Project 2: The Emojifier!

In-class Date: Tuesday 07 October
Due Date: Monday 20 October

Objectives

To create a program which receives sentences (or longer) and inserts emoji icons in the correct location.

Background

As discussed in Chapter 5, strings are a collection of individual characters. For most characters, specially in European languages, the ASCII table is used (visit `https://www.asciitable.com/`). It shows the mapping between a number and a character. For example, the integer 65 is a capital 'A', and the integer 48 represents the number zero when shown in a string. The ASCII table only allows for 256 possible values, and there are a lot more than 256 different characters to be displayed when considering the world's many languages! For that, there's Unicode. Part of Unicode also encodes for emojis. The full list of all emoji characters and the byte sequences that produce them are available at `https://unicode.org/emoji/charts/full-emoji-list.html`.

Assignment

Using the starter code below,

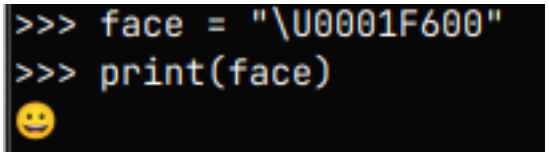
```
# Name: Alice Smith (your name here!)

emoji = {
    'happy': "\U0001F600",
}

user_input = ???

for fragment in user_input.split():
    # If 'fragment' is in 'emoji', replace 'fragment' with 'emoji[fragment]'
    ???

print(user_input)

```

Notes

To prevent Python from misinterpreting the emoji bytes or from returning an error:

1. The emoji codes are all four bytes in length (`0x00, 0x01, 0xF6, 0x00`, you'll need leading zeroes).
2. The bytes must be in quotes.
3. The bytes must start with `\U`.

In addition, ensure to handle punctuation, such as "happy" vs. "happy,".



Reminder

Assignments are your own effort. Do not share your code.

Extra Credit

Add aliases, so two or more words may map to the same emoji without having the emoji code duplicated.

Grading Rubric

Script prints:

- Program works: 75 points.
- At least 100 Emojis: 25 points.
  - Extra credit: +10 points.

What to Submit

- Your Python script.

Last modified: 29 July 2024