LAB 1 (08-08-2024) (BTECH/10029/22)

QUESTION 1

Perform the following operations using python: a) updating an existing string. b) using built-in string methods to manipulate strings ex: len(), find(), decode(), isalpha(), isdigit(), count(), etc.

In [5]:
```
# a) updating an existing string

original_string = "Hello, World!"

# Change "World" to "Python"
updated_string = original_string.replace("World", "Python")

print("Original String:", original_string)
print("Updated String:", updated_string)
```

```
Original String: Hello, World!
Updated String: Hello, Python!
```

In [4]:
```
# b) using built-in string methods to manipulate strings ex: len(), find(), decode(

sample_string = "Hello, Python 123!"

# len()
string_length = len(sample_string)
print("Length of the string:", string_length)

# find()
position = sample_string.find("Python")
print("Position of 'Python':", position)

# isalpha()
is_alpha = sample_string.isalpha()
print("Is the string alphabetic?", is_alpha)

# isdigit()
is_digit = sample_string.isdigit()
print("Is the string numeric?", is_digit)

# count()
count_o = sample_string.count('o')
print("Count of 'o':", count_o)

# decode()
encoded_string = b'Hello, World!'
decoded_string = encoded_string.decode('utf-8')
print("Decoded String:", decoded_string)

# Check if all characters are alphabetic
all_alpha = all(char.isalpha() or char.isspace() for char in sample_string)
print("Are all characters alphabetic (with spaces)?", all_alpha)
```

```
Length of the string: 18
Position of 'Python': 7
Is the string alphabetic? False
Is the string numeric? False
Count of 'o': 2
Decoded String: Hello, World!
Are all characters alphabetic (with spaces)? False
```

QUESTION 2

Perform the following on 'Lists' using Python a) Updating, slicing, indexing. b) using following built-in methods to manipulate lists: append(), extend(), insert(), pop(), remove(), reverse(), sort.

In [9]:
```python
# a) Updating, slicing, indexing

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

# Updating a list element
my_list[2] = 100
print("Updated List:", my_list)

# Slicing a list
sliced_list = my_list[1:4]
print("Sliced List:", sliced_list)

# Indexing a list
first_element = my_list[0]
last_element = my_list[-1]
print("First Element:", first_element)
print("Last Element:", last_element)
```

```
Updated List: [10, 20, 100, 40, 50]
Sliced List: [20, 100, 40]
First Element: 10
Last Element: 50
```

In [10]:
```python
# b) using following built-in methods to manipulate lists: append(), extend(), inse

# 1. append()
my_list.append(60)
print("List after append:", my_list)

# 2. extend()
my_list.extend([70, 80])
print("List after extend:", my_list)

# 3. insert()
my_list.insert(1, 15)
print("List after insert:", my_list)

# 4. pop()
popped_element = my_list.pop()
print("Popped Element:", popped_element)
print("List after pop:", my_list)

# 5. remove()
my_list.remove(100)
print("List after remove(100):", my_list)

# 6. reverse()
my_list.reverse()
```

```
print("List after reverse:", my_list)

# 7. sort()
my_list.sort()
print("List after sort:", my_list)
```

```
List after append: [10, 20, 100, 40, 50, 60]
List after extend: [10, 20, 100, 40, 50, 60, 70, 80]
List after insert: [10, 15, 20, 100, 40, 50, 60, 70, 80]
Popped Element: 80
List after pop: [10, 15, 20, 100, 40, 50, 60, 70]
List after remove(100): [10, 15, 20, 40, 50, 60, 70]
List after reverse: [70, 60, 50, 40, 20, 15, 10]
List after sort: [10, 15, 20, 40, 50, 60, 70]
```

QUESTION 3

Perform the following on 'tuples' using python a) updating, indexing, deleting, slicing b) using following built-in methods to manipulate typles: max(), min(), len(), tuple().

In [12]:
```
# a) updating, indexing, deleting, slicing

my_tuple = (10, 20, 30, 40, 50)
print(my_tuple[0])
print(my_tuple[1])
print(my_tuple[-1])

# Slicing
sliced_tuple = my_tuple[1:4]
print(sliced_tuple)

# Updating
my_tuple = (10, 20, 30, 40, 50)
new_tuple = my_tuple[:2] + (99,) + my_tuple[3:]
print(new_tuple)

# Deleting
my_tuple = (10, 20, 30, 40, 50)
# Delete the element 30 by creating a new tuple
new_tuple = my_tuple[:2] + my_tuple[3:]
print(new_tuple)
```

```
10
20
50
(20, 30, 40)
(10, 20, 99, 40, 50)
(10, 20, 40, 50)
```

In [13]:
```
# b) using following built-in methods to manipulate typles: max(), min(), len(), tu

# max()
my_tuple = (10, 20, 30, 5, 40, 15)
max_value = max(my_tuple)
print(max_value)

# min()
min_value = min(my_tuple)
print(min_value)

# len()
length = len(my_tuple)
print(length)
```

```
# tuple()
my_list = [1, 2, 3, 4]
new_tuple = tuple(my_list)
print(new_tuple)
```

```
40
5
6
(1, 2, 3, 4)
```

QUESTION 4

Perform the following on 'dictionary' using python a) updating, deleting. b) using following built-in methods to manipulate dictionary: update(), valyes(), get(), clear (), copy(), type(), len().

In [14]:
```
# a) updating, deleting.

my_dict = {
    'name': 'Alice',
    'age': 30,
    'city': 'New York'
}

print("Original Dictionary:", my_dict)

# Updating
my_dict.update({'age': 31})
my_dict.update({'country': 'USA'})
print("Updated Dictionary:", my_dict)

# Deleting
del my_dict['city']
print("Dictionary after deletion:", my_dict)
my_dict.clear()
print("Dictionary after clearing:", my_dict)
```

```
Original Dictionary: {'name': 'Alice', 'age': 30, 'city': 'New York'}
Updated Dictionary: {'name': 'Alice', 'age': 31, 'city': 'New York', 'country': 'U
SA'}
Dictionary after deletion: {'name': 'Alice', 'age': 31, 'country': 'USA'}
Dictionary after clearing: {}
```

In [15]:
```
# b) using following built-in methods to manipulate dictionary: update(), valyes(),

my_dict = {
    'name': 'Alice',
    'age': 30,
    'city': 'New York'
}

# Getting the values of the dictionary
values = my_dict.values()
print("Values in Dictionary:", values)

# Getting a specific value using get() method
name = my_dict.get('name')
print("Name from Dictionary using get():", name)

# Making a copy of the dictionary
copy_dict = my_dict.copy()
```

```
print("Copied Dictionary:", copy_dict)

# Checking the type of dictionary
dict_type = type(my_dict)
print("Type of my_dict:", dict_type)

# Checking the number of items in the dictionary
dict_length = len(my_dict)
print("Number of items in my_dict:", dict_length)
```

```
Values in Dictionary: dict_values(['Alice', 30, 'New York'])
Name from Dictionary using get(): Alice
Copied Dictionary: {'name': 'Alice', 'age': 30, 'city': 'New York'}
Type of my_dict: <class 'dict'>
Number of items in my_dict: 3
```

QUESTION 5

Create seperate sets for indian cricket players playing T20, ODI and Test Match for current West Indies tour: Also perform the union(), intersection() , difference() operations on the above sets.

In [16]:

```
# Define sets for Indian cricket players for each format
t20_players = {'Rohit Sharma', 'Virat Kohli', 'Jasprit Bumrah', 'Rishabh Pant', 'Ha
odi_players = {'Rohit Sharma', 'Virat Kohli', 'Shikhar Dhawan', 'Jasprit Bumrah', '
test_players = {'Virat Kohli', 'Rohit Sharma', 'Jasprit Bumrah', 'Rishabh Pant', 'C

# Perform union operation
all_players = t20_players.union(odi_players).union(test_players)

# Perform intersection operation
common_players = t20_players.intersection(odi_players).intersection(test_players)

# Perform difference operation
t20_not_in_odi = t20_players.difference(odi_players)
odi_not_in_test = odi_players.difference(test_players)
test_not_in_t20 = test_players.difference(t20_players)

print("T20 Players:", t20_players)
print("ODI Players:", odi_players)
print("Test Players:", test_players)
print("Union of all players:", all_players)
print("Common players in all formats:", common_players)
print("T20 Players not in ODI:", t20_not_in_odi)
print("ODI Players not in Test:", odi_not_in_test)
print("Test Players not in T20:", test_not_in_t20)
```

```
T20 Players: {'Rohit Sharma', 'Rishabh Pant', 'Hardik Pandya', 'Jasprit Bumrah',
'Virat Kohli'}
ODI Players: {'Shikhar Dhawan', 'Rohit Sharma', 'Rishabh Pant', 'Jasprit Bumrah',
'Virat Kohli'}
Test Players: {'Cheteshwar Pujara', 'Rohit Sharma', 'Rishabh Pant', 'Jasprit Bumra
h', 'Virat Kohli'}
Union of all players: {'Shikhar Dhawan', 'Hardik Pandya', 'Virat Kohli', 'Cheteshw
ar Pujara', 'Rohit Sharma', 'Rishabh Pant', 'Jasprit Bumrah'}
Common players in all formats: {'Jasprit Bumrah', 'Rohit Sharma', 'Rishabh Pant',
'Virat Kohli'}
T20 Players not in ODI: {'Hardik Pandya'}
ODI Players not in Test: {'Shikhar Dhawan'}
Test Players not in T20: {'Cheteshwar Pujara'}
```

QUESTION 6

Find the largest of the 4 strings using conditional statements (if-elif-else) using python.

In [17]:
```
def find_largest_string(str1, str2, str3, str4):
    # Initialize the largest string as the first string
    largest = str1

    # Compare the second string with the current largest
    if str2 > largest:
        largest = str2

    # Compare the third string with the current largest
    if str3 > largest:
        largest = str3

    # Compare the fourth string with the current largest
    if str4 > largest:
        largest = str4

    return largest

string1 = "apple"
string2 = "banana"
string3 = "grape"
string4 = "pear"

largest_string = find_largest_string(string1, string2, string3, string4)
print("The largest string is:", largest_string)
```

```
The largest string is: pear
```

QUESTION 7

Write a function to generate even numbers between 1 to 30. create a list squaring these numbers and display the list as well. create another list by filtering the squared list using 'anonymous' function to get those numbers which are even numbers (Hint: Use filter() method and 'Lambda' keyword).

In [18]:
```
def generate_even_squares():
    # Generate even numbers between 1 and 30
    even_numbers = [num for num in range(1, 31) if num % 2 == 0]

    # Create a list of squares of the even numbers
    squared_numbers = [num ** 2 for num in even_numbers]

    # Filter the squared numbers to keep only the even squares
    even_squares = list(filter(lambda x: x % 2 == 0, squared_numbers))

    # Display the lists
    print("Even Numbers:", even_numbers)
    print("Squared Numbers:", squared_numbers)
    print("Even Squares:", even_squares)

# Call the function
generate_even_squares()
```

```
Even Numbers: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30]
Squared Numbers: [4, 16, 36, 64, 100, 144, 196, 256, 324, 400, 484, 576, 676, 784,
900]
Even Squares: [4, 16, 36, 64, 100, 144, 196, 256, 324, 400, 484, 576, 676, 784, 90
0]
```

In [ ]: