Assignment 1

Name-Abhishek Sahani

Question 1:Explain the key features of python that make it a popular

choice for programming.

Ans1: features of python

Readability and simplicity

Wide range of libraries and framework

Object-oriented programming capabilities

Easy to learn and use

Large community and Ecosystem

Strong support for integration

High level language

Versatile and Multi-Paradigm

Question 2:Describe the role of predefined keywords in python and

provide examples of how they are used in a program.

Ans 2: python keywords are some predefine and reserved words in python that have

special meanings. Keywords are used to define the syntax of the coding. The

keywords cannot be used as an identifier , function, or variable name. All keywords in

python are written in lower case except True and false

HowThey Are Used:

1. Control Flow: if, elif, else, for, while manage conditional and repetitive execution.

2. Function/Class Definition: def, class define reusable code blocks and data

structures.

3. Exception Handling: try, except, finally manage errors.

4. Logical Operations: and, or, not handle Boolean logic.

5. Loops: break, continue control loop behavior.

Example:

python

try:

for i in range(5):

if i == 3:

break # Stops the loop

print(i)

except Exception as e:

print("An error occurred:", e)

Question 3: compare the constraint mutable and immutable objects in

python with examples.

Ans 3: Mutable objects : (example lists,dictionaries ) can be change after creation

allowing modifications without creating new objects.

Example of mutable : x=[1,2,3,4]

x[0]=10

print(x)

Output: [10,2,3,4]

Immutable objects: (eg. tuple , strings) cannot be change once created; any

modification result in a new objects.

Example of immutable: y=”hello”

y[0]=’m’

print(y)

Output: error because string are immutable

Question 4: Discuss the different type of operators in python and

provide examples of how they are used .

Ans: Here's a concise overview of Python operators:

1. Arithmetic: +,-, \*, /, %, \*\*, //- Example: 5 + 3 → 8

2. Comparison: ==, !=, >, <, >=, <=- Example: 5 > 3 → True

3. Logical: and, or, not- Example: True and False → False

4. Assignment: =, +=,-=, \*=, /=- Example: x += 2 (increments x by 2)

5. Bitwise: &, |, ^, ~, <<, >>- Example: 5 & 3 → 1

6. Identity: is, is not- Example: a is b (checks if a and b are the same object)

7. Membership: in, not in- Example: 'a' in 'apple' → True

Question 5: Explain the concept of type casting in Python with examples.

Ans 5: Typecasting in Python is converting a variable from one data type to another.

This is useful for performing operations or ensuring compatibility between different

types.

### Examples:

1. Converting to Integer:

python

num\_str = "42"

num\_int = int(num\_str) # Converts string to integer

print(num\_int) # Output: 42

2. Converting to String:

python

num=100

num\_str = str(num) # Converts integer to string

print(num\_str) # Output: "100"

3. Converting to Float:

python

num\_str = "3.14"

num\_float = float(num\_str) # Converts string to float

print(num\_float) # Output: 3.14

Question 6: How do conditional statements work in Python? Illustrate

with examples.

Ans 6: Conditional statements in Python control the flow of execution based on

conditions. They evaluate expressions and execute code blocks if the conditions are

true.

### Types of Conditional Statements:

1. \*if\*: Executes a block if the condition is true.

python

x =10

if x > 5:

print("x is greater than 5") # Output: x is greater than 5

2. \*elif\*: Provides additional conditions if the if condition is false.

python

x =10

if x > 15:

print("x is greater than 15")

elif x > 5:

print("x is greater than 5") # Output: x is greater than 5

3. \*else\*: Executes a block if none of the previous conditions are true.

python

x =3

if x > 5:

print("x is greater than 5")

else:

print("x is 5 or less") # Output: x is 5 or less

Question 7:Describe the different types of loops in Python and their use

cases with examples.

Ans 7: Python supports three main types of loops:

1. for Loop: Iterates over a sequence (like a list, tuple, or string) or range of numbers.- Use Case: When you need to iterate through items in a collection or perform an

action a specific number of times.- Example:

python

for i in range(3):

print(i) # Output: 0 1 2

2. while Loop: Repeats as long as a condition is true.- Use Case: When the number of iterations is not known beforehand and depends

on a condition.- Example:

python

count = 0

while count < 3:

print(count) # Output: 0 1 2

count += 1

3. nested Loops: Loops within loops.- Use Case: When you need to perform operations on multi-dimensional data (e.g.,

matrices) or combinations of items.- Example:

python

for i in range(2):

for j in range(2):

print(i, j) # Output: 0 0 0 1 1 0 1 1