assessment-1

February 23, 2024

[ ]:
```
#Write a Python program to calculate the area of a rectangle given its length␣
 ↪and width

def calculate_rectangle_area(length, width):
    area = length * width
    return area

# Taking input from the user
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))

# Calculating the area
area = calculate_rectangle_area(length, width)

# Displaying the result
print(f"The area of the rectangle with length {length} and width {width} is:␣
 ↪{area}")
```

```
Enter the length of the rectangle: 10
Enter the width of the rectangle: 5
The area of the rectangle with length 10.0 and width 5.0 is: 50.0
```

[ ]:
```
# Write a program to convert miles to kilometers

def miles_to_kilometers(miles):
    kilometers = miles * 1.60934
    return kilometers

# Taking input from the user
miles = float(input("Enter the distance in miles: "))

# Converting miles to kilometers
kilometers = miles_to_kilometers(miles)

# Displaying the result
print(f"{miles} miles is equal to {kilometers} kilometers")
```

```
Enter the distance in miles: 100
```

```
100.0 miles is equal to 160.934 kilometers
```

[ ]:
```python
# Write a function to check if a given string is a palindrome.

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

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

# Example usage:
input_string = input("Enter a string: ")
if is_palindrome(input_string):
    print(f"{input_string} is a palindrome.")
else:
    print(f"{input_string} is not a palindrome.")
```

```
Enter a string: RADAR
RADAR is a palindrome.
```

[ ]:
```python
# Write a Python program to find the second largest element in a list.

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

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

    # Finding the second largest element
    second_largest = sorted_list[1]

    return second_largest

# Example usage:
try:
    input_list = [int(x) for x in input("Enter elements of the list separated␣
 ↪by space: ").split()]
    result = second_largest_element(input_list)
    print(f"The second largest element in the list is: {result}")
except ValueError:
    print("Invalid input. Please enter a list of integers.")
```

```
Enter elements of the list separated by space: 3 5 2 4 5
The second largest element in the list is: 5
```

[ ]:
```
# Explain what indentation means in Python

# Indentation is used to define the structure and scope of the code.
# Unlike many other programming languages that use braces or keywords to␣
 ↪indicate the beginning and end of blocks of code (such as if statements,␣
 ↪loops, and functions), Python uses indentation.
# Indentation refers to the spaces or tabs at the beginning of a line of code.
# It is used to group statements into blocks. Blocks of code with the same␣
 ↪level of indentation are considered part of the same block or scope.
# The standard convention in Python is to use four spaces for each level of␣
 ↪indentation.
```

[ ]:
```
# Write a program to perform set difference operation.

def set_difference(set1, set2):
    # Using the - operator for set difference
    difference_result_operator = set1 - set2

    # Using the difference() method for set difference
    difference_result_method = set1.difference(set2)

    return difference_result_operator, difference_result_method

# Example usage:
set_a = {1, 2, 3, 4, 5}
set_b = {3, 4, 5, 6, 7}

result_operator, result_method = set_difference(set_a, set_b)

print(f"Set Difference using - operator: {result_operator}")
print(f"Set Difference using difference() method: {result_method}")
```

```
Set Difference using - operator: {1, 2}
Set Difference using difference() method: {1, 2}
```

[ ]:
```
# Write a Python program to print numbers from 1 to 10 using a while loop.

# Initialize a variable to start from 1
number = 1

# Use a while loop to print numbers from 1 to 10
while number <= 10:
    print(number)
    number += 1  # Increment the number in each iteration

# The loop will exit when number becomes 11
```

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

[ ]:
```python
# Write a program to calculate the factorial of a number using a while loop.

def calculate_factorial(number):
    if number < 0:
        return "Factorial is undefined for negative numbers."
    elif number == 0 or number == 1:
        return 1
    else:
        factorial = 1
        while number > 1:
            factorial *= number
            number -= 1
        return factorial

# Example usage:
try:
    user_input = int(input("Enter a number to calculate its factorial: "))
    result = calculate_factorial(user_input)
    print(f"The factorial of {user_input} is: {result}")
except ValueError:
    print("Invalid input. Please enter a valid integer.")
```

```
Enter a number to calculate its factorial: 5
The factorial of 5 is: 120
```

[ ]:
```python
# Write a Python program to check if a number is positive, negative, or zero␣
↪using if-elif-else statements.

# Taking input from the user
try:
    number = float(input("Enter a number: "))

    # Checking if the number is positive, negative, or zero
    if number > 0:
        print(f"{number} is a positive number.")
    elif number < 0:
```

```
        print(f"{number} is a negative number.")
    else:
        print("The entered number is zero.")
except ValueError:
    print("Invalid input. Please enter a valid number.")
```

```
Enter a number: 6
6.0 is a positive number.
```

[ ]:
```
# Write a program to determine the largest among three numbers using␣
 ↪conditional statements.

# Taking input from the user
try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))
    num3 = float(input("Enter the third number: "))

    # Using conditional statements to determine the largest number
    if num1 >= num2 and num1 >= num3:
        largest = num1
    elif num2 >= num1 and num2 >= num3:
        largest = num2
    else:
        largest = num3

    print(f"The largest number among {num1}, {num2}, and {num3} is: {largest}")
except ValueError:
    print("Invalid input. Please enter valid numbers.")
```

```
Enter the first number: 3
Enter the second number: 10
Enter the third number: 6
The largest number among 3.0, 10.0, and 6.0 is: 10.0
```

[ ]:
```
# Write a Python program to create a numpy array filled with ones of given shape

import numpy as np

def create_ones_array(shape):
    try:
        shape = tuple(map(int, shape))
        ones_array = np.ones(shape)
        return ones_array
    except ValueError:
        return "Invalid shape. Please enter valid dimensions as integers."
```

```
# Example usage:
shape_input = input("Enter the shape of the array (comma-separated values): ")

result_array = create_ones_array(shape_input)

if isinstance(result_array, np.ndarray):
    print(f"Array of ones with shape {result_array.shape}:\n{result_array}")
else:
    print(result_array)
```

[ ]:
```
# Write a program to create a 2D numpy array initialized with random integers.

import numpy as np

def create_random_int_array(rows, cols, min_value=0, max_value=100):
    try:
        rows, cols = int(rows), int(cols)
        min_value, max_value = int(min_value), int(max_value)
        random_int_array = np.random.randint(min_value, max_value + 1,␣
 ↪size=(rows, cols))
        return random_int_array
    except ValueError:
        return "Invalid input. Please enter valid integer values."

# Example usage:
rows_input = input("Enter the number of rows: ")
cols_input = input("Enter the number of columns: ")
min_value_input = input("Enter the minimum random integer value: ")
max_value_input = input("Enter the maximum random integer value: ")

result_array = create_random_int_array(rows_input, cols_input, min_value_input,␣
 ↪max_value_input)

if isinstance(result_array, np.ndarray):
    print(f"2D Array of random integers:\n{result_array}")
else:
    print(result_array)
```

[ ]:
```
# Write a Python program to generate an array of evenly spaced numbers over a␣
 ↪specified range using linespace

import numpy as np

def create_random_int_array(rows, cols, min_value=0, max_value=100):
    try:
        rows, cols = int(rows), int(cols)
        min_value, max_value = int(min_value), int(max_value)
```

```
        random_int_array = np.random.randint(min_value, max_value + 1,␣
 ↪size=(rows, cols))
        return random_int_array
    except ValueError:
        return "Invalid input. Please enter valid integer values."

# Example usage:
rows_input = input("Enter the number of rows: ")
cols_input = input("Enter the number of columns: ")
min_value_input = input("Enter the minimum random integer value: ")
max_value_input = input("Enter the maximum random integer value: ")

result_array = create_random_int_array(rows_input, cols_input, min_value_input,␣
 ↪max_value_input)

if isinstance(result_array, np.ndarray):
    print(f"2D Array of random integers:\n{result_array}")
else:
    print(result_array)
```

[ ]:
```
# Write a program to generate an array of 10 equally spaced values between 1␣
 ↪and 100 using line space

import numpy as np

# Using linspace to generate the array
result_array = np.linspace(1, 100, 10)

# Displaying the result
print(f"Array of 10 equally spaced values between 1 and 100:\n{result_array}")
```

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

[ ]:
```
# Write a Python program to create an array containing even numbers from 2 to␣
 ↪20 using arrange.

import numpy as np

# Using arange to generate the array
result_array = np.arange(2, 21, 2)

# Displaying the result
print(f"Array containing even numbers from 2 to 20:\n{result_array}")
```

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

[ ]:
```python
# Write a program to create an array containing numbers from 1 to 10 with a␣
 ↪step size of 0.5 using arrange

import numpy as np

# Using arange to generate the array
result_array = np.arange(1, 10.5, 0.5)

# Displaying the result
print(f"Array containing numbers from 1 to 10 with a step size of 0.5:
 ↪\n{result_array}")
```

```
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. ]
```

[ ]:
```python

```