range(1, n+1):
    print(str(i) * i)
```

11. Write a program to calculate the sum of the following series: (5)
    S= 1/2+2/3+3/4+....+n/(n+1) [n should be user input]
    Ans:-

Code:-

```
n = int(input("Enter the value of n: "))
s = 0
for i in range(1, n + 1):
    s += i / (i + 1)
print("Sum of the series =", s)
```

12. Difference between break and continue with example. (5)

Ans:-

break Statement:-The break statement terminates the loop immediately and transfers control to the statement following the loop.

Example:

```
for i in range(1, 6):
    if i == 3:
        break
    print(i)
```

continue Statement:- The continue statement skips the current iteration of the loop and moves control to the next iteration.

Example:

```
for i in range(1, 6):
    if i == 3:
```

```
    continue
  print(i)
```

13. Define a function which will take principal amount, time in years and rate of interest in percentage as input and calculate the simple interest. All the three arguments have default values – 1000 (principal amount), 5 (time in years) and 10 (rate of interest in percentage). Call the function from main module with all possible ways and calculate the simple interest each time. (5)

Ans:-

Code:-

```
def simple_interest(p=1000, t=5, r=10):
  si = (p * t * r) / 100
  print("Principal =", p, ", Time =", t, ", Rate =", r)
  print("Simple Interest =", si)
  print("-" * 30)

simple_interest()
simple_interest(2000)
simple_interest(2000, 3)
simple_interest(2000, 3, 8)
simple_interest(r=12, p=1500, t=4)
```

14. Define a function which will calculate factorial of a number and call the function from main module with a value taken from user.(5)

Ans:-

Code:-

```
def fact(n):
  f = 1
  for i in range(1, n + 1):
    f *= i
  return f
```

```
n = int(input("Enter a number: "))
print("Factorial =", fact(n))
```

15. Define a function which will accept name of a student and a set of sports that he/she likes and this function will display the student name and his/her favorite sports. Use variable length argument in the function parameter so that different student can have different number of favorite sports. (5)

Ans:-

Code:-

```
def sp(name, *sports):
   print("Student Name:", name)
   print("Favourite Sports:")
   for s in sports:
      print("-", s)
sp("Amit", "Cricket", "Football")
sp("Riya", "Badminton")
sp("Rahul", "Tennis", "Swimming", "Hockey")
```

16. What is function? What is the advantage and disadvantage of using function? (2+2+2)

Ans: - A **function** is a block of reusable code that performs a specific task. It runs only when it is called and helps in organizing a program into smaller, manageable parts.

Advantages of Using Functions (2 marks)

- **Code reusability:** The same function can be used multiple times, reducing repetition.
- **Modularity & readability:** Programs become easier to understand, debug, and maintain.

Disadvantages of Using Functions (2 marks)

- **Function call overhead:** Using many functions may slightly increase execution time.

- **Complexity for small programs:** For very small tasks, functions may make the program unnecessarily lengthy.

17. Using lambda function to check whether a number is Positive, negative or zero.(3)

Ans:-

Code:-

```
c = lambda n: "Positive" if n > 0 else "Negative" if n < 0 else "Zero"
num = int(input("Enter a number: "))
print(c(num))
```

18. Using lambda function to check whether the number is even or odd(3)

Ans:-

Code:-

```
c = lambda n: "Even" if n % 2 == 0 else "Odd"
num = int(input("Enter a number: "))
print(c(num))
```

19. What is the role of return and def keyword with respect to function? (2)

Ans:-

def keyword:
The def keyword is used to **define (declare) a function** and specify its name, parameters, and code block.

return keyword:
The return keyword is used to **send a value back to the caller** from a function and terminate the function execution.

20. Define recursion. Differentiate it with iteration. (2+3)

Ans: - **Recursion** is a programming technique in which a function **calls itself** to solve a problem by breaking it into smaller sub-problems. A recursive function must have a **base condition** to stop the recursion.

Difference between Recursion and Iteration

Recursion	Iteration
Function calls itself	Uses loops (`for`, `while`)
Needs a base case	Needs a loop condition
Uses more memory (stack)	Uses less memory
Code is often shorter	Code is usually faster

21. Write a python program to print Fibonacci series using recursion(5)

Ans:-

Code:-

```
def fib(n):
   if n <= 1:
      return n
   return fib(n-1) + fib(n-2)
n = int(input("Enter number of terms: "))
for i in range(n):
   print(fib(i), end=" ")
```

22. Write a python program to print Factorial using recursion(5)

Ans:-

Code:-

```
def fact(n):
   if n == 0 or n == 1:
      return 1
   return n * fact(n-1)
num = int(input("Enter a number: "))
print("Factorial:", fact(num))
```

23. Define a recursive function to find out sum of digits of a number (number should be user input). (5)

Ans:-

Code:-

```
def s(n):
```

```
    return 0 if n == 0 else n % 10 + s(n // 10)
a = int(input("Enter number: "))
print("Sum of digits = ",s(a))
```

24. Difference between class and object.(3)

Ans:-

Class: A class is a blueprint or template used to create objects. It defines properties and methods.

Object: An object is an instance of a class. It represents a real-world entity and uses the class's properties and methods.

Example:

class Student: → Class

s1 = Student() → Object

25. Define the following: a) Abstraction b) encapsulation (3+3)

Ans:-

a) Abstraction

Abstraction is the process of **hiding implementation details** and showing only the **essential features** of an object to the user. It helps reduce complexity and improves code readability.

Example: Using a function without knowing its internal logic.

b) Encapsulation

Encapsulation is the technique of **wrapping data and methods together** into a single unit (class) and restricting direct access to data using access modifiers.

Example: Private variables accessed through methods.

26. What is inheritance? Advantages of Inheritance. Types of Inheritance. (2+2+3)

Ans:-

Inheritance

Inheritance is an **OOP concept** in which a **child class acquires the properties and methods of a parent class**. It helps in code reusability and establishes a relationship between classes.

Advantages of Inheritance

- **Code reusability:** Existing code can be reused in new classes.
- **Easy maintenance:** Changes in parent class reflect in child classes.

Types of Inheritance

1. **Single Inheritance** – One child inherits from one parent
2. **Multiple Inheritance** – One child inherits from multiple parents
3. **Multilevel Inheritance** – A class derived from another derived class
4. **Hierarchical Inheritance** – Multiple children inherit from one parent

27. Frame class *Account* with object variable account holders name, account number and balance. Create multiple objects of *Account* class and perform withdraw, deposit and check balance operations on that account. Also, keep track of how many accounts you have created by using a class variable.(5)

Ans:-

Code:-

```
class Account:
   count = 0
   def __init__(self, name, acc_no, balance):
      self.name = name
      self.acc_no = acc_no
      self.balance = balance
      Account.count += 1
   def deposit(self, amt):
```

```
        self.balance += amt
    def withdraw(self, amt):
        if amt <= self.balance:
            self.balance -= amt
        else:
            print("Insufficient balance")
    def check_balance(self):
        print("Balance:", self.balance)
a1 = Account("Rahul", 101, 5000)
a2 = Account("Anita", 102, 3000)
a1.deposit(1000)
a1.withdraw(2000)
a1.check_balance()
a2.check_balance()
print("Total accounts created:", Account.count)
```

28. Frame a *Student* class with object variable student name, roll and marks. Create two student objects and display their details.(5)

Ans:-

Code:-

```
class Student:

    def __init__(self, name, roll, marks):

        self.name = name

        self.roll = roll

        self.marks = marks

    def display(self):

        print("Name:", self.name)

        print("Roll No:", self.roll)

        print("Marks:", self.marks)

s1 = Student("Amit", 101, 85)
```

```
s2 = Student("Riya", 102, 92)

s1.display()

s2.display()
```

29. Frame a base class *Person* which contains object variable name and age of a person. It should have a display_person() method which will display details of a person. Frame another two classes *Student* and *Teacher* (both should inherit the base class *Person*) which will contain roll and marks (for *Student*) and subject and experience (for *Teacher*).*Student* class should contain display_student() method which will display student record and *Teacher* class should contain display_teacher() method which will display teacher record. Write a program in python to test the above classes.(10)

Ans:-

Code:-

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def display_person(self):
        print("Name:", self.name)
        print("Age:", self.age)
class Student(Person):
    def __init__(self, name, age, roll, marks):
        super().__init__(name, age)
        self.roll = roll
        self.marks = marks
    def display_student(self):
        self.display_person()
        print("Roll No:", self.roll)
        print("Marks:", self.marks)
class Teacher(Person):
    def __init__(self, name, age, subject, experience):
        super().__init__(name, age)
        self.subject = subject
        self.experience = experience
```

```
    def display_teacher(self):
        self.display_person()
        print("Subject:", self.subject)
        print("Experience (years):", self.experience)
s1 = Student("Amit", 20, 101, 88)
s2 = Student("Riya", 19, 102, 92)
t1 = Teacher("Mr. Sharma", 45, "Math", 20)
print("Name of Students")
s1.display_student()
s2.display_student()
print("Name of Subject Teacher")
t1.display_teacher()
```

30. Explain Method overriding with an example.(5)

Ans:-

Method overriding occurs when a child class provides its own implementation of a method that is already defined in the parent class, using the same method name and parameters. It allows runtime polymorphism.

Example in Python

```
class Parent:
    def show(self):
        print("This is Parent class method")
class Child(Parent):
    def show(self):
        print("This is Child class method")
obj = Child()
obj.show()
```

31. Which special method is automatically called to create objects in Python? What does self keyword represent?(2+2)

Ans: - The __init__() method is the special method that is automatically called when an object is created in Python.

Meaning of self Keyword

The self keyword represents the current object of the class. It is used to access object variables and methods inside the class.

32. Reverse a string using Slicing.(3)

Ans:-

Code:-

```
s = input("Enter a string: ")
print("Reversed string:", s[::-1])
```

33. Take a string and check Palindrome or not.(5)

Ans:-

Code:-

```
s = input("Enter a string: ")
if s == s[::-1]:
   print("Palindrome")
else:
   print("Not Palindrome")
```

34. Explain the following string methods with example: a) isalnum() b) upper() c) lower() d) startswith() e) endswith() Each carries 2 mark

Ans:-

a) isalnum():-Checks whether the string contains **only alphabets and numbers** (no spaces or symbols).

Example:

```
s = "Python123"
print(s.isalnum())
```

b) upper():-Converts all characters in the string to **uppercase**.

Example:

```
s = "python"
print(s.upper())
```

c) lower():-Converts all characters in the string to **lowercase**.

Example:

```
s = "PYTHON"
print(s.lower())
```

d) startswith():-Checks whether a string **starts with a specified substring**.

Example:

```
s = "python programming"
print(s.startswith("python"))
```

e) endswith():-Checks whether a string **ends with a specified substring**.

Example:

```
s = "file.txt"
print(s.endswith(".txt"))
```

35. Differentiate between list aliasing and list cloning with example.(5)

Ans:-

List aliasing

List aliasing occurs when two or more variables refer to the same list object in memory. Any change made using one variable is automatically reflected in the other.

Example:

```
a = [1, 2, 3]
b = a
b.append(4)
print(a)   # [1, 2, 3, 4]
```

List cloning occurs when a new variable **refers to a separate copy of the list**. Changes made in one list do not affect the other list.

Example:

```
a = [1, 2, 3]
b = a.copy()
b.append(4)
print(a)
```

36. Write a program that uses list comprehension to create a list which contains sum of squares of one even number and one odd number where the even number will come from the numbers 1to 5 and the odd number will come from the numbers 6 to 10.(5)

Ans:-

Code:-

```
r = [(e**2 + o**2)
      for e in range(1, 6) if e % 2 == 0
      for o in range(6, 11) if o % 2 != 0]
print(r)
```

37. Write a program that uses list comprehension to create a list which contains cubes of all even numbers from the numbers 1 to 20.(3)

Ans:-

Code:-

```
c = [x**3 for x in range(1, 21) if x % 2 == 0]
print(c)
```

38. Implement a STACK using a list.(5)

Ans:-

Code:-

```
stack = []
stack.append(10)
stack.append(20)
stack.append(30)
print("Stack:", stack)
print("Popped:", stack.pop())
```

```
print("Stack after pop:", stack)
```

39. Implement a QUEUE using a list. (5)

Ans:-

Code:-

```
queue = []
queue.append(10)   # enqueue
queue.append(20)
queue.append(30)
print("Queue:", queue)
print("Dequeued:", queue.pop(0))  # dequeue
print("Queue after dequeue:", queue)
```

40. Write a program that uses filter() function to filter out only even numbers from a list.(3)

Ans:-

Code:-

```
n = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
e = list(filter(lambda x: x % 2 == 0,n))
print(e)
```

41. Write a program that uses map() function to print the square value of each element in a list. (3)

Ans:-

Code:-

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

42. Write a program that uses filter() function to create a list of numbers from 1 to 50 that are either divisible by 3 or divisible by 6. (5)

Ans:-

Code:-

```
n=range(1, 51)
r = list(filter(lambda x: x % 3 == 0 or x % 6 == 0, n))
print(r)
```

43. Write a program to find multiplication of all values in a list using reduce() function.(3)

Ans:-

Code:-

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

44. Explain the following list methods with example: (2)
    a) append()
    b)insert()
    c) remove() d) count() e) extend() f)index()

Ans:-

a) append()

The append() method is used to add an element at the end of a list. It increases the size of the list by one.

Examples

```
l = [1, 2, 3]
l.append(4)
print(l)
```

b) insert()

The insert() method is used to insert an element at a specific index position in the list.

Example

```
l = [1, 2, 4]
l.insert(2, 3)
print(l)
```

c) remove()
The remove() method deletes the first occurrence of a specified element from the list.

Example

```
l = [1, 2, 3, 2]
l.remove(2)
print(l)
```

d) count()
The count() method returns the number of times a particular element appears in the list.

Example

```
l = [1, 2, 2, 3]
print(l.count(2))
```

e) extend()
The extend() method is used to add multiple elements from another list to the end of the current list.

Example

```
l = [1, 2]
l.extend([3, 4, 5])
print(l)
```

f) index()
The index() method returns the index position of the first occurrence of a given element in the list.

Example

```
l = [10, 20, 30]
print(l.index(20))
```

45. Difference between del(), pop() and remove(). (3)

Ans:-

del():-The del statement is used to delete an element using its index or to delete the entire list. It does not return any value.

Example

```
l = [1, 2, 3]
del l[1]
print(l)
```

pop():-The pop() method removes an element from the list using its index and returns the removed element. If no index is given, it removes the last element.

Example

```
l = [1, 2, 3]
print(l.pop())
print(l)
```

remove():-The remove() method deletes the first occurrence of a specified element from the list and does not return any value.

Example

```
l = [1, 2, 3]
l.remove(2)
print(l)
```

46. Difference between sort() and sorted().(3)

Ans:-

sort():-It is a list method that sorts the elements of the list in place, meaning it modifies the original list and does not return a new list.

```
l = [3, 1, 2]
l.sort()
print(l)
```

sorted():-It is a built-in function that returns a new sorted list without changing the original list.

```
l = [3, 1, 2]
new_list = sorted(l)
print(new_list)
```

47. Difference between list and tuple. (3)

Ans:-

List

A list is a collection of elements that are ordered, mutable, and can store different data types. Elements of a list can be modified after creation.

Tuple

A tuple is a collection of elements that are ordered and immutable. Once a tuple is created, its elements cannot be changed.

48. Write a program that defines a list of countries that are members of SAARC. Take one country name from user and check whether that country is a member of SAARC or not. .(3)

Ans:-

Code:-

```
saarc = ["India", "Pakistan", "Bangladesh", "Sri Lanka","Nepal", "Bhutan", "Maldives", "Afghanistan"]
country = input("Enter country name: ")
if country in saarc:
   print("Member of SAARC")
else:
   print("Not a member of SAARC")
```

49. Write a program to remove all duplicate objects from a list. .(5)

Ans:-

Code:-

```
l = [1, 2, 3, 2, 1, 4]
u = []
for i in l:
   if i not in u:
```

```
        u.append(i)
print(u)
```

50. Write a program that creates a list of ten random integers. Then create two lists- Odd list and even list that has all odd and even values in the list respectively. (5)

Ans:-

Code:-

```
import random
l = [random.randint(1, 100) for _ in range(10)]
even = [x for x in l if x % 2 == 0]
odd = [x for x in l if x % 2 != 0]
print("List:", l)
print("Even:", even)
print("Odd:", odd)
```

51. State the enumerate () function with an example. (3)

Ans:-

enumerate () Function

The enumerate() function is used to add an index (counter) to each element of an iterable and returns (index, value) pairs.

Example:

```
a = ["apple", "banana", "cherry"]
for i, f in enumerate(a):
   print(i, f)
```

52. State with an example how zip() function works. (3)

Ans:-

zip() Function

The zip() function combines two or more iterables element-wise into tuples, returning a zip object.

Example

```
names = ["Amit", "Riya", "Sara"]
marks = [85, 92, 78]

result = list(zip(names, marks))
print(result)
```

53. Write a program that has two sequences. First which stores some questions and second stores the corresponding answers.Use the zip() function to form a valid question answer series. (5)

Ans:-

Code:-

```
questions = ["Capital of India?", "5 + 7?", "Python is a?"]
answers = ["New Delhi", 12, "Programming Language"]
s = list(zip(questions, answers))
for q, a in s:
    print(f"Q: {q}  A: {a}")
```

54. Describe the following functions with an example a)filter() b)map() c)reduce() [3 mark each]

Ans:-

a) filter()

The filter() function is used to **select elements from a sequence** for which a given function returns True.

Example:

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)
```

b) map()

The map() function is used to **apply a function to each element** of a sequence and return a new sequence with the results.

Example:

```
numbers = [1, 2, 3, 4]
squares = list(map(lambda x: x**2, numbers))
print(squares)
```

c) reduce()

The reduce() function (from functools) **applies a function cumulatively to the items of a sequence** to reduce it to a single value.

Example:

```
from functools import reduce
numbers = [1, 2, 3, 4]
product = reduce(lambda x, y: x * y, numbers)
print(product)
```

55. Differentiate between append() and insert() with respect to list. (3)
Ans:-

append()

The append() method is used to add an element at the end of a list. It increases the size of the list by one.

Examples

```
l = [1, 2, 3]
l.append(4)
print(l)
```

insert()
The insert() method is used to insert an element at a specific index position in the list.

Example

```
l = [1, 2, 4]
l.insert(2, 3)
print(l)
```

56. Differentiate between list and set. (3)
Ans:-

List: A list is an ordered collection of elements that can contain duplicates. Elements can be accessed by their index.
Example:

```
lst = [1, 2, 2, 3]
print(lst[1])
```

Set: A set is an unordered collection of unique elements. It does not allow duplicates and elements cannot be accessed by index.
Example:

```
s = {1, 2, 2, 3}
print(s)
```

57.Describe the following set operations using functions and operators
a)union
b)intersection
c)difference
d)issubset
e)issuperset
[each 2 mark]
Ans:-

a) Union:-Returns a set containing **all elements from both sets**.

- **Function:** set1.union(set2)
- **Operator:** set1 | set2

Example:

```
a = {1, 2, 3}
b = {3, 4, 5}
print(a.union(b))  # {1, 2, 3, 4, 5}
print(a | b)      # {1, 2, 3, 4, 5}
```

b) Intersection:-Returns a set containing **only elements present in both sets**.

- **Function:** set1.intersection(set2)
- **Operator:** set1 & set2

Example:

```
a = {1, 2, 3}
b = {3, 4, 5}
print(a.intersection(b))  # {3}
print(a & b)              # {3}
```

c) Difference:-Returns a set containing **elements in the first set but not in the second**.

- **Function:** set1.difference(set2)
- **Operator:** set1 - set2

Example:

```
a = {1, 2, 3}
b = {2, 3, 4}
print(a.difference(b))  # {1}
print(a - b)          # {1}
```

d) issubset:-Checks whether **all elements of a set are in another set**. Returns True or False.

- **Function:** set1.issubset(set2)
- **Operator:** <=

Example:

```
a = {1, 2}
b = {1, 2, 3}
print(a.issubset(b))  # True
print(a <= b)         # True
```

e) issuperset:-Checks whether **a set contains all elements of another set**. Returns True or False.

- **Function:** set1.issuperset(set2)
- **Operator:** >=

Example:

```
a = {1, 2, 3}
b = {1, 2}
print(a.issuperset(b))
print(a >= b)
```

58. Define dictionary in python. Differentiate it with a list. (2+3)

Ans:-A dictionary in Python is an unordered collection of data stored in key–value pairs. Each key is unique and is used to access its corresponding value.

Example:

```
d = {"name": "Amit", "age": 20}
```

Difference between Dictionary and List (3 marks)

- **Storage:**
  A dictionary stores data as key–value pairs, whereas a list stores data as a sequence of elements.
- **Access:**
  Dictionary elements are accessed using keys, while list elements are accessed using index positions.
- **Uniqueness:**
  Dictionary keys must be unique, but a list can contain duplicate elements.

59.Describe the following operations on dictionary
a)add an item
b)modify an item
c)delete an item
d)return a list of keys
e)return a list of value [2 mark each]

Ans:-

a) Add an item

A new item can be added to a dictionary by assigning a value to a new key.

```
d = {"name": "Amit", "age": 20}
d["city"] = "Kolkata"
print(d)
```

b) Modify an item

An existing item can be modified by assigning a new value to its key.

```
d = {"name": "Amit", "age": 20}
d["age"] = 21
print(d)
```

c) Delete an item

An item can be removed from a dictionary using the del statement.

```
d = {"name": "Amit", "age": 20}
del d["age"]
print(d)
```

d) Return a list of keys

The keys() method returns all keys of the dictionary.

```
d = {"name": "Amit", "age": 20}
print(list(d.keys()))
```

e) Return a list of values

The values() method returns all values of the dictionary.

```
d = {"name": "Amit", "age": 20}
print(list(d.values()))
```

60)WAP that creates a dictionary of cubes of odd numbers in the range 1-10. (3)
Ans:-
Code:-

```
d = {x: x**3 for x in range(1, 11) if x % 2 != 0}
print(d)
```

61) Create a user defined module which contains a function that will calculate maximum of two numbers.WAP in main module that will take two integer values from user and find out the larger of the two values using the above function. (5)
Ans:-

File 1: maxmodule.py *(User-defined module)*

```
def max(a, b):
   if a > b:
      return a
   else:
      return b
```

File 2: main.py *(Main module)*

```
import maxmodule
x = int(input("Enter first number: "))
y = int(input("Enter second number: "))
r = maxmodule.max(x, y)
print("Larger number is:", r)
```

62)State the difference between local and global variables with a snippet of code. (5)

Ans:-

Local Variable:
A local variable is declared inside a function. It can be accessed only within that function where it is defined. Local variables are created when the function is called and destroyed after the function execution ends.

Global Variable:
A global variable is declared outside all functions. It can be accessed anywhere in the program, both inside and outside functions. Global variables remain in memory for the entire duration of the program.

Code Snippet:

```
x = 10    # Global variable

def test():

   y = 5 # Local variable

   print(x)

   print(y)

test()

print(x)
```

63)Create your own module and use that module in another module to solve a particular problem. (5)
Ans:-
Code:-

User-defined Module (mymath.py)

```
def square(n):
```

```
    return n * n
```

Main Module (main.py)

```
import mymath

num = int(input("Enter a number: "))
print("Square of the number is:", mymath.square(num))
```

64)Use the numpy module to perform the following operations
a) inverse of a matrix
b) rank of a matrix
c)transpose of a matrix
d)multiplication of two matrices
e)find out determinant of a matrix

[Each 2 mark]

Ans:-

Using NumPy for Matrix Operations

```
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
```

a) Inverse of a matrix

The inverse of a matrix is calculated using numpy.linalg.inv().

```
inv_A = np.linalg.inv(A)
print(inv_A)
```

b) Rank of a matrix

The rank of a matrix is found using numpy.linalg.matrix_rank().

```
rank_A = np.linalg.matrix_rank(A)
print(rank_A)
```

c) Transpose of a matrix

The transpose of a matrix is obtained using .T.

```
trans_A = A.T
print(trans_A)
```

d) Multiplication of two matrices

Matrix multiplication is done using dot().

```
mul = np.dot(A, B)
print(mul)
```

e) Determinant of a matrix

The determinant of a matrix is calculated using numpy.linalg.det().

```
det_A = np.linalg.det(A)
print(det_A)
```

65)WAP using numpy module to create a 3×3 identity matrix. (2)

Ans:-

Code:-

```
import numpy as np

I = np.identity(3)

print(I)
```

66)WAP using numpy module to create a 3×2 matrix whose every element is filled with the number 2. (2)

Ans:-

Code:-

```
import numpy as np

A = np.full((3, 2), 2)
```

```
print(A)
```

67) WAP using numpy module to create a 3×3 matrix whose every element is filled with random integers numbers between 5-9. (2)

Ans:-

Code:-

```
import numpy as np
A = np.random.randint(5, 10, (3, 3))
print(A)
```

68) WAP in Python using matplotlib to create a pie chart showing how hours in a day are spent on the basis of the activities like ‘Sleeping', 'Eating', 'Working' and 'Playing'. (5)

Ans:-

Code:-

```
import matplotlib.pyplot as p
a = ['Sleeping', 'Eating', 'Working', 'Playing']
h = [8, 2, 8, 6]
p.pie(h, labels=a, autopct='%1.1f%%', startangle=90)
p.title("Hours Spent on Daily Activities")
p.show()
```

69) WAP in Python to create a scatter plot using matplotlib that shows the number of hours students spend practicing programming over a week and their corresponding scores in a weekly coding test. (5)

Ans:-

Code:-

```
import matplotlib.pyplot as p
h= [2, 3, 4, 5, 6, 7, 8]
s = [50, 55, 60, 65, 70, 75, 80]
p.scatter(h, s, color='blue')
p.title("Hours Practiced vs Test Scores")
p.xlabel("Hours Practiced")
p.ylabel("Test Scores")
p.show()
```

70) WAP in Python to create a bar graph using matplotlib that represents the marks obtained by a student in five subjects: English, Math, Science, History, and Computer Science. (5)

Ans:-

Code:-

```
import matplotlib.pyplot as p
s = ['English', 'Math', 'Science', 'History', 'CS']
m = [85, 90, 78, 88, 95]
p.bar(s, m, color='green')
p.title("Marks Obtained in Subjects")
p.xlabel("Subjects")
p.ylabel("Marks")
p.show()
```

**********END**********