week7_programs

January 15, 2025

[5]:
```python
"""Write and test a function that takes a string as a parameter and returns a␣
↪sorted list
of all the unique letters used in the string. So, if the string is cheese, the␣
↪list
returned should be ['c', 'e', 'h', 's']."""
def sortedunique_letters(word):
    unique_letters = sorted(set(word))
    return unique_letters
def user_input():
    word=input("Enter a random word:")
    return word
def display():
    word=user_input()
    letters=sortedunique_letters(word)
    print(f"The sorted unique letters are {letters}.")
display()
```

```
Enter a random word: ssabina
```

```
The sorted unique letters are ['a', 'b', 'i', 'n', 's'].
```

[8]:
```python
"""Write and test three functions that each take two words (strings) as␣
↪parameters and
return sorted lists (as defined above) representing respectively:
Letters that appear in at least one of the two words.
Letters that appear in both words.
Letters that appear in either word, but not in both.
Hint: These could all be done programmatically, but consider carefully what␣
↪topic we
have been discussing this week! Each function can be exactly one line."""
def union_of_words(word1, word2, word3):
    return sorted(list(set(word1) | set(word2) | set(word3)))

def common_letters(word1, word2, word3):
    return sorted(list(set(word1) & set(word2) & set(word3)))

def different_letters(word1, word2, word3):
    return sorted(list(set(word1) ^ set(word2) ^ set(word3)))
```

```
def user_inputs():
    word1 = input("Enter the first word: ")
    word2 = input("Enter the second word: ")
    word3 = input("Enter the third word: ")
    return word1, word2, word3

def display():
    word1, word2, word3 = user_inputs()
    union_result = union_of_words(word1, word2, word3)
    common_result = common_letters(word1, word2, word3)
    different_result = different_letters(word1, word2, word3)
    print(f"Letters in at least one word: {union_result}")
    print(f"Letters in all three words: {common_result}")
    print(f"Letters in either word but not in all: {different_result}")

display()
```

```
Enter the first word:  apple
Enter the second word:  banana
Enter the third word:  coconut

Letters in at least one word: ['a', 'b', 'c', 'e', 'l', 'n', 'o', 'p', 't', 'u']
Letters in all three words: []
Letters in either word but not in all: ['b', 'c', 'e', 'l', 'o', 'p', 't', 'u']
```

[11]:
```
"""Write a program that manages a list of countries and their capital cities.␣
 ↪It should
prompt the user to enter the name of a country. If the program already "knows"
the name of the capital city, it should display it. Otherwise it should ask the␣
 ↪user to
enter it. This should carry on until the user terminates the program (how this
happens is up to you).
Note: A good solution to this task will be able to cope with the country being␣
 ↪entered
variously as, for example, "Wales", "wales", "WALES" and so on."""
def countries_capital():
    country_capital = {}

    print("Country-Capital Manager")
    print("Type 'exit' to quit the program.\n")

    while True:
        country = input("Enter the name of a country (or 'exit' to stop): ").
 ↪strip()
        if country.lower() == "exit":
            print("\nExiting the program. Thank you!")
            break
        normalized_country = country.lower()
        if normalized_country in country_capital:
            print(f"The capital of {country} is␣
↪{country_capital[normalized_country].title()}.")
        else:
            capital = input(f"I don't know the capital of {country}. Please␣
↪enter it: ").strip()
            country_capital[normalized_country] = capital.lower()
            print(f"Thank you! I've recorded that the capital of {country} is␣
↪{capital}.\n")

if __name__ == "__main__":
    countries_capital()
```

```
Country-Capital Manager
Type 'exit' to quit the program.


Enter the name of a country (or 'exit' to stop):  Nepal
I don't know the capital of Nepal. Please enter it:  kathmandu

Thank you! I've recorded that the capital of Nepal is kathmandu.


Enter the name of a country (or 'exit' to stop):  Nepal

The capital of Nepal is Kathmandu.

Enter the name of a country (or 'exit' to stop):  exit


Exiting the program. Thank you!
```

[17]:
```
"""One approach to analysing some encrypted data where a substitution is␣
↪suspected
is frequency analysis. A count of the different symbols in the message can be␣
↪used
to identify the language used, and sometimes some of the letters. In English,␣
↪the
most common letter is "e", and so the symbol representing "e" should appear most
in the encrypted text.
Write a program that processes a string representing a message and reports the␣
↪six
most common letters, along with the number of times they appear. Case should
not matter, so "E" and "e" are considered the same."""
def frequency_analysis(message):
    filtered_message = ''.join(char.lower() for char in message if char.
↪isalpha())
    frequency_dict = {}
    for char in filtered_message:
        if char in frequency_dict:
            frequency_dict[char] += 1
        else:
            frequency_dict[char] = 1
    sorted_letters = sorted(frequency_dict.items(), key=lambda x: x[1],␣
 ↪reverse=True)
    top_six = sorted_letters[:6]
    return top_six

def user_inputs():
    print("Frequency Analysis of Encrypted Message")
    message = input("Enter the encrypted message: ")
    return message
def display():
    message=user_inputs()
    top_letters = frequency_analysis(message)
    print("\nThe six most common letters and their counts are:")
    for letter, count in top_letters:
        print(f"'{letter}': {count}")

if __name__ == "__main__":
    display()
```

```
Frequency Analysis of Encrypted Message

Enter the encrypted message:  ssabbuudana


The six most common letters and their counts are:
'a': 3
's': 2
'b': 2
'u': 2
'd': 1
'n': 1
```

```
[ ]:
```