June 2022

1)What is the output of the following print statement in python?

(a)print(9//2) (b)print(9/2)

Ans:

(a) print(9//2)

The // operator in Python performs integer (floor) division. It divides the number and truncates the decimal part, returning an integer result.

9 // 2 = 4

Output: 4

(b) print(9/2)

The / operator in Python performs floating-point division. It divides the numbers and returns the result as a float.

9 / 2 = 4.5

Output: 4.5

2)Write a python program to count numbers of even numbera to odd numbers in a given set of n numbers

n = int(input("Enter the number of elements"))

even\_count = 0

odd\_count = 0

print("Enter the numbers:")

for \_ in range(n):

num = int(input())

if num % 2 == 0:

even\_count += 1

else:

odd\_count += 1

print("Number of even numbers:", even\_count)

print("Number of odd numbers:", odd\_count)

3)Write a python program to generate the following type of pattern for the given N rows

1

1 2

1 2 3

1 2 3 4

Ans

N = int(input("Enter the number of rows: "))

for i in range(1, N + 1):

for j in range(1, i + 1):

print(j, end=" ")

print()

4)Mention the different types of loops and control statements in Python and explain each type with suitable example

1. Loops in Python

a. For Loop

Purpose:iterates over a sequence or range of numbers

Syntax:

for variable in sequence:

# code block

Example:

for i in range(5):

print(i)

Output:

0

1

2

3

4

b. While Loop

Purpose: Repeats as long as a condition is True.

Syntax:

while condition:

# code block

Example:

count = 0

while count < 5:

print(count)

count += 1

Output:

0

1

2

3

4

2. Control Statements in Python

a. Break

Purpose: Terminates the loop prematurely.

Example:

for i in range(5):

if i == 3:

break

print(i)

Output:

0

1

2

b. Continue

Purpose: Skips the current iteration and moves to the next.

Example:

for i in range(5):

if i == 3:

continue

print(i)

Output:

0

1

2

4

c. Pass

Purpose: Acts as a placeholder, does nothing when executed.

Example:

for i in range(5):

if i == 3:

pass # Placeholder for future code

print(i)

Output:

0

1

2

3

4

d. Else with Loops

Purpose: Executes a block of code if the loop completes without being interrupted by break.

Example:

for i in range(5):

if i == 6:

break

else:

print("Loop completed without break")

5) Write the Python program to print all the prime numbers less than 1000.

Ans:

for num in range(2, 1000):

for i in range(2, num):

if num % i == 0:

break

else:

print(num, end=" ")

6)Write a Python program to find the distance between two points (x1, y1 )and (x2, y2.)

Ans:

X1= float(input("Enter x1: "))

y1 = float(input("Enter y1: "))

x2 = float(input("Enter x2: "))

y2 = float(input("Enter y2: "))

# Distance formula

distance = ((x2 - x1) \*\* 2 + (y2 - y1) \*\* 2) \*\* 0.5

print("The distance between the two points is:", distance)

Jan 2024

Q. Explain type conversion with example.

Ans: Implicit Type Conversion:

This occurs automatically when the data types are compatible, and there is no loss of information.

eg: Python converts num\_int (an integer) to a float automatically before performing the addition

num\_int = 10

num\_float = num\_int + 2.5

print(num\_float)

print(type(num\_float))

Output: 12.5

Output: <class 'float'>

Explicit Type Conversion:

This requires the programmer to use predefined functions like int(), float(), str(), etc., to convert data types.

eg: the programmer explicitly specifies the desired data type using conversion functions.

num\_float = 5.7

num\_int = int(num\_float)

print(num\_int)

print(type(num\_int))

Output: 5

Output: <class 'int'>

Q. Write a program that accepts the lengths of three sides of a triangle as inputs and

outputs whether or not the triangle is a right triangle.

Ans:def is\_right\_triangle(a, b, c):

sides = sorted([a, b, c])

return sides[0]\*2 + sides[1]2 == sides[2]\*2

try:

side1 = float(input("Enter the length of the first side: "))

side2 = float(input("Enter the length of the second side: "))

side3 = float(input("Enter the length of the third side: "))

if is\_right\_triangle(side1, side2, side3):

print("The triangle is a right triangle.")

else:

print("The triangle is NOT a right triangle.")

except ValueError:

print("Please enter valid numeric values for the sides.")

Output:

Enter the length of the first side: 3

Enter the length of the second side: 4

Enter the length of the third side: 5

The triangle is a right triangle.

Q. Write a Python program to print all numbers between 100 and 1000 whose sum

of digits is divisible by 9.

Ans: def sum\_of\_digits(num):

return sum(int(digit) for digit in str(num))

print("Numbers between 100 and 1000 whose sum of digits is divisible by 9:")

for number in range(100, 1000):

if sum\_of\_digits(number) % 9 == 0:

print(number, end=" ")

Output:

Numbers between 100 and 1000 whose sum of digits is divisible by 9:

108 117 126 135 144 153 162 171 180 189 198 207 216 225 234 243 252 261 270 279 288 297 306 315 324 333 342 351 360 369 378 387 396 405 414 423 432 441 450 459 468 477 486 495 504 513 522 531 540 549 558 567 576 585 594 603 612 621 630 639 648 657 666 675 684 693 702 711 720 729 738 747 756 765 774 783 792 801 810 819 828 837 846 855 864 873 882 891 900 909 918 927 936 945 954 963 972 981 990 999

Q. Illustrate the use of range() in Python.

Ans: The range() function in Python generates a sequence of numbers. It is commonly used in loops to iterate over a sequence of values.

Syntax:

range(stop)

range(start, stop)

range(start, stop, step)

eg: Generate numbers from 1 to 10 with a step of 2

for i in range(1, 11, 2):

print(i, end=" ")

Output: 1 3 5 7 9

Q. Write a Python program to print all prime factors of a given number.

Ans: def prime\_factors(n):

print(f"Prime factors of {n} are:", end=" ")

while n % 2 == 0:

print(2, end=" ")

n //= 2

factor = 3

while factor \* factor <= n:

while n % factor == 0:

print(factor, end=" ")

n //= factor

factor += 2

if n > 2:

print(n, end=" ")

print()

try:

number = int(input("Enter a positive integer: "))

if number > 0:

prime\_factors(number)

else:

print("Please enter a positive integer.")

except ValueError:

print("Invalid input. Please enter a valid positive integer.")

Output: Enter a positive integer: 56

Prime factors of 56 are: 2 2 2 7

JUNE 2023

# 1.Python program to calculate the sum of the cosine series

import math

def cosine\_series\_sum(x, n):

cos\_sum = 1

term = 1

for i in range(1, n):

term = -x\*2 / ((2 \* i - 1) \* (2 \* i))

cos\_sum += term

return cos\_sum

angle\_degrees = float(input("Enter the angle in degrees: "))

n\_terms = int(input("Enter the number of terms: "))

angle\_radians = math.radians(angle\_degrees)

result = cosine\_series\_sum(angle\_radians, n\_terms)

print(f"The sum of the cosine series up to {n\_terms} terms is: {result}")

#2. write a python program to find X^Y or pow(X,Y) without using standard function

def power(x, y):

Result = 1

Is\_negative\_exponent = FFals

If y < 0:

Is\_negative\_exponent = True

Y = -y

For \_ in range(y):

Result \*= x

If is\_negative\_exponent:

Result = 1 / result

Return result

Base = float(input(“Enter the base (X): “))

Exponent = int(input(“Enter the exponent (Y): “))

Result = power(base, exponent)

Print(f”{base}^{exponent} = {result}”)

# 3.Write a python program to generate the following type of pattern for the given N

rows where N <= 26.

A

A B

A B C D

A B C D E

Def generate\_pattern(n):

If n > 26:

Print(“N must be less than or equal to 26.”)

Return

For I in range(1, n + 1):

Row = “ “.join(chr(65 + j) for j in range(i))

Print(row)

N = int(input(“Enter the number of rows (N <= 26): “))

Generate\_pattern(n)

# 4.Write a python program to generate prime numbers within a certain range

Def is\_prime(num):

If num <= 1:

Return False

For I in range(2, int(num\*\*0.5) + 1):

If num % I == 0:

Return False

Return True

Def generate\_primes(start, end):

Primes = []

For num in range(start, end + 1):

If is\_prime(num):

Primes.append(num)

Return pprime

Start = int(input(“Enter the starting value of the range: “))

End = int(input(“Enter the ending value of the range: “))

Prime\_numbers = generate\_primes(start, end)

Print(f”Prime numbers between {start} and {end}: {prime\_numbers}”)

What is the output of the following Python code. Justify your answer.

x = 'abcd'

for i in range(len(x)):

print(i)

Output:

0

1

2

3

2. Write the syntax and semantics of the multiway-if statement.

if condition1:

# Block executed if condition1 is True

elif condition2:

# Block executed if condition1 is False and condition2 is True

elif condition3:

# Block executed if all previous conditions are False and condition3 is True

else:

# Block executed if all conditions are False

3.Write a Python program to find the roots of a quadratic equation, ax

2

+ bx + c =

0 . Consider the cases of both real and imaginary roots.

a = float(input("Enter coefficient a: "))

b = float(input("Enter coefficient b: "))

c = float(input("Enter coefficient c: "))

if a == 0:

print("This is not a quadratic equation.")

else:

discriminant = b\*\*2 - 4\*a\*c

if discriminant > 0:

root1 = (-b + discriminant\*\*0.5) / (2 \* a)

root2 = (-b - discriminant\*\*0.5) / (2 \* a)

print("Roots are real and distinct:")

print("Root 1 =", root1)

print("Root 2 =", root2)

elif discriminant == 0:

root = -b / (2 \* a)

print("Roots are real and equal:")

print("Root =", root)

else:

real\_part = -b / (2 \* a)

imaginary\_part = (-discriminant)\*\*0.5 / (2 \* a)

print("Roots are imaginary:")

print(f"Root 1 = {real\_part} + {imaginary\_part}i")

print(f"Root 2 = {real\_part} - {imaginary\_part}i")

4. Write a Python program to check whether a number is Armstrong number or not.

An Armstrong number is an n-digit number that is equal to the sum of the n

th

powers of its digits.

Examples: 153 = 1^3

+ 5^3

+ 3^3

, 1634= 1

^4

+ 6^4

+ 3^4

+ 4^4

num = int(input("Enter a number: "))

num\_str = str(num)

num\_digits = len(num\_str)

sum\_of\_powers = 0

for digit in num\_str:

sum\_of\_powers += int(digit) \*\* num\_digits

if sum\_of\_powers == num:

print(f"{num} is an Armstrong number.")

else:

print(f"{num} is not an Armstrong number.")

5.Write a Python program to display the sum of odd numbers between a

programmer specified upper and lower limit.

lower\_limit = int(input("Enter the lower limit: "))

upper\_limit = int(input("Enter the upper limit: "))

odd\_sum = 0

for num in range(lower\_limit, upper\_limit + 1):

if num % 2 != 0:

odd\_sum += num

print(f"The sum of odd numbers between {lower\_limit} and {upper\_limit} is: {odd\_sum}")

6.

e ^x=1+x+ x^2/2!+ x^3/3! +x^4/4!+…

Given the value of x, write a Python program to evaluate the following series upto

n terms:

x = float(input("Enter the value of x: "))

n = int(input("Enter the number of terms (n): "))

result = 1

term = 1

for i in range(1, n):

term = term \* (x / i)

result += term

print(f"The value of e^{x} up to {n} terms is: {result}")

1.Illustrate the use of negative indexing of list with example.

Negative indexing in Python allows you to access elements in a list from the end, starting with -1 for the last element, -2 for the second-to-last, and so on.

example:

fruits = ['apple', 'banana', 'cherry', 'date', 'elderberry']

print(fruits[-1]) Output: elderberry (last element)

print(fruits[-2]) Output: date (second-to-last element)

print(fruits[-3]) Output: cherry (third-to-last element)

3a) Python program to count how many times each character appears in a given string and store the count in a dictionary with the character as the key:

def count\_characters(input\_string):

char\_count = {}

for char in input\_string:

if char in char\_count:

char\_count[char] += 1

else:

char\_count[char] = 1

return char\_count

input\_string = input("Enter a string: ")

result = count\_characters(input\_string)

print("Character count:", result)

3b) Create a function min\_max() that takes a list of numbers as an argument and returns the smallest and largest numbers:

def min\_max(numbers):

if not numbers:

return None, None

smallest = min(numbers)

largest = max(numbers)

return smallest, largest

numbers = [int(x) for x in input("Enter numbers separated by spaces: ").split()]

smallest, largest = min\_max(numbers)

print(f"The smallest number is {smallest}, and the largest number is {largest}.")

14a) Python program to read n integers into a list and separate the positive and negative numbers into two different lists:

def separate\_numbers(numbers):

positive\_numbers = []

negative\_numbers = []

for num in numbers:

if num >= 0:

positive\_numbers.append(num)

else:

negative\_numbers.append(num)

return positive\_numbers, negative\_numbers

numbers = [int(x) for x in input("Enter numbers separated by spaces: ").split()]

positive\_numbers, negative\_numbers = separate\_numbers(numbers)

print("Positive numbers:", positive\_numbers)

print("Negative numbers:", negative\_numbers)

14b) Create a dictionary of names and birthdays. Write a Python program that asks the user to enter a name, and the program displays the birthday of that person:

def get\_birthday(name, birthday\_dict):

if name in birthday\_dict:

return birthday\_dict[name]

else:

return "Birthday not found."

birthday\_dict = {

'Alice': '1990-04-25',

'Bob': '1985-11-12',

'Charlie': '1992-06-30'

}

name = input("Enter a name to find the birthday: ")

birthday = get\_birthday(name, birthday\_dict)

print(f"The birthday of {name} is: {birthday}")