**Name : Hadvani Shruti Vipulbhai**

**Module – 7 DA**

**Introduction to Python**

**1) What are the types of Applications?**

Applications are generally categorized as:

* **Web Applications** – Runs on web browsers (e.g., Gmail, Facebook).
* **Mobile Applications** – Built for smartphones and tablets (e.g., WhatsApp, Instagram).
* **Desktop Applications** – Installed and run on personal computers (e.g., MS Word).
* **Console Applications** – Run in command-line interface, without GUI.
* **Enterprise Applications** – Large-scale apps used in organizations (e.g., ERP systems).
* **Cloud-based Applications** – Accessed via cloud servers (e.g., Google Drive).
* **Games** – Interactive applications for entertainment.

**2) What is Programming?**

**Programming** is the process of writing instructions (code) that a computer can follow to perform a specific task or solve a problem.

**3) What is Python?**

**Python** is a high-level, interpreted, general-purpose programming language. It is known for its simple syntax and readability. It's widely used in web development, data science, automation, AI, and more.

**4) Python program to check if a number is positive, negative or zero:**

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

if num > 0:

print("Positive number")

elif num < 0:

print("Negative number")

else:

print("Zero")

**5) Python program to get the Factorial of a number:**

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

factorial = 1

if num < 0:

print("Factorial does not exist for negative numbers")

elif num == 0:

print("Factorial is 1")

else:

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

factorial \*= i

print("Factorial of", num, "is", factorial)

**6) Python program to get the Fibonacci series of given range:**

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

a, b = 0, 1

print("Fibonacci Series:")

for i in range(n):

print(a, end=" ")

a, b = b, a + b

**7) How is memory managed in Python?**

* **Automatic Garbage Collection** – Python automatically deletes unused objects to free memory.
* **Private heap space** – All Python objects and data structures are stored in a private heap.
* **Memory management modules** like gc help in controlling the garbage collection process.

**8) What is the purpose of the continue statement in Python?**

The continue statement is used to **skip the current iteration** of a loop and jump to the next one.

for i in range(5):

if i == 2:

continue

print(i)

**9) Python program to swap two numbers (with and without temp variable):**

**With temp:**

a = 5

b = 10

temp = a

a = b

b = temp

print("After swapping: a =", a, ", b =", b)

**Without temp:**

a = 5

b = 10

a, b = b, a

print("After swapping: a =", a, ", b =", b)

**10) Python program to check if a number is even or odd:**

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

if num % 2 == 0:

print("Even number")

else:

print("Odd number")

**11) Python program to check if a letter is a vowel:**

letter = input("Enter a letter: ").lower()

if letter in ['a', 'e', 'i', 'o', 'u']:

print("It's a vowel")

else:

print("It's not a vowel")

**12) Python program to sum of three integers; if two values are equal, sum is 0:**

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

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

c = int(input("Enter third number: "))

if a == b or b == c or a == c:

print("Sum is 0")

else:

print("Sum is:", a + b + c)

**13) Python program to return true if two integers are equal or their sum/difference is 5:**

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

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

if a == b or abs(a - b) == 5 or (a + b) == 5:

print("True")

else:

print("False")

**14) Write a Python program to sum of the first n positive integers:**

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

sum\_n = n \* (n + 1) // 2

print("Sum of first", n, "positive integers is:", sum\_n)

**15) Write a Python program to calculate the length of a string:**

string = input("Enter a string: ")

print("Length of the string is:", len(string))

**16) Write a Python program to count the number of characters (character frequency) in a string:**

string = input("Enter a string: ")

frequency = {}

for char in string:

if char in frequency:

frequency[char] += 1

else:

frequency[char] = 1

print("Character frequency:")

for char, count in frequency.items():

print(f"{char}: {count}")

**17) What are negative indexes and why are they used?**

**Negative indexes** in Python are used to access elements from the **end of a list or string**.

* -1 refers to the last element,
* -2 refers to the second last, and so on.

They are useful when we don’t know the exact length but want to access values from the end.

Example:

text = "Python"

print(text[-1]) # Output: n

print(text[-3]) # Output: h

**18) Write a Python program to count occurrences of a substring in a string:**

string = input("Enter the main string: ")

substring = input("Enter the substring to count: ")

count = string.count(substring)

print(f"'{substring}' occurs {count} times in the given string.")

**19) Write a Python program to count the occurrences of each word in a given sentence:**

sentence = input("Enter a sentence: ")

words = sentence.split()

word\_count = {}

for word in words:

word = word.lower() # Optional: makes it case-insensitive

if word in word\_count:

word\_count[word] += 1

else:

word\_count[word] = 1

print("Word frequencies:")

for word, count in word\_count.items():

print(f"{word}: {count}")

**20) Write a Python program to get a single string from two given strings, separated by a space, and swap the first two characters of each string:**

str1 = input("Enter first string: ")

str2 = input("Enter second string: ")

new\_str1 = str2[:2] + str1[2:]

new\_str2 = str1[:2] + str2[2:]

combined = new\_str1 + " " + new\_str2

print("Resulting string:", combined)

**21) Python program to modify a string based on ending:**

def modify\_string(s):

if len(s) < 3:

return s

elif s.endswith("ing"):

return s + "ly"

else:

return s + "ing"

# Example

string = input("Enter a string: ")

print("Modified string:", modify\_string(string))

**22) Python function to reverse a string if its length is a multiple of 4:**

def reverse\_if\_multiple\_of\_four(s):

if len(s) % 4 == 0:

return s[::-1]

return s

# Example

string = input("Enter a string: ")

print("Result:", reverse\_if\_multiple\_of\_four(string))

**23) Python program to get a string made of the first 2 and last 2 chars of a given string:**

def first\_last\_two(s):

if len(s) < 2:

return ""

return s[:2] + s[-2:]

# Example

string = input("Enter a string: ")

print("Result:", first\_last\_two(string))

**24) Python function to insert a string in the middle of another string:**

def insert\_middle(main\_str, insert\_str):

middle\_index = len(main\_str) // 2

return main\_str[:middle\_index] + insert\_str + main\_str[middle\_index:]

# Example

main = input("Enter main string: ")

insert = input("Enter string to insert: ")

print("Modified string:", insert\_middle(main, insert))

**25) What is a List? How will we reverse a list?**

* A **list** in Python is a **collection of ordered, changeable items**, and it allows **duplicates**.
* Lists are created using square brackets:
* my\_list = [1, 2, 3]

**To reverse a list:**

my\_list = [1, 2, 3, 4]

my\_list.reverse()

print(my\_list)

Or using slicing:

reversed\_list = my\_list[::-1]

print(reversed\_list)

**26) How will we remove the last object from a list?**

my\_list = [10, 20, 30, 40]

my\_list.pop()

print(my\_list)

**27) Suppose list1 = [2, 33, 222, 14, 25], what is list1[-1]?**

list1[-1] gives the **last element** of the list.  
So the result is:

25

**28) Differentiate between append() and extend() methods:**

| **Feature** | **append()** | **extend()** |
| --- | --- | --- |
| Function | Adds **one item** to the end of the list | Adds **all elements** from another iterable |
| Syntax | list.append(element) | list.extend(iterable) |
| Example | a.append([4, 5]) → [1, 2, 3, [4, 5]] | a.extend([4, 5]) → [1, 2, 3, 4, 5] |

### ****29) Python function to get the largest number, smallest number, and sum of all from a list:****

def list\_summary(numbers):

largest = max(numbers)

smallest = min(numbers)

total = sum(numbers)

return largest, smallest, total

# Example

num\_list = [10, 25, 5, 70, 40]

largest, smallest, total = list\_summary(num\_list)

print("Largest:", largest)

print("Smallest:", smallest)

print("Sum:", total)

### ****30) How will we compare two lists?****

We can compare two lists using the **equality operator (==)**:

list1 = [1, 2, 3]

list2 = [1, 2, 3]

list3 = [3, 2, 1]

print(list1 == list2) # True (same order & elements)

print(list1 == list3) # False (different order)

We can also compare **sorted lists** to check if they have the same elements regardless of order:

print(sorted(list1) == sorted(list3)) # True

### ****31) Python program to count strings with length ≥2 and same first and last character:****

def match\_ends(words):

count = 0

for word in words:

if len(word) >= 2 and word[0] == word[-1]:

count += 1

return count

# Example

word\_list = ['abc', 'xyz', 'aba', '1221']

print("Count:", match\_ends(word\_list))

### ****32) Python program to remove duplicates from a list:****

my\_list = [1, 2, 2, 3, 4, 4, 5]

unique\_list = list(set(my\_list))

print("List after removing duplicates:", unique\_list)

### ****33) Python program to check if a list is empty or not:****

my\_list = []

if not my\_list:

print("The list is empty.")

else:

print("The list is not empty.")

**34) Function to check if two lists have at least one common member:**

def common\_member(list1, list2):

for item in list1:

if item in list2:

return True

return False

# Example

print(common\_member([1, 2, 3], [4, 5, 3])) # Output: True

**35) Generate and print list of first and last 5 squares between 1 and 30:**

squares = [i\*\*2 for i in range(1, 31)]

print("First 5 squares:", squares[:5])

print("Last 5 squares:", squares[-5:])

**36) Function that returns unique elements from a list:**

def unique\_list(lst):

return list(set(lst))

# Example

print(unique\_list([1, 2, 2, 3, 4, 4, 5]))

**37) Convert a list of characters into a string:**

char\_list = ['P', 'y', 't', 'h', 'o', 'n']

string = ''.join(char\_list)

print("String:", string)

**38) Select a random item from a list:**

import random

items = [10, 20, 30, 40, 50]

print("Random item:", random.choice(items))

**39) Find the second smallest number in a list:**

def second\_smallest(numbers):

unique\_nums = list(set(numbers))

unique\_nums.sort()

return unique\_nums[1]

# Example

print("Second smallest:", second\_smallest([5, 1, 4, 3, 2, 1]))

**40) Get unique values from a list:**

my\_list = [1, 2, 2, 3, 4, 4, 5]

unique\_values = list(set(my\_list))

print("Unique values:", unique\_values)

**41) Check whether a list contains a sub list:**

def is\_sublist(main\_list, sub\_list):

for i in range(len(main\_list) - len(sub\_list) + 1):

if main\_list[i:i+len(sub\_list)] == sub\_list:

return True

return False

# Example

print(is\_sublist([1, 2, 3, 4, 5], [3, 4])) # Output: True

**42) Split a list into different variables:**

my\_list = [10, 20, 30]

a, b, c = my\_list

print("a:", a, "b:", b, "c:", c)

**43) What is a Tuple? Difference between List and Tuple:**

* A **tuple** is an **ordered, immutable** collection of elements.
* **List** is mutable (can be changed), but **Tuple** is immutable (cannot be changed).

| **Feature** | **List** | **Tuple** |
| --- | --- | --- |
| Syntax | [] (square brackets) | () (round brackets) |
| Mutability | Mutable | Immutable |
| Methods Available | Many (append, pop) | Few (count, index) |
| Performance | Slower | Faster |

**44) Create a tuple with different data types:**

mixed\_tuple = (10, "Hello", 3.14, True)

print(mixed\_tuple)

**45) Unzip a list of tuples into individual lists:**

pairs = [(1, 'a'), (2, 'b'), (3, 'c')]

num, char = zip(\*pairs)

print("Numbers:", list(num))

print("Characters:", list(char))

### ****46) Convert a list of tuples into a dictionary:****

list\_of\_tuples = [("a", 1), ("b", 2), ("c", 3)]

dictionary = dict(list\_of\_tuples)

print("Dictionary:", dictionary)

### ****47) Create a dictionary using tuples in Python:****

# Each tuple has two values: (key, value)

tuple\_list = (("name", "Shuti"), ("age", 23), ("city", "Surat"))

my\_dict = dict(tuple\_list)

print("Dictionary from tuples:", my\_dict)

### ****48) Sort a dictionary by value (ascending and descending):****

my\_dict = {'apple': 10, 'banana': 5, 'cherry': 20}

# Ascending order

asc\_sorted = dict(sorted(my\_dict.items(), key=lambda item: item[1]))

print("Ascending:", asc\_sorted)

# Descending order

desc\_sorted = dict(sorted(my\_dict.items(), key=lambda item: item[1], reverse=True))

print("Descending:", desc\_sorted)

### ****49) Concatenate the following dictionaries to create a new one:****

dict1 = {1: "a", 2: "b"}

dict2 = {3: "c", 4: "d"}

dict3 = {5: "e", 6: "f"}

# Using update()

new\_dict = {}

new\_dict.update(dict1)

new\_dict.update(dict2)

new\_dict.update(dict3)

print("Concatenated Dictionary:", new\_dict)

### ****50) Check if a given key already exists in a dictionary:****

my\_dict = {"name": "Shuti", "age": 23}

key = "age"

if key in my\_dict:

print(f"'{key}' key exists in the dictionary.")

else:

print(f"'{key}' key does not exist in the dictionary.")

### ****51) How do we traverse through a dictionary in Python?****

We can **loop through keys and values** like this:

my\_dict = {"a": 1, "b": 2, "c": 3}

for key, value in my\_dict.items():

print("Key:", key, "| Value:", value)

### ****52) How do we check the presence of a key in a dictionary?****

We can use the **in keyword**:

my\_dict = {"x": 100, "y": 200}

if "x" in my\_dict:

print("Key 'x' is present.")

else:

print("Key 'x' is not present.")

### ****53) Print a dictionary where keys are numbers between 1 and 15:****

my\_dict = {x: x\*x for x in range(1, 16)}

print("Dictionary with squares from 1 to 15:", my\_dict)

### ****54) Check if multiple keys exist in a dictionary:****

def check\_keys\_exist(dictionary, keys):

return all(key in dictionary for key in keys)

my\_dict = {'a': 1, 'b': 2, 'c': 3}

keys\_to\_check = ['a', 'b']

if check\_keys\_exist(my\_dict, keys\_to\_check):

print("All keys exist in the dictionary.")

else:

print("One or more keys are missing.")

### ****55) Merge two Python dictionaries:****

dict1 = {'a': 100, 'b': 200}

dict2 = {'c': 300, 'd': 400}

# Method 1: Using the update() method

merged\_dict = dict1.copy()

merged\_dict.update(dict2)

# Method 2: Using dictionary unpacking (Python 3.5+)

# merged\_dict = {\*\*dict1, \*\*dict2}

print("Merged dictionary:", merged\_dict)

### ****56) Map two lists into a dictionary:****

from collections import Counter

keys = ['a', 'b', 'c', 'd']

values = [400, 400, 300, 400]

mapped\_dict = dict(zip(keys, values))

counted = Counter(mapped\_dict)

print("Mapped dictionary with Counter:", counted)

### ****57) Find the highest 3 values in a dictionary:****

my\_dict = {'a': 100, 'b': 300, 'c': 200, 'd': 400, 'e': 150}

top\_three = sorted(my\_dict.values(), reverse=True)[:3]

print("Top 3 highest values:", top\_three)

### ****58) Combine values in a list of dictionaries:****

from collections import Counter

data = [

{'item': 'item1', 'amount': 400},

{'item': 'item2', 'amount': 300},

{'item': 'item1', 'amount': 750}

]

result = Counter()

for d in data:

result[d['item']] += d['amount']

print("Combined values:", result)

# Output: Counter({'item1': 1150, 'item2': 300})

### ****59) Create a dictionary from a string (counting letters):****

from collections import Counter

def string\_to\_dict(s):

return dict(Counter(s))

# Example

input\_string = "hello"

print("Letter count:", string\_to\_dict(input\_string))

### ****60) Count character frequency in a given string:****

from collections import Counter

sample\_string = 'w3resource'

result = dict(Counter(sample\_string))

print("Character frequencies:", result)

# Output: {'w': 1, '3': 1, 'r': 2, 'e': 2, 's': 1, 'o': 1, 'u': 1, 'c': 1}

### ****61) Function to calculate factorial of a number:****

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

# Example

print("Factorial of 5:", factorial(5))

### ****62) Check whether a number is in a given range:****

def in\_range(num, start, end):

return start <= num <= end

# Example

print("Is 5 in range 1 to 10?", in\_range(5, 1, 10)) # Output: True

### ****63) Check whether a number is perfect or not:****

A perfect number is a number that is equal to the sum of its proper positive divisors, excluding itself.

def is\_perfect(n):

if n < 1:

return False

total = sum(i for i in range(1, n) if n % i == 0)

return total == n

# Example

print("Is 28 a perfect number?", is\_perfect(28)) # Output: True

**64) Check if a string is a palindrome:**

def is\_palindrome(s):

return s == s[::-1]

# Example

print("Is 'madam' a palindrome?", is\_palindrome("madam")) # Output: True

**65) How many basic types of functions are available in Python?**

There are **two basic types of functions** in Python:

1. **Built-in Functions** – Provided by Python (e.g., print(), len(), range())
2. **User-defined Functions** – Functions created by the programmer using def

**66) Pick a random item from a list or tuple:**

import random

my\_list = [10, 20, 30, 40]

print("Random item from list:", random.choice(my\_list))

my\_tuple = ('a', 'b', 'c')

print("Random item from tuple:", random.choice(my\_tuple))

**67) Pick a random item from a range:**

import random

print("Random number from range 1 to 10:", random.choice(range(1, 11)))

**68) Get a random number in Python:**

import random

# Random float between 0 and 1

print("Random float (0-1):", random.random())

# Random integer between 10 and 50

print("Random integer (10-50):", random.randint(10, 50))

**69) Set the starting value in generating random numbers:**

We can use random.seed() to set the starting point (for reproducibility):

import random

random.seed(10) # Sets the seed

print("Random number with seed 10:", random.randint(1, 100))

**70) Randomize the items of a list in place:**

import random

my\_list = [1, 2, 3, 4, 5]

random.shuffle(my\_list)

print("Shuffled list:", my\_list)

**71) What is a File function in Python? What are keywords to create and write a file?**

* **File functions** in Python help to create, read, write, and modify files.
* Common keywords/functions:
  + open(filename, mode) – Open a file ('w' for write, 'a' for append, 'r' for read)
  + write() – Write content to file
  + read() – Read content from file
  + close() – Close the file after use

**72) Read an entire text file:**

with open('sample.txt', 'r') as file:

content = file.read()

print(content)

**73) Append text to a file and display the text:**

with open('sample.txt', 'a') as file:

file.write("\nThis is an appended line.")

with open('sample.txt', 'r') as file:

print(file.read())

**74) Read first n lines of a file:**

def read\_first\_n\_lines(filename, n):

with open(filename, 'r') as file:

for i in range(n):

print(file.readline().strip())

**75) Read last n lines of a file:**

def read\_last\_n\_lines(filename, n):

with open(filename, 'r') as file:

lines = file.readlines()

for line in lines[-n:]:

print(line.strip())

**76) Read file line by line and store in a list:**

with open('sample.txt', 'r') as file:

lines = file.readlines()

print(lines)

**77) Read file line by line and store in a variable:**

with open('sample.txt', 'r') as file:

content = ""

for line in file:

content += line

print(content)

**78) Find the longest words in a file:**

def longest\_words(filename):

with open(filename, 'r') as file:

words = file.read().split()

max\_len = len(max(words, key=len))

return [word for word in words if len(word) == max\_len]

print(longest\_words('sample.txt'))

**79) Count number of lines in a file:**

with open('sample.txt', 'r') as file:

lines = file.readlines()

print("Number of lines:", len(lines))

**80) Count frequency of words in a file:**

from collections import Counter

with open('sample.txt', 'r') as file:

words = file.read().split()

word\_count = Counter(words)

print(word\_count)

**81) Write a list to a file:**

lines = ['Python\n', 'is\n', 'awesome!\n']

with open('list\_output.txt', 'w') as file:

file.writelines(lines)

**82) Copy contents of a file to another file:**

with open('sample.txt', 'r') as src:

with open('copy.txt', 'w') as dst:

dst.write(src.read())

**83) Exception Handling & Error:**

* **Exception handling**: Process of catching and managing runtime errors.
* **Syntax**:

try:

# risky code

except ExceptionType:

# handling code

finally:

# always executes

* **Error**: A problem in a program that causes it to crash. Errors can be syntax or runtime errors.

**84) How many except statements can a try-except block have?**

* As many as needed.
* Example:

try:

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

except ValueError:

print("That's not a number!")

except ZeroDivisionError:

print("Cannot divide by zero!")

**Some built-in exceptions:**

* ValueError, TypeError, ZeroDivisionError, IndexError, KeyError, IOError

**85) When is else part executed in try-except-else?**

* When **no exception** is raised in the try block.

**86) Can one block of except handle multiple exceptions?**

Yes, like this:

try:

# some code

except (ValueError, TypeError):

print("Handled both ValueError and TypeError")

**87) When is finally block executed?**

* Always executed **after try and except**, whether an exception occurred or not.

**88) What happens when '1' == 1 is executed?**

* It returns **False** because:
  + '1' is a **string**
  + 1 is an **integer**
  + Different data types cannot be equal without conversion.

### ****89) How Do We Handle Exceptions with Try/Except/Finally in Python?****

Python uses the try/except/finally block to handle exceptions. Here's the explanation and an example:

* **try block**: Code that may cause an error.
* **except block**: Code that runs if an error occurs.
* **finally block**: Code that always runs, no matter what.

#### Example:

try:

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

result = 10 / num

print("Result is:", result)

except ZeroDivisionError:

print("We cannot divide by zero!")

except ValueError:

print("Please enter a valid number!")

finally:

print("This block always runs (cleanup or exit message)")

### ****90) Write Python program that allows user to enter only odd numbers, else raises an exception****

#### Example:

try:

number = int(input("Enter an odd number: "))

if number % 2 == 0:

raise ValueError("This is not an odd number!")

print("Thank we! We entered an odd number.")

except ValueError as ve:

print("Error:", ve)