m2_case_study_1

October 10, 2024

1 Module 2 – Sequences and File Operations

1.1 Case Study – 1

1.1.1 1. Write a program that will find factors of the given number and find whether the factor is even or odd.

Hint: Use Loop with if-else statements

[1]:
```python
def find_factors_and_classify(num):
    """
    This function finds all factors of a given number and classifies
    each factor as either even or odd.

    Parameters:
    num (int): The number for which the factors are to be found.

    Returns:
    None: It prints out the factors and their classifications (even or odd).
    """

    print(f"Factors of {num} and their classification:")

    # Loop through all numbers from 1 to the given number
    for i in range(1, num + 1):
        # Check if i is a factor of num
        if num % i == 0:
            # Determine if the factor is even or odd
            if i % 2 == 0:
                print(f"{i} is even")
            else:
                print(f"{i} is odd")


# Input: Asking user to input a number
number = int(input("Enter a number: "))

# Function call
find_factors_and_classify(number)
```

```
Enter a number:  10

Factors of 10 and their classification:
1 is odd
2 is even
5 is odd
10 is even
```

1.1.2 2. Write a code that accepts a sequence of words as input and prints the words in a sequence after sorting them alphabetically.

Hint: In the case of input data being supplied to the question, it should be assumed to be a console input.

[2]:
```python
def sort_words():
    """
    This function accepts a sequence of words as input, sorts them␣
 ↪alphabetically,
    and prints the sorted sequence.

    Parameters:
    None: The function reads input directly from the console.

    Returns:
    None: The sorted words are printed as output.
    """

    # Input: Accepting a sequence of words from the user
    words = input("Enter a sequence of words (separated by spaces): ").split()

    # Sorting the words alphabetically
    words.sort()

    # Printing the sorted words
    print("Sorted sequence of words:")
    print(" ".join(words))

# Function call
sort_words()
```

```
Enter a sequence of words (separated by spaces):  Orange Kiwi Banana Raspberry
Apple Cherry

Sorted sequence of words:
Apple Banana Cherry Kiwi Orange Raspberry
```

1.1.3 3. Write a program, which will find all the numbers between 1000 and 3000 (both included) such that each digit of a number is an even number.

The numbers obtained should be printed in a comma-separated sequence on a single line. Hint: In the case of input data being supplied to the question, it should be assumed to be a console input. Divide each digit by 2 and verify whether is it even or not.

[3]:
```python
def find_even_digit_numbers():
    """
    This function finds all numbers between 1000 and 3000 (both inclusive)
    where each digit of the number is an even number. It prints these numbers
    in a comma-separated sequence on a single line.

    Parameters:
    None: The function operates within the specified range without any input.

    Returns:
    None: It prints the result as a comma-separated sequence.
    """

    even_digit_numbers = []

    # Loop through numbers from 1000 to 3000 (inclusive)
    for num in range(1000, 3001):
        num_str = str(num)  # Convert the number to string for digit-wise␣
 ↪checking
        if all(int(digit) % 2 == 0 for digit in num_str):  # Check if all␣
 ↪digits are even
            even_digit_numbers.append(num_str)  # Append the number if all␣
 ↪digits are even

    # Print the numbers in a comma-separated sequence
    print(",".join(even_digit_numbers))

# Function call
find_even_digit_numbers()

'''
Difference between List Comprehension and Generator Expression:
List comprehension would create an entire list in memory (e.g., [int(digit) % 2␣
 ↪== 0 for digit in num_str]), evaluating each element and storing the results␣
 ↪in a list.
Generator expression is more memory efficient because it generates the values␣
 ↪on the fly, without creating and storing an entire list in memory. It's␣
 ↪evaluated lazily, meaning the values are generated one by one only when␣
 ↪needed.
```

```
This creates a generator that yields one True or False at a time, which is then␣
↪consumed by the all() function.
'''
```

```
2000,2002,2004,2006,2008,2020,2022,2024,2026,2028,2040,2042,2044,2046,2048,2060,
2062,2064,2066,2068,2080,2082,2084,2086,2088,2200,2202,2204,2206,2208,2220,2222,
2224,2226,2228,2240,2242,2244,2246,2248,2260,2262,2264,2266,2268,2280,2282,2284,
2286,2288,2400,2402,2404,2406,2408,2420,2422,2424,2426,2428,2440,2442,2444,2446,
2448,2460,2462,2464,2466,2468,2480,2482,2484,2486,2488,2600,2602,2604,2606,2608,
2620,2622,2624,2626,2628,2640,2642,2644,2646,2648,2660,2662,2664,2666,2668,2680,
2682,2684,2686,2688,2800,2802,2804,2806,2808,2820,2822,2824,2826,2828,2840,2842,
2844,2846,2848,2860,2862,2864,2866,2868,2880,2882,2884,2886,2888
```

1.1.4 4. Write a program that accepts a sentence and calculates the number of letters and digits.

Suppose the entered string is: Python0325 Then the output will be: LETTERS: 6 DIGITS:4 Hint: Use built-in functions of string.

[4]:
```
def count_letters_digits(sentence):
    """
    This function accepts a sentence and calculates the number of letters and␣
↪digits in it.

    Parameters:
    sentence (str): The input sentence to be analyzed.

    Returns:
    None: It prints the count of letters and digits.
    """

    letters = 0
    digits = 0

    # Loop through each character in the sentence
    for char in sentence:
        if char.isalpha():  # Check if the character is a letter
            letters += 1
        elif char.isdigit():  # Check if the character is a digit
            digits += 1

    # Print the results
    print(f"LETTERS: {letters}")
    print(f"DIGITS: {digits}")

# Input: Asking user to input a sentence
sentence = input("Enter a sentence: ")
```

```
# Function call
count_letters_digits(sentence)
```

```
Enter a sentence:  Alpha12345K5

LETTERS: 6
DIGITS: 6
```

1.1.5 5. Design a code that will find whether the given number is a Palindrome number or

not. Hint: Use built-in functions of string

[5]:
```
def is_palindrome(number):
    """
    This function checks whether the given number is a palindrome or not.

    Parameters:
    number (int): The number to be checked.

    Returns:
    None: It prints whether the number is a palindrome or not.
    """

    # Convert the number to a string
    num_str = str(number)

    # Check if the string is equal to its reverse
    if num_str == num_str[::-1]:
        print(f"{number} is a palindrome.")
    else:
        print(f"{number} is not a palindrome.")

# Input: Asking user to input a number
number = int(input("Enter a number: "))

# Function call
is_palindrome(number)
```

```
Enter a number:  1234321

1234321 is a palindrome.
```

[6]:
```
#### Mr Akram M'Tir 10-10-2024
```

[ ]:
```

```