Assessment answers

DATE: 14-FEB-2024

1. Write a Python program to calculate the area of a rectangle given its length and width.

A)

PROGRAM

```
# Function to calculate the area of a rectangle
def calculate_rectangle_area(length, width):
    area = length * width
    return area

# Input: Length and Width
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))

# Calculate and display the area
area = calculate_rectangle_area(length, width)
print("The area of the rectangle is:", area)
```

OUTPUT;

```
Enter the length of the rectangle: 5
Enter the width of the rectangle: 4
The area of the rectangle is: 20.0
```

2. Write a program to convert miles to kilometers

A)

PROGRAM:

```
# Function to convert miles to kilometers
def miles_to_kilometers(miles):
    kilometers = miles * 1.60934
    return kilometers
# Taking input for miles from the user
miles = float(input("Enter the distance in miles: "))
# Converting miles to kilometers using the function
kilometers = miles_to_kilometers(miles)
# Displaying the result
print(f"{miles} miles is equal to {kilometers} kilometers")
```

OUTPUT :

```
Enter the distance in miles: 12
12.0 miles is equal to 19.31208 kilometers
```

3. Write a function to check if a given string is a palindrome.

A)

PROGRAM

```
def is_palindrome(s):
    # Remove spaces and convert to lowercase for case-insensitive comparison
    cleaned_string = s.replace(" ", "").lower()

    # Compare the string with its reverse
    return cleaned_string == cleaned_string[::-1]

# Example usage
input_string = input("Enter a string to check if it's a palindrome: ")

if is_palindrome(input_string):
    print("The given string is a palindrome.")
else:
    print("The given string is not a palindrome.")
```

OUTPUT:

```
Enter a string to check if it's a palindrome: 5
The given string is a palindrome.
```

4. Write a Python program to find the second largest element in a list.

A)

PROGRAM

```
def second_largest_element(lst):
    if len(lst) < 2:
        return "List should have at least two elements"

    # Sorting the list in descending order
    sorted_list = sorted(lst, reverse=True)

    # The second element in the sorted list is the second largest
    return sorted_list[1]

# Example usage
input_list = [12, 45, 8, 23, 56, 32]

result = second_largest_element(input_list)

print(f"The second largest element in the list is: {result}")
```

OUTPUT:

```
The second largest element in the list is: 45
```

5) Explain what indentation means in Python.

A)

In Python, indentation is a way of indicating the grouping of statements within a block of code. It is used to define the structure and scope of the code. Instead of using curly braces or keywords like "begin" and "end" as in some other programming languages, Python uses indentation to show which statements belong to a particular block of code.

Indentation is typically done using spaces or tabs at the beginning of lines. Consistent indentation is crucial because it determines the code's structure. Statements with the same level of indentation are considered part of the same block.

For example:

```
if x > 0:

print("x is positive")

print("This statement is also inside the if block")

# Statements without indentation are outside the block

 print("This statement is outside the if block")
```

In the above example, the two **print** statements are inside the **if** block because they are indented under it. The last **print** statement is not indented and is outside the **if** block.

6) Write a program to perform set difference operation.

A)

PROGRAM

```
# Function to perform set difference using the - operator

def set_difference_using_operator(set1, set2):

   return set1 - set2


# Function to perform set difference using the difference() method

def set_difference_using_method(set1, set2):

   return set1.difference(set2)


# Example usage

set_a = {1, 2, 3, 4, 5}

set_b = {3, 4, 5, 6, 7}


# Using the - operator

result_operator = set_difference_using_operator(set_a, set_b)
```

```
print(f"Set difference using operator: {result_operator}")

# Using the difference() method
result_method = set_difference_using_method(set_a, set_b)
print(f"Set difference using method: {result_method}")
```

OUTPUT:

```
Set difference using operator: {1, 2}
Set difference using method: {1, 2}
```

7. Write a Python program to print numbers from 1 to 10 using a while loop.

A)

PROGRAM

```
# Initialize the counter
counter = 1

# Use a while loop to print numbers from 1 to 10
while counter <= 10:
    print(counter)
    counter += 1
```

OUTPUT:

```
1
2
3
4
5
6
7
8
9
10
```

8. Write a program to calculate the factorial of a number using a while loop.

A)

PROGRAM

```
def factorial(n):
  if n == 0:
    return 1
   else:
     return n * factorial(n - 1)
# Example usage:
number = 10
result = factorial(number)
print(f"The factorial of {number} is {result}")
```

OUTPUT:

```
The factorial of 10 is 3628800
```

9. Write a Python program to check if a number is positive, negative, or zero using if-elif-else statement

A)

PROGRAM

```
# Get user input for the number
number = float(input("Enter a number: "))


# Check if the number is positive, negative, or zero
if number > 0:
   print("The number is positive.")
elif number < 0:
   print("The number is negative.")
else:
   print("The number is zero.")
```

OUTPUT:

```
Enter a number: 9
The number is positive.
```

10. Write a program to determine the largest among three numbers using conditional statements.

A)

PROGRAM

```
# Get user input for three numbers
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Determine the largest number using conditional statements
if num1 >= num2 and num1 >= num3:
   largest = num1
elif num2 >= num1 and num2 >= num3:
   largest = num2
else:
   largest = num3

# Display the result
print(f"The largest number among {num1}, {num2}, and {num3} is: {largest}")
```

OUTPUT:

```
Enter the first number: 2
Enter the second number: 65
Enter the third number: 85
The largest number among 2.0, 6.0, and 85.0 is: 85.0
```

11. Write a Python program to create a numpy array filled with ones of given shape.

A)

PROGRAM

```
import numpy as np

# Get user input for the shape of the array
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))

# Create a NumPy array filled with ones of the given shape
ones_array = np.ones((rows, columns))

# Display the resulting array
print(f"NumPy array filled with ones of shape ({rows}, {columns}):\n{ones_array}")
```

OUTPUT:

```
Enter the number of rows: 5
Enter the number of columns: 5
NumPy array filled with ones of shape (5, 5):
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

12. Write a program to create a 2D numpy array initialized with random integers.

A)

PROGRAM

```
import numpy as np
```

```
# Get user input for the dimensions of the array
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))

# Get user input for the range of random integers
low_limit = int(input("Enter the lower limit for random integers: "))
high_limit = int(input("Enter the upper limit for random integers: "))

# Create a 2D NumPy array initialized with random integers
random_array = np.random.randint(low_limit, high_limit + 1, size=(rows, columns))

# Display the resulting array
print(f"2D NumPy array initialized with random integers:\n{random_array}")
```

OUTPUT:

```
Enter the number of rows: 5
Enter the number of columns: 5
Enter the lower limit for random integers: 5
Enter the upper limit for random integers: 6
2D NumPy array initialized with random integers:
[[5 5 5 6 6]
 [5 5 6 5 6]
 [6 6 6 6 6]
 [6 6 5 5 5]
 [5 5 6 6 6]]
```

13. Write a Python program to generate an array of evenly spaced numbers over a specified range using linspace.

A)

PROGRAM

```
import numpy as np
```

```
# Get user input for the start, stop, and number of elements
start = float(input("Enter the start value: "))
stop = float(input("Enter the stop value: "))
num_elements = int(input("Enter the number of elements: "))

# Generate an array of evenly spaced numbers over the specified range using linspace
evenly_spaced_array = np.linspace(start, stop, num_elements)

# Display the resulting array
print(f"Array of evenly spaced numbers over the range ({start}, {stop}) with {num_elements} elements:\n{evenly_spaced_array}")
```

OUTPUT:

```
Enter the start value: 2
Enter the stop value: 9
Enter the number of elements: 9
Array of evenly spaced numbers over the range (2.0, 9.0) with 9 elements:
 [2.    2.875 3.75  4.625 5.5   6.375 7.25  8.125 9.   ]
```

14. Write a program to generate an array of 10 equally spaced values between 1 and 100 using linspace.

A)

PROGRAM

```
import numpy as np

# Generate an array of 10 equally spaced values between 1 and 100 using linspace
equally_spaced_array = np.linspace(1, 100, 10)

# Display the resulting array
print(f"Array of 10 equally spaced values between 1 and 100:\n{equally_spaced_array}")
```

OUTPUT:

```
Array of 10 equally spaced values between 1 and 100:
[  1.  12.  23.  34.  45.  56.  67.  78.  89. 100.]
```

15. Write a Python program to create an array containing even numbers from 2 to 20 using arange.

A)

PROGRAM

```
import numpy as np

# Create an array containing even numbers from 2 to 20 using arange
even_numbers_array = np.arange(2, 21, 2)

# Display the resulting array
print(f"Array containing even numbers from 2 to 20:\n{even_numbers_array}")
```

OUTPUT:

```
Array containing even numbers from 2 to 20:
[ 2  4  6  8 10 12 14 16 18 20]
```

16. Write a program to create an array containing numbers from 1 to 10 with a step size of 0.5

using arange.

A)

PROGRAM

```
import numpy as np
# Create an array containing numbers from 1 to 10 with a step size of 0.5 using arange
array_with_step = np.arange(1, 11, 0.5)

# Display the resulting array
print(f"Array containing numbers from 1 to 10 with a step size of 0.5:\n{array_with_step}")
```

OUTPUT:

```
Array containing numbers from 1 to 10 with a step size of 0.5:
[ 1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5  7.   7.5
  8.   8.5  9.   9.5 10.  10.5]
```