1. Write a Python program that concatenates two strings, appends one string to another,and multiplies a string by an integer..

```
# Function to concatenate two strings
def concatenate_strings(str1, str2):
    return str1 + str2

# Function to append one string to another (same as concatenation)
def append_string(str1, str2):
    return str1 + str2

# Function to multiply a string by an integer
def multiply_string(string, times):
    return string * times

# Test the functions
if __name__ == "__main__":
    # Example strings
    string1 = "Hello, "
    string2 = "World!"

    # Concatenate strings
    concatenated_result = concatenate_strings(string1, string2)
    print(f"Concatenated String: {concatenated_result}")

    # Append string
    appended_result = append_string(string1, string2)
    print(f"Appended String: {appended_result}")

    # Multiply string by an integer
    multiplied_result = multiply_string(string2, 3)
    print(f"Multiplied String: {multiplied_result}")
```

```
Concatenated String: Hello, World!
Appended String: Hello, World!
Multiplied String: World!World!World!
```

2. Write a Python program that demonstrates slicing on a string. Extract and print a substring from the given string: "Hello, World!"..

```
# Function to demonstrate string slicing
def slice_string(string, start, end):
    # Extract substring using slicing
    return string[start:end]

# Test the function
if __name__ == "__main__":
    # Given string
    original_string = "Hello, World!"

    # Extract a substring using slicing
    # For example, extract "World" from the string
    start_index = 7
    end_index = 12  # 'World' ends at index 11, but slicing is non-
inclusive for the end
    sliced_string = slice_string(original_string, start_index,
end_index)

    print(f"Original String: {original_string}")
    print(f"Sliced Substring: {sliced_string}")
```

```
Original String: Hello, World!
Sliced Substring: World
```

3. Write a Python program that attempts to change a character in a string and explain why it results in an error. Show how to create a new string with the desired change..

```
# Function to demonstrate immutability of strings
def attempt_string_modification():
    original_string = "Hello, World!"

    try:
        # Attempt to change a character in the string
        original_string[7] = 'w'
    except TypeError as e:
        # Catch the TypeError and display it
        print(f"Error: {e}")

# Function to create a new string with a character changed
def modify_string(original_string, index, new_char):
    # Create a new string by slicing and concatenating the parts
    new_string = original_string[:index] + new_char +
original_string[index+1:]
    return new_string

# Test the functions
if __name__ == "__main__":
    # Demonstrate immutability
    attempt_string_modification()

    # Given string
    original_string = "Hello, World!"
    # Create a new string with 'w' replacing 'W' at index 7
    modified_string = modify_string(original_string, 7, 'w')

    print(f"Original String: {original_string}")
    print(f"Modified String: {modified_string}")
```

```
Error: 'str' object does not support item assignment
Original String: Hello, World!
Modified String: Hello, world!
```

4. Write a Python program that uses the string formatting operator (%) to format and print a string with a name and age..

```
# Function to format name and age using the % operator
def format_name_age(name, age):
    # Using the % operator to format the string
    formatted_string = "My name is %s and I am %d years old." % (name,
age)
    return formatted_string

# Test the function
if __name__ == "__main__":
    # Example name and age
    name = "Alice"
    age = 25

    # Format and print the string
    result = format_name_age(name, age)
    print(result)
```

```
My name is Alice and I am 25 years old.
```

5. Write a Python program that demonstrates the use of at least five built-in string methods such as lower(), upper(), strip(), replace(), and split()..

```python
# Sample string
text = "   Hello, World! Welcome to the world of Python.   "

# 1. lower() - Convert the string to lowercase
lowercase_text = text.lower()
print("Lowercase version:", lowercase_text)

# 2. upper() - Convert the string to uppercase
uppercase_text = text.upper()
print("Uppercase version:", uppercase_text)

# 3. strip() - Remove leading and trailing whitespace
stripped_text = text.strip()
print("Stripped version (no leading/trailing spaces):", stripped_text)

# 4. replace() - Replace a substring with another substring
replaced_text = text.replace("world", "universe")
print("Replaced 'world' with 'universe':", replaced_text)

# 5. split() - Split the string into a list of words
split_text = stripped_text.split()
print("Split into words:", split_text)
```

```
Lowercase version:    hello, world! welcome to the world of python.
Uppercase version:    HELLO, WORLD! WELCOME TO THE WORLD OF PYTHON.
Stripped version (no leading/trailing spaces): Hello, World! Welcome
to the world of Python.
Replaced 'world' with 'universe':    Hello, World! Welcome to the
universe of Python.
Split into words: ['Hello,', 'World!', 'Welcome', 'to', 'the',
'world', 'of', 'Python.']
```

6. Write a Python program that uses slice operation to reverse a string.

```
# Sample string
text = "Python Programming"

# Using slice operation to reverse the string
reversed_text = text[::-1]

# Output the reversed string
print("Original string:", text)
print("Reversed string:", reversed_text)
```

```
Original string: Python Programming
Reversed string: gnimmargorP nohtyP
```

7. Write a Python program that takes a character as input and prints its ASCII value using ord(). Then take an ASCII value as input and print the corresponding character using chr().

).

```
# Taking a character as input
char_input = input("Enter a character: ")

# Using ord() to get the ASCII value of the character
ascii_value = ord(char_input)
print(f"The ASCII value of '{char_input}' is: {ascii_value}")

# Taking an ASCII value as input
ascii_input = int(input("Enter an ASCII value: "))

# Using chr() to get the corresponding character from the ASCII value
char_from_ascii = chr(ascii_input)
print(f"The character corresponding to ASCII value {ascii_input} is: 
'{char_from_ascii}'")
```

```
Enter a character:  r

The ASCII value of 'r' is: 114
```

```
Enter an ASCII value:  115

The character corresponding to ASCII value 115 is: 's'
```

8. Write a Python program that checks if a substring is present in a given string using the in operator and checks if a substring is not present using the not in operator..

```python
# Input string
text = input("Enter a string: ")

# Input substring to check for presence
substring = input("Enter a substring to search for: ")

# Check if the substring is present in the string using 'in' operator
if substring in text:
    print(f"The substring '{substring}' is present in the string.")
else:
    print(f"The substring '{substring}' is not present in the 
string.")

# Check if the substring is not present in the string using 'not in' 
operator
if substring not in text:
    print(f"The substring '{substring}' is not present in the 
string.")
else:
    print(f"The substring '{substring}' is present in the string.")
```

```
Enter a string:  Hello, I am Scara Robot!
Enter a substring to search for:  Scara

The substring 'Scara ' is present in the string.
The substring 'Scara ' is present in the string.
```

9. Write a Python program that compares two strings and prints whether they are equal, or if one is greater than the other..

```
# Input two strings
string1 = input("Enter the first string: ")
string2 = input("Enter the second string: ")

# Compare the two strings
if string1 == string2:
    print("The two strings are equal.")
elif string1 > string2:
    print(f"'{string1}' is greater than '{string2}'.")
else:
    print(f"'{string1}' is less than '{string2}'.")
```

```
Enter the first string:  Solapur
Enter the second string:  Nashik

'Solapur' is greater than 'Nashik'.
```

10. Write a Python program that iterates over each character in a string and prints it in separate line..

```
# Input string from the user
text = input("Enter a string: ")

# Iterate over each character in the string
for char in text:
    print(char)
```

```
Enter a string:  Solapur

S
o
l
a
p
u
r
```

11. Write a Python program that uses the string module to print all the ASCII letters and digits..

```python
import string

# Print all ASCII letters (uppercase and lowercase)
ascii_letters = string.ascii_letters
print("ASCII Letters:")
print(ascii_letters)

# Print all ASCII digits
ascii_digits = string.digits
print("\nASCII Digits:")
print(ascii_digits)
```

```
ASCII Letters:
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

ASCII Digits:
0123456789
```

12. Write a Python program that creates a dictionary with three key-value pairs. Access and print the value of a specific key..

```python
# Create a dictionary with three key-value pairs
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}

# Specify the key to access
key_to_access = "age"

# Access and print the value of the specified key
if key_to_access in my_dict:
    print(f"The value of '{key_to_access}' is:
{my_dict[key_to_access]}")
else:
    print(f"Key '{key_to_access}' not found in the dictionary.")
```

```
The value of 'age' is: 30
```

13. Write a Python program that adds a new key-value pair to the dictionary created in the previous task. Update an existing value and print the updated dictionary..

```python
# Create a dictionary with three key-value pairs
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}

# Print the original dictionary
print("Original Dictionary:")
print(my_dict)

# Add a new key-value pair
my_dict["occupation"] = "Engineer"

# Update an existing value
my_dict["age"] = 31

# Print the updated dictionary
print("\nUpdated Dictionary:")
print(my_dict)
```

```
Original Dictionary:
{'name': 'Alice', 'age': 30, 'city': 'New York'}

Updated Dictionary:
{'name': 'Alice', 'age': 31, 'city': 'New York', 'occupation':
'Engineer'}
```

14. Write a Python program that removes a key-value pair from a dictionary using the del statement. Print the dictionary before and after removing the key-value pair..

```python
# Create a dictionary with three key-value pairs
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}

# Print the original dictionary
print("Original Dictionary:")
print(my_dict)

# Key to remove
key_to_remove = "age"

# Remove the key-value pair using del statement
if key_to_remove in my_dict:
    del my_dict[key_to_remove]
    print(f"\nRemoved the key-value pair for '{key_to_remove}'.")
else:
    print(f"\nKey '{key_to_remove}' not found in the dictionary.")

# Print the updated dictionary
print("\nUpdated Dictionary:")
print(my_dict)
```

```
Original Dictionary:
{'name': 'Alice', 'age': 30, 'city': 'New York'}

Removed the key-value pair for 'age'.

Updated Dictionary:
{'name': 'Alice', 'city': 'New York'}
```

15. Write a Python program that removes a key-value pair from a dictionary using the pop() method and prints the removed value and the updated dictionary..

```python
# Create a dictionary with three key-value pairs
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}

# Print the original dictionary
print("Original Dictionary:")
print(my_dict)
```

```python
# Key to remove
key_to_remove = "age"

# Remove the key-value pair using pop() method
removed_value = my_dict.pop(key_to_remove, None)

# Check if the key was present and print the removed value
if removed_value is not None:
    print(f"\nRemoved value for key '{key_to_remove}':
{removed_value}")
else:
    print(f"\nKey '{key_to_remove}' not found in the dictionary.")

# Print the updated dictionary
print("\nUpdated Dictionary:")
print(my_dict)
```

```
Original Dictionary:
{'name': 'Alice', 'age': 30, 'city': 'New York'}

Removed value for key 'age': 30

Updated Dictionary:
{'name': 'Alice', 'city': 'New York'}
```

16. Write a Python program that iterates over a dictionary and prints each key and its corresponding value..

```python
# Create a dictionary with some key-value pairs
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York",
    "occupation": "Engineer"
}

# Iterate over the dictionary and print each key-value pair
print("Key-Value Pairs in the Dictionary:")
for key, value in my_dict.items():
    print(f"{key}: {value}")
```

```
Key-Value Pairs in the Dictionary:
name: Alice
age: 30
```

```
city: New York
occupation: Engineer
```

17. Write a Python program that uses dictionary comprehension to create a dictionary with keys as numbers from 1 to 5 and values as their squares..

```
# Using dictionary comprehension to create a dictionary of squares
squares_dict = {num: num**2 for num in range(1, 6)}

# Print the resulting dictionary
print("Dictionary of squares from 1 to 5:")
print(squares_dict)
```

```
Dictionary of squares from 1 to 5:
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

18. Write a Python program that merges two dictionaries into one and prints the resulting dictionary. Use the update() method and the unpacking operator (**)..

by using () method

```
# Create two dictionaries
dict1 = {
    "name": "Alice",
    "age": 30
}

dict2 = {
    "city": "New York",
    "occupation": "Engineer"
}

# Merging using update() method
merged_dict = dict1.copy()  # Create a copy of dict1 to avoid
```

```
modifying it
merged_dict.update(dict2)    # Update the copy with dict2

# Print the resulting dictionary
print("Merged Dictionary using update():")
print(merged_dict)

Merged Dictionary using update():
{'name': 'Alice', 'age': 30, 'city': 'New York', 'occupation':
'Engineer'}
```

by using (**) operator

```
# Merging using the unpacking operator
merged_dict_unpacking = {**dict1, **dict2}

# Print the resulting dictionary
print("\nMerged Dictionary using unpacking operator (**):")
print(merged_dict_unpacking)


Merged Dictionary using unpacking operator (**):
{'name': 'Alice', 'age': 30, 'city': 'New York', 'occupation':
'Engineer'}
```

19. 19. Write a Python program that checks if a specific key is present in a dictionary and prints an appropriate message..

```
# Create a sample dictionary
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}

# Specify the key to check
key_to_check = input("Enter the key you want to check: ")

# Check if the specified key is present in the dictionary
if key_to_check in my_dict:
    print(f"The key '{key_to_check}' is present in the dictionary.")
else:
    print(f"The key '{key_to_check}' is not present in the
dictionary.")
```

```
Enter the key you want to check:  age

The key 'age' is present in the dictionary.
```

20. Write a Python program that prints all the keys and all the values of a dictionary using the keys() and values() methods..

```
# Create a sample dictionary
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York",
    "occupation": "Engineer"
}

# Print all keys using keys() method
print("Keys in the dictionary:")
for key in my_dict.keys():
    print(key)

# Print all values using values() method
print("\nValues in the dictionary:")
for value in my_dict.values():
    print(value)
```

```
Keys in the dictionary:
name
age
city
occupation

Values in the dictionary:
Alice
30
New York
Engineer
```

21. Write a Python program that uses the setdefault() method to get the value of a key if it is present, or insert the key with a specified value if it is not present. Print the dictionary before and after using setdefault().

).

```python
# Create a sample dictionary
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}

# Print the original dictionary
print("Original Dictionary:")
print(my_dict)

# Key to check and value to set if not present
key_to_check = "occupation"
value_to_set = "Engineer"

# Use setdefault() to get the value or insert the key with the
specified value
result_value = my_dict.setdefault(key_to_check, value_to_set)

# Print the result of setdefault()
print(f"\nValue for key '{key_to_check}': {result_value}")

# Print the updated dictionary
print("\nUpdated Dictionary after using setdefault():")
print(my_dict)
```

```
Original Dictionary:
{'name': 'Alice', 'age': 30, 'city': 'New York'}

Value for key 'occupation': Engineer

Updated Dictionary after using setdefault():
{'name': 'Alice', 'age': 30, 'city': 'New York', 'occupation':
'Engineer'}
```

22.Write a Python program that creates a nested dictionary representing a collection of students with their names and corresponding marks in different subjects. Print the nested dictionary.

y.

```python
# Create a nested dictionary representing students and their marks
students = {
    "Om": {
        "Math": 85,
        "Science": 90,
        "English": 78
    },
    "Sai": {
        "Math": 92,
        "Science": 88,
        "English": 82
    },
    "Ram": {
        "Math": 70,
        "Science": 75,
        "English": 80
    },
    "Chetan": {
        "Math": 95,
        "Science": 92,
        "English": 89
    }
}

# Print the nested dictionary
print("Students and their marks:")
print(students)
```

```
Students and their marks:
{'Om': {'Math': 85, 'Science': 90, 'English': 78}, 'Sai': {'Math': 92,
'Science': 88, 'English': 82}, 'Ram': {'Math': 70, 'Science': 75,
'English': 80}, 'Chetan': {'Math': 95, 'Science': 92, 'English': 89}}
```

23. Write a Python program that sorts a dictionary by its keys and prints the sorted dictionary..

```python
# Create a sample dictionary
my_dict = {
    "banana": 3,
    "apple": 5,
    "orange": 2,
    "grape": 4
}

# Print the original dictionary
print("Original Dictionary:")
print(my_dict)

# Sort the dictionary by keys
sorted_dict = dict(sorted(my_dict.items()))

# Print the sorted dictionary
print("\nSorted Dictionary by Keys:")
print(sorted_dict)
```

```
Original Dictionary:
{'banana': 3, 'apple': 5, 'orange': 2, 'grape': 4}

Sorted Dictionary by Keys:
{'apple': 5, 'banana': 3, 'grape': 4, 'orange': 2}
```

24. Write a Python program that uses the defaultdict from the collections module to create a dictionary with default values. Demonstrate adding and accessing elements..

```python
from collections import defaultdict

# Create a defaultdict with default value type as int
default_dict = defaultdict(int)

# Add elements to the defaultdict
default_dict['apple'] += 1  # Adding 1 to the count of apples
default_dict['banana'] += 2  # Adding 2 to the count of bananas
default_dict['orange'] += 3  # Adding 3 to the count of oranges
```

```python
# Print the defaultdict
print("Defaultdict after adding elements:")
print(default_dict)

# Accessing elements
apple_count = default_dict['apple']
banana_count = default_dict['banana']
cherry_count = default_dict['cherry']  # This key doesn't exist yet

# Print the counts
print("\nAccessing elements:")
print(f"Count of apples: {apple_count}")
print(f"Count of bananas: {banana_count}")
print(f"Count of cherries (default): {cherry_count}")  # This will
show the default value for int, which is 0
```

```
Defaultdict after adding elements:
defaultdict(<class 'int'>, {'apple': 1, 'banana': 2, 'orange': 3})

Accessing elements:
Count of apples: 1
Count of bananas: 2
Count of cherries (default): 0
```