Name - Shubham Sarkar

Roll number - 2107017

**1.1**



Examples of Database Management Projects in Python include :-

1. General purpose database systems like Informix, MySQL,Oracle etc.
2. Embedding Applications based database systems like SQLite,ThinkSQL,asql and GadFly.
3. **Raw Data in Database Systems like** Read or WriteFile,Spreadsheets (CSV) and Spreadsheets (xlsx, xls).

Examples of ML projects in Python include:-

1. Spam Email Classification.
2. Sentiment Analysis using text.
3. Image classification

IV)Fraud detection in transactions

Examples of AI projects using python are:-

1. Chatbot
2. Virtual Assistant
3. Speech recognition systems
4. Autonomous vehicles

Examples of Data Science projects using python are:-

1. Exploratory data analysis
2. Predictive analysis
3. Customer segmentation
4. Fraud detection
5. Python libraries used in different IT domains are:-

I)Pandas library is used in various IT domains like Data Analysis,Data Science and Machine Learning,Business Analytics,Data visualization etc.

II) numpy library is used in various IT domains like Data Science and Machine Learning, Scientific computing, Image and signal processing,Computational mathematics etc.

III)matplotlib library is used in domains like Data visualization,Data analysis and exploration,Machine learning, finance and economics etc.

IV)tkinter library is used in domains like GUI(Graphical User Interfacing), Desktop application development,Education and e-learning,Gaming and entertainment etc.

V)sklearn library is used in domains like Data Science, Machine Learning,Data mining,natural language processing etc.

VI)tensorflow library is used in domains like Machine Learning,Deep Learning,Artificial Intelligence,Natural language processing etc.

**1.2**

1. The given code is used to make a height vs frequency plot of various observations imported from the libraries provided. Here the x axis represents height and y axis represents frequency of height of different people. Here plt.figure(dpi = 200 ) means that in the figure the dots per inch is 200, the plt.title(“Height distribution”) it defines the title of the graph, plt.xlabel and plt.ylabel are used for plotting the graph wrt x and y axis respectively and plt.show is used to show the graph as output.

4)A compiler is a special program that translates a programming language's source code into machine code, bytecode or another programming language. The source code is typically written in a high-level, human-readable language such as Java or C++. Wheras in computer science, an interpreter is a computer program that directly executes instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program.

5)A data structure is a specialized format for organizing, processing, retrieving and storing data. There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. We need it in programming because data structures make it easy for users to access and work with the data they need in appropriate ways.

**1.3**

1. A data structure is a specialized format for organizing, processing, retrieving and storing data. There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. Examples of data structures include array,linked list,stacks,queue etc.

The real life applications of data structures include :-

1. Arrays are used to store data in tabular form
2. Images are linked with each other. So, an image viewer software uses a linked list to view the previous and the next images using the previous and next buttons.
3. Most internet processes and requests use queue.

3)Friends = [‘Ram’, ‘Shyam’, ‘Aryan’, ‘Priya’, ‘Madhuri’]

There are several inbuilt methods of list data type in python and the use of them is given in the table below which was found using help() function in Python.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

4)An ordered collection means that the elements of the collection have a specific order. The order is independent of the value. A List is an example. A sorted/unordered collection means that not only does the collection have order, but the order depends on the value of the element.

1. The types of each value in python are given below :-

1. 100 is of integer data type
2. 105.5 is of float data type
3. 192.56 j is of imaginary data type
4. 10 + 6j is of imaginary data type
5. ‘10’ is of string data type
6. ‘Hello world’5 is of invalid data type
7. [10,20,50,100] is of list data type
8. {‘name’: ‘sachin’, ‘age’: 24, ‘language’: ‘python’} is of dictionary data type

**1.4**

1. Some examples of lists that store useful information in python are :-

Language list

languages = ['Python', 'Java', 'JavaScript', 'C++', 'Ruby']

City list

cities = ['New York', 'London', 'Tokyo', 'Paris', 'Sydney']

Products list

products = ['Phone', 'Laptop', 'Headphones', 'Camera', 'Smartwatch']

Food list

foods = ['Pizza', 'Burger', 'Sushi', 'Pasta', 'Tacos']

Cars list

cars = ['Toyota', 'Honda', 'BMW', 'Ford', 'Tesla']

5)In Python, identifiers are names used to identify variables, functions, classes, modules, or other objects. They are user-defined names that follow certain rules and conventions. Here are the rules for naming identifiers in Python:

Valid Characters: An identifier can consist of letters (both lowercase and uppercase), digits, and underscores (\_). It must start with a letter or an underscore.

Case-Sensitive: Python is case-sensitive, so myVariable and myvariable are considered different identifiers.

Reserved Keywords: You cannot use reserved keywords as identifiers because they have special meanings in Python. For example, you cannot use if, for, while, etc., as identifiers.

Length: Identifiers can be of any length, but it is recommended to keep them meaningful and not excessively long.

Meaningful Names: It is good practice to choose descriptive and meaningful names for identifiers to enhance code readability and understanding.

Avoid Using Built-in Names: Avoid using names that are already used by Python built-in functions, modules, or libraries to prevent conflicts and confusion. For example, it is best not to use list, str, int, etc., as id

**1.5**

1)In Python, the concepts of shallow copy and deep copy are used when dealing with objects, including lists, dictionaries, and custom objects. The main difference between shallow copy and deep copy lies in how the original object and its copy are related and whether the copied object contains references to the same or new objects.

Shallow Copy:

A shallow copy creates a new object but references the same objects as the original.

Changes made to the copied object will affect the original object, and vice versa.

Shallow copy is created using the copy() method or the slice operator ([:]).

It is a relatively faster operation as it doesn't create new copies of referenced objects.

Deep Copy:

A deep copy creates a new object and recursively copies all objects referenced within the original.

Changes made to the copied object will not affect the original object, and vice versa.

Deep copy is created using the deepcopy() function from the copy module.

It is a relatively slower operation as it creates new copies of all objects, including referenced objects.

2)# List for storing book information

books = [

{

'title': 'Python Crash Course',

'author': 'Eric Matthes',

'isbn': '978-1593279288',

'available': 5

},

{

'title': 'Clean Code: A Handbook of Agile Software Craftsmanship',

'author': 'Robert C. Martin',

'isbn': '978-0132350884',

'available': 3

},

# Add more books...

]

# List for storing student information

students = [

{

'id': 'S001',

'name': 'Alice',

'books\_borrowed': [

'978-1593279288', # ISBN of 'Python Crash Course'

'978-0132350884' # ISBN of 'Clean Code'

]

},

{

'id': 'S002',

'name': 'Bob',

'books\_borrowed': [

'978-1593279288' # ISBN of 'Python Crash Course'

]

},

]

1. Expressions can be assigned or used as operands, while statements can only be declared. Statements create side effects to be useful, while expressions are values or execute to values. Expressions are unique in meaning, while statements are two-sided in execution.
2. Atomic data types - The data types have values that cannot be divided or broken down further. Atomic data types can be either primitive or derived. Numbers and strings are atomic data types because their values cannot be described using smaller parts.There are four primitive atomic data types: booleans, integers, characters and floats.

Secondary data types - It is a data type that derived from an existing data type. You can use secondary data types to extend the built-in types already available and create your own customized data types.

1. A user-defined data type (UDT) is a data type that derived from an existing data type. You can use UDTs to extend the built-in types already available and create your own customized data types.

**1.6**

1. The ten project ideas I want to implement using python are :-
2. Mad Libs Generator
3. Number Guessing
4. Text-based Adventure Game
5. Dice Rolling Simulator
6. Hangman
7. Contact Book
8. Binary search algorithm
9. Desktop Notifier App
10. Python Story Generator
11. Python Website Blocker

2) # Creating a list of 5 objects

objects = []

# Object 1: Laptop

laptop = {

"Brand": "Apple",

"Model": "MacBook Pro",

"RAM": "16GB",

"Storage": "512GB SSD"

}

objects.append(laptop)

# Object 2: Bicycle

bicycle = {

"Brand": "Giant",

"Type": "Mountain Bike",

"Frame Size": "Medium",

"Color": "Black"

}

objects.append(bicycle)

# Object 3: Book

book = {

"Title": "To Kill a Mockingbird",

"Author": "Harper Lee",

"Genre": "Fiction",

"Publication Year": 1960

}

objects.append(book)

# Object 4: Smartphone

smartphone = {

"Brand": "Samsung",

"Model": "Galaxy S21",

"Color": "Phantom Black",

"Storage": "128GB"

}

objects.append(smartphone)

# Object 5: Car

car = {

"Make": "Toyota",

"Model": "Camry",

"Year": 2022,

"Color": "Silver",

"Fuel Type": "Petrol"

}

objects.append(car)

# Printing the list of objects

for obj in objects:

print(obj)

3)Duck typing in Python means that the data type of the variables can change as long as the syntax is valid. In Python, dunder methods are methods that allow instances of a class to interact with the built-in functions and operators of the language.

4)Operators are the constructs which can manipulate the value of operands. These are symbols used for the purpose of logical, arithmetic and various other operations. In Python, there are seven different types of operators: arithmetic operators, assignment operators, comparison operators, logical operators, identity operators, membership operators, and boolean operators.

**1.8**

1. The two values of boolean data type are True and False which represent the logical states of true and false respectively. To write the boolean data types true and false we simply assign the values true and false to variables , like for example:-

A = True

B = False

1. The three boolean operators are AND,OR and NOT.
2. The truth table for the three logic operators are:-

The truth table for the AND gate, which is a logical operator, shows the output based on the combination of two input values. Here is the truth table for the AND gate:

| Input A | Input B | Output |

|---------|---------|--------|

| False | False | False |

| False | True | False |

| True | False | False |

| True | True | True |

In the table above, the AND gate returns `True` only when both input values are `True`. Otherwise, it returns `False`.

The truth table for the OR gate, which is a logical operator, shows the output based on the combination of two input values. Here is the truth table for the OR gate:

| Input A | Input B | Output |

|---------|---------|--------|

| False | False | False |

| False | True | True |

| True | False | True |

| True | True | True |

In the table above, the OR gate returns `True` if at least one of the input values is `True`. It returns `False` only when both input values are `False`.

The truth table for the NOT gate, which is a logical operator, shows the output based on a single input value. Here is the truth table for the NOT gate:

| Input | Output |

|-------|--------|

| False | True |

| True | False |

In the table above, the NOT gate negates the input value. It returns `True` if the input value is `False`, and `False` if the input value is `True`.

1. I)The expression (5>4) and (3==5) evaluates to False.

II) The expression not(5>4) evaluates to False.

III) The expression not ((5 > 4) or (3 == 5)) evaluates to False.

IV) The expression (True and True) and (True == False) evaluates to

False.

V) The expression (not False) or (not True) evaluates to True.

1. The six comparison operators are :- equal to, not equal to, greater than or equal to, less than or equal to , greater than and less than.
2. The equal to value is used to evaluate if two operands are equal to each other or not wheras the assignment operator is used to assign a particular value to a variable.
3. A condition in programming is a logical statement which determines if a particular segment of a code should be executed or not. We can use conditions in various places like loops, functions and if else statements.

**1.9**

1)

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

if number > 0:

print("The number is positive.")

elif number < 0:

print("The number is negative.")

else:

print("The number is zero.")

```

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

if number % 5 == 0 and number % 11 == 0:

print("The number is divisible by both 5 and 11.")

else:

print("The number is not divisible by both 5 and 11.")

3)year = int(input("Enter a year: "))

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

print(year, "is a leap year.")

else:

print(year, "is not a leap year.")

4)character = input("Enter a character: ")

if character.isalpha():

print("The input character is an alphabet.")

else:

print("The input character is not an alphabet.")

5)alphabet = input("Enter an alphabet: ")

if alphabet.lower() in ('a', 'e', 'i', 'o', 'u'):

print("The entered alphabet is a vowel.")

else:

print("The entered alphabet is a consonant.")

6)character = input("Enter a character: ")

if character.isalpha():

print("The entered character is an alphabet.")

elif character.isdigit():

print("The entered character is a digit.")

else:

print("The entered character is a special character.")

7)character = input("Enter a character: ")

if character.isalpha():

if character.isupper():

print("The entered character is an uppercase alphabet.")

else:

print("The entered character is a lowercase alphabet.")

else:

print("The entered character is not an alphabet.")

8)week\_number = int(input("Enter the week number (1-7): "))

if week\_number == 1:

print("Monday")

elif week\_number == 2:

print("Tuesday")

elif week\_number == 3:

print("Wednesday")

elif week\_number == 4:

print("Thursday")

elif week\_number == 5:

print("Friday")

elif week\_number == 6:

print("Saturday")

elif week\_number == 7:

print("Sunday")

else:

print("Invalid week number. Please enter a number between 1 and 7.")

9)angle1 = int(input("Enter the first angle of the triangle: "))

angle2 = int(input("Enter the second angle of the triangle: "))

angle3 = int(input("Enter the third angle of the triangle: "))

# Check if the angles form a valid triangle

if angle1 + angle2 + angle3 == 180 and angle1 > 0 and angle2 > 0 and angle3 > 0:

print("The triangle is valid.")

else:

print("The triangle is not valid.")

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

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

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

# Check if the sides form a valid triangle

if side1 + side2 > side3 and side1 + side3 > side2 and side2 + side3 > side1:

print("The triangle is valid.")

else:

print("The triangle is not valid.")

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

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

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

# Check the type of triangle

if side1 == side2 == side3:

print("The triangle is an equilateral triangle.")

elif side1 == side2 or side1 == side3 or side2 == side3:

print("The triangle is an isosceles triangle.")

else:

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

12)physics = float(input("Enter the marks obtained in Physics: "))

chemistry = float(input("Enter the marks obtained in Chemistry: "))

biology = float(input("Enter the marks obtained in Biology: "))

mathematics = float(input("Enter the marks obtained in Mathematics: "))

computer = float(input("Enter the marks obtained in Computer: "))

total\_marks = physics + chemistry + biology + mathematics + computer

percentage = (total\_marks / 500) \* 100

# Determine the grade based on the percentage

if percentage >= 90:

grade = "A"

elif percentage >= 80:

grade = "B"

elif percentage >= 70:

grade = "C"

elif percentage >= 60:

grade = "D"

elif percentage >= 40:

grade = "E"

else:

grade = "F"

print("Percentage: {:.2f}%".format(percentage))

print("Grade: ", grade)

13)basic\_salary = float(input("Enter the basic salary of the employee: "))

if basic\_salary <= 10000:

hra = basic\_salary \* 0.2

da = basic\_salary \* 0.8

elif basic\_salary <= 20000:

hra = basic\_salary \* 0.25

da = basic\_salary \* 0.9

else:

hra = basic\_salary \* 0.3

da = basic\_salary \* 0.95

gross\_salary = basic\_salary + hra + da

print("Gross Salary: ", gross\_salary)

14)unit\_charges = float(input("Enter the electricity unit charges: "))

total\_bill = 0

if unit\_charges <= 50:

total\_bill = unit\_charges \* 0.50

elif unit\_charges <= 150:

total\_bill = (50 \* 0.50) + ((unit\_charges - 50) \* 0.75)

elif unit\_charges <= 250:

total\_bill = (50 \* 0.50) + (100 \* 0.75) + ((unit\_charges - 150) \* 1.20)

else:

total\_bill = (50 \* 0.50) + (100 \* 0.75) + (100 \* 1.20) + ((unit\_charges - 250) \* 1.50)

total\_bill += total\_bill \* 0.20 # Adding 20% surcharge

print("Total Electricity Bill: Rs.", total\_bill)

1.10

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

original\_num = num

reverse\_num = 0

while num > 0:

digit = num % 10

reverse\_num = (reverse\_num \* 10) + digit

num = num // 10

if original\_num == reverse\_num:

print("The number is a palindrome.")

else:

print("The number is not a palindrome.")

2)start = int(input("Enter the starting number of the interval: "))

end = int(input("Enter the ending number of the interval: "))

print("Multiples of 3 and 5 in the interval [{}, {}]: ".format(start, end))

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

if num % 3 == 0 or num % 5 == 0:

print(num)

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

factorial = 1

if num < 0:

print("Factorial is not defined for negative numbers.")

elif num == 0:

print("The factorial of 0 is 1.")

else:

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

factorial \*= i

print("The factorial of {} is {}.".format(num, factorial))

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

# First two terms of the sequence

first\_term = 0

second\_term = 1

# Check if the number of terms is valid

if num\_terms <= 0:

print("Number of terms should be a positive integer.")

elif num\_terms == 1:

print("Fibonacci sequence up to", num\_terms, "term:")

print(first\_term)

else:

print("Fibonacci sequence up to", num\_terms, "terms:")

print(first\_term)

print(second\_term)

for i in range(2, num\_terms):

next\_term = first\_term + second\_term

print(next\_term)

first\_term = second\_term

second\_term = next\_term

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

order = len(str(num))

sum = 0

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* order

temp //= 10

if num == sum:

print(num, "is an Armstrong number.")

else:

print(num, "is not an Armstrong number.")

**1.11**

11

1)a)n = int(input("Enter the number of rows: "))

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

print('\*' \* i)

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

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

print(str(i) \* i)

c)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()

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

start\_num = 2

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

for j in range(i):

print(start\_num, end='')

start\_num += 1

print()

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

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

for j in range(i):

if (i+j) % 2 == 0:

print("1", end='')

else:

print("0", end='')

print()

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

num = 1

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

for j in range(i):

print(num, end='')

num += 1

print()

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

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

for j in range(n, n-i, -1):

print(j, end='')

print()

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

for i in range(n, 0, -1):

for j in range(i):

print(i, end='')

print()

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

for i in range(n, 0, -1):

print('\*' \* i)

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

num = n

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

print(str(num) \* i)

num -= 1

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

for i in range(n, 0, -1):

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

print(j, end='')

print()

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

for i in range(n, 0, -1):

for j in range(i):

print(i, end='')

print()

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

for i in range(n, 0, -1):

for j in range(n, n-i, -1):

print(j, end='')

print()

2)r)n = int(input("Enter the number of rows: "))

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

spaces = " " \* (n-i)

stars = "\*" \* (2\*i - 1)

print(spaces + stars)

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

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

spaces = " " \* (n-i)

numbers = str(i) \* (2\*i - 1)

print(spaces + numbers)

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

num = 1

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

spaces = " " \* (n-i)

numbers = "".join(str(j) for j in range(1, 2\*i))

print(spaces + numbers)

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

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

spaces = " " \* (n-i)

stars = "\*" \* (2\*i - 1)

print(spaces + stars)

3)v)n = int(input("Enter the number of rows: "))

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

spaces = " " \* (n - i)

numbers = ""

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

numbers += str(j)

for j in range(i-1, 0, -1):

numbers += str(j)

print(spaces + numbers)

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

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

spaces = " " \* (n - i)

numbers = ""

for j in range(n, n-i, -1):

numbers += str(j)

for j in range(n-i+2, n+1):

numbers += str(j)

print(spaces + numbers)

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

for i in range(n, 0, -1):

spaces = " " \* (n - i)

stars = "\*" \* (2\*i - 1)

print(spaces + stars)

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

for i in range(n, 0, -1):

spaces = " " \* (n - i)

numbers = str(i) \* (2\*i - 1)

print(spaces + numbers)

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

for i in range(n, 0, -1):

spaces = " " \* (n - i)

numbers = " ".join(str(x) for x in range(1, 2\*i))

print(spaces + numbers)

z1)n = int(input("Enter the number of rows: "))

for i in range(n, 0, -1):

spaces = " " \* (n - i)

numbers = " ".join(str(x) for x in range(i, 2\*i))

print(spaces + numbers + str(i))

z2)n = int(input("Enter the number of rows: "))

for i in range(n, 0, -1):

spaces = " " \* (n - i)

numbers = " ".join(str(x) for x in range(n, n-i, -1))

print(spaces + numbers + numbers[::-1])

4)z3)n = int(input("Enter the number of rows: "))

# Upper half of the pattern

for i in range(n):

spaces = " " \* (n - i - 1)

stars = "\*" \* (2 \* i + 1)

print(spaces + stars)

# Lower half of the pattern

for i in range(n - 2, -1, -1):

spaces = " " \* (n - i - 1)

stars = "\*" \* (2 \* i + 1)

print(spaces + stars)

z4)n = int(input("Enter the number of rows: "))

# Upper half of the pattern

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

spaces = " " \* (n - i)

numbers = str(i) \* (2 \* i - 1)

print(spaces + numbers)

# Lower half of the pattern

for i in range(n - 1, 0, -1):

spaces = " " \* (n - i)

numbers = str(i) \* (2 \* i - 1)

print(spaces + numbers)

z5)n = int(input("Enter the number of rows: "))

# Upper half of the pattern

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

spaces = " " \* (n - i)

numbers = "".join(str(x) for x in range(1, 2 \* i))

print(spaces + numbers)

# Lower half of the pattern

for i in range(n - 1, 0, -1):

spaces = " " \* (n - i)

numbers = "".join(str(x) for x in range(1, 2 \* i))

print(spaces + numbers)

z6)n = int(input("Enter the number of rows: "))

# Upper half of the pattern

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

spaces = " " \* (n - i)

numbers = "".join(str(x % 10) for x in range(i, 2 \* i))

print(spaces + numbers + str(i))

# Lower half of the pattern

for i in range(n - 1, 0, -1):

spaces = " " \* (n - i)

numbers = "".join(str(x % 10) for x in range(i, 2 \* i))

print(spaces + numbers + str(i))

z7)n = int(input("Enter the number of rows: "))

# Upper half of the pattern

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

spaces = " " \* (n - i)

numbers = "".join(str(x % 10) for x in range(n, n - i, -1))

print(spaces + numbers)

# Lower half of the pattern

for i in range(n - 1, 0, -1):

spaces = " " \* (n - i)

numbers = "".join(str(x % 10) for x in range(n, n - i, -1))

print(spaces + numbers)

1. To print the pattern

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

The python code is given below

n = 7 # Number of rows

for i in range(n):

print("\*" \* n)

To print the pattern

\*\*\*\*\*\*\*

\* \*

\* \*

\* \*

\* \*

\* \*

\*\*\*\*\*\*\*

The python code is given below

n = 7 # Number of rows

for i in range(n):

if i == 0 or i == n-1:

print("\*" \* n)

else:

print("\*" + " " \* (n-2) + "\*")

To print the pattern

\\*\*\*\*\*/

\*\\*\*\*/\*

\*\*\\*/\*\*

\*\*\*\\*\*

\*\*/\*\\*\*

\*/\*\*\*\\*

/\*\*\*\*\*\

The python program is given below

n = 7 # Number of rows

for i in range(n):

pattern = ""

for j in range(n):

if j == i:

pattern += "\\"

elif j == n - i - 1:

pattern += "/"

else:

pattern += "\*"

print(pattern)

5)a)n = 11 # Number of rows

m = 33 # Number of columns

welcome = "WELCOME"

for i in range(n):

if i == n // 2:

line = welcome.center(m, "-")

else:

line = ".|." \* (2\*i + 1)

line = line.center(m, "-")

print(line)

b)n = 17 # Number of rows

for i in range(n):

line = ""

for j in range(n):

if j <= i:

line += chr(96 + n - j)

else:

line += "-"

for j in range(n - 1, -1, -1):

if j <= i:

line += chr(97 + n - j)

else:

line += "-"

print(line.center(2\*n - 1, "-"))

**1.12**

12

1)import random

def guess\_game():

# Generate a random number between 1 and 50

actual\_number = random.randint(1, 50)

attempts = 5

print("Welcome to the Guess Game!")

print("Guess the number between 1 and 50.")

while attempts > 0:

try:

guess = int(input("Enter your guess: "))

if guess < 1 or guess > 50:

print("Think within the limits 1-50 only!")

continue

if guess == actual\_number:

print("Congratulations! You guessed the correct number!")

return

if guess < actual\_number:

print("Hint: Think of a bigger number!")

else:

print("Hint: Think of a smaller number!")

attempts -= 1

print("Attempts left:", attempts)

except ValueError:

print("Invalid input. Please enter a valid number.")

print("Game over. You have run out of attempts.")

print("The actual number was:", actual\_number)

guess\_game()

2)x = int(input("Enter the value of x: "))

y = int(input("Enter the value of y: "))

z = int(input("Enter the value of z: "))

n = int(input("Enter the value of n: "))

coordinates = [[i, j, k] for i in range(x + 1) for j in range(y + 1) for k in range(z + 1) if (i + j + k) != n]

print("List of possible coordinates:")

print(coordinates)

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

list\_ = []

for \_ in range(n):

command = input("Enter the command: ").split()

if command[0] == "insert":

index = int(command[1])

element = int(command[2])

list\_.insert(index, element)

elif command[0] == "print":

print(list\_)

elif command[0] == "remove":

element = int(command[1])

list\_.remove(element)

elif command[0] == "append":

element = int(command[1])

list\_.append(element)

elif command[0] == "sort":

list\_.sort()

elif command[0] == "pop":

list\_.pop()

elif command[0] == "reverse":

list\_.reverse()

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

student\_records = {}

for \_ in range(n):

name, \*marks = input("Enter the name and marks: ").split()

marks = list(map(int, marks))

student\_records[name] = marks

query\_name = input("Enter the name of the student to query: ")

if query\_name in student\_records:

marks = student\_records[query\_name]

average = sum(marks) / len(marks)

print("{:.2f}".format(average))

else:

print("Student not found.")

1. a)def bubble\_sort(arr):

n = len(arr)

# Traverse through all array elements

for i in range(n - 1):

# Last i elements are already in place

for j in range(n - i - 1):

# Swap if the element found is greater than the next element

if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

return arr

b)def insertion\_sort(arr):

n = len(arr)

# Traverse through 1 to n

for i in range(1, n):

key = arr[i]

j = i - 1

# Move elements of arr[0..i-1], that are greater than key, to one position ahead

while j >= 0 and arr[j] > key:

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = key

return arr

c)def selection\_sort(arr):

n = len(arr)

# Traverse through 0 to n-1

for i in range(n - 1):

# Find the minimum element in the unsorted part of the list

min\_idx = i

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

if arr[j] < arr[min\_idx]:

min\_idx = j

# Swap the found minimum element with the first element of the unsorted part

arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

return arr

**1.13**

1)def count\_vowels(string):

vowels = 'aeiou'

vowel\_count = {}

# Initializing the vowel count dictionary

for vowel in vowels:

vowel\_count[vowel] = 0

# Counting the vowels in the string

for char in string.lower():

if char in vowels:

vowel\_count[char] += 1

return vowel\_count

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

vowel\_counts = count\_vowels(input\_string)

print("Number of each vowel in the string:")

for vowel, count in vowel\_counts.items():

print(vowel, ":", count)

2)def count\_statistics(string):

word\_count = len(string.split())

char\_count = len(string)

space\_count = string.count(' ')

return word\_count, char\_count, space\_count

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

word\_count, char\_count, space\_count = count\_statistics(input\_string)

print("Number of words:", word\_count)

print("Number of characters:", char\_count)

print("Number of spaces:", space\_count)

3)def remove\_punctuation(string):

punctuations = '''!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~'''

new\_string = ""

for char in string:

if char not in punctuations:

new\_string += char

return new\_string

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

result = remove\_punctuation(input\_string)

print("String after removing punctuation:", result)

4)def create\_matrix(m, n):

matrix = []

for i in range(m):

row = []

for j in range(n):

element = int(input("Enter element at position ({}, {}): ".format(i, j)))

row.append(element)

matrix.append(row)

return matrix

# Get matrix dimensions from the user

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

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

# Create the matrix

matrix = create\_matrix(m, n)

# Print the matrix

print("Matrix:")

for row in matrix:

print(row)

5)def create\_matrix(m, n):

matrix = []

for i in range(m):

row = []

for j in range(n):

element = int(input("Enter element at position ({}, {}): ".format(i, j)))

row.append(element)

matrix.append(row)

return matrix

def add\_matrices(matrix1, matrix2):

result = []

for i in range(len(matrix1)):

row = []

for j in range(len(matrix1[0])):

element = matrix1[i][j] + matrix2[i][j]

row.append(element)

result.append(row)

return result

def multiply\_matrices(matrix1, matrix2):

result = []

for i in range(len(matrix1)):

row = []

for j in range(len(matrix2[0])):

element = 0

for k in range(len(matrix2)):

element += matrix1[i][k] \* matrix2[k][j]

row.append(element)

result.append(row)

return result

def transpose\_matrix(matrix):

result = []

for j in range(len(matrix[0])):

row = []

for i in range(len(matrix)):

element = matrix[i][j]

row.append(element)

result.append(row)

return result

# Get matrix dimensions from the user

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

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

print("Enter elements of Matrix 1:")

matrix1 = create\_matrix(m, n)

print("Enter elements of Matrix 2:")

matrix2 = create\_matrix(m, n)

# Addition of matrices

addition\_result = add\_matrices(matrix1, matrix2)

print("Addition of the two matrices:")

for row in addition\_result:

print(row)

# Multiplication of matrices

if len(matrix1[0]) == len(matrix2):

multiplication\_result = multiply\_matrices(matrix1, matrix2)

print("Multiplication of the two matrices:")

for row in multiplication\_result:

print(row)

else:

print("Cannot multiply the matrices. The number of columns in Matrix 1 must be equal to the number of rows in Matrix 2.")

# Transpose of matrices

transpose\_result1 = transpose\_matrix(matrix1)

transpose\_result2 = transpose\_matrix(matrix2)

print("Transpose of Matrix 1:")

for row in transpose\_result1:

print(row)

print("Transpose of Matrix 2:")

for row in transpose\_result2:

print(row)

**1.14**

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

while num > 9:

# Calculate the sum of digits

sum\_digits = 0

while num > 0:

sum\_digits += num % 10

num //= 10

num = sum\_digits

return num

# Input a number from the user

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

# Find the sum of digits until reduced to 1 digit

result = sum\_of\_digits(number)

print("Sum of digits until reduced to 1 digit:", result)

2)def find\_lcm(x, y):

# Find the maximum of the two numbers

max\_num = max(x, y)

# Find the LCM

lcm = max\_num

while True:

if lcm % x == 0 and lcm % y == 0:

break

lcm += max\_num

return lcm

def find\_hcf(x, y):

# Find the minimum of the two numbers

min\_num = min(x, y)

# Find the HCF

hcf = 1

for i in range(1, min\_num + 1):

if x % i == 0 and y % i == 0:

hcf = i

return hcf

# Input two numbers from the user

num1 = int(input("Enter first number: "))

num2 = int(input("Enter second number: "))

# Find the LCM

lcm = find\_lcm(num1, num2)

print("LCM of", num1, "and", num2, "is:", lcm)

# Find the HCF

hcf = find\_hcf(num1, num2)

print("HCF of", num1, "and", num2, "is:", hcf)

3)n = int(input("Enter a number: "))

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

if i % 3 != 0:

continue

cube = i \*\* 3

print("Cube of", i, "is:", cube)

4)def get\_valid\_roll\_number():

while True:

roll\_number = int(input("Enter roll number (between 1000 and 9999): "))

if 1000 <= roll\_number <= 9999:

return roll\_number

else:

print("Invalid roll number. Please try again.")

# Initialize variables

total\_marks\_above\_200 = 0

highest\_total\_marks = 0

highest\_total\_roll\_numbers = []

# Read roll number and marks for 10 students

for \_ in range(10):

print("Enter details for student", \_ + 1)

roll\_number = get\_valid\_roll\_number()

# Read marks for 3 subjects

marks = []

for subject in range(3):

marks.append(int(input("Enter marks for subject {}: ".format(subject + 1))))

# Calculate total marks

total\_marks = sum(marks)

# Check if student gets more than or equal to 40 marks in each subject

if all(mark >= 40 for mark in marks):

# Increment the count if total marks are more than 200

if total\_marks > 200:

total\_marks\_above\_200 += 1

# Check if the current student has the highest total marks

if total\_marks > highest\_total\_marks:

highest\_total\_marks = total\_marks

highest\_total\_roll\_numbers = [roll\_number]

elif total\_marks == highest\_total\_marks:

highest\_total\_roll\_numbers.append(roll\_number)

# Print the results

print("Number of students with total marks above 200:", total\_marks\_above\_200)

print("Roll number(s) of student(s) with the highest total marks:", highest\_total\_roll\_numbers)

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

# Handle the case when the input number is 0

if num == 0:

digit\_count = 1

else:

digit\_count = 0

# Take the absolute value of the number to handle negative numbers

num = abs(num)

# Count the digits using a while loop

while num > 0:

num //= 10

digit\_count += 1

print("Number of digits:", digit\_count)

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

reverse\_num = 0

temp = num

# Reverse the number

while temp > 0:

digit = temp % 10

reverse\_num = reverse\_num \* 10 + digit

temp //= 10

# Double the reverse number

double\_reverse\_num = reverse\_num \* 2

print("Reverse number:", reverse\_num)

print("Double of the reverse number:", double\_reverse\_num)