CSCA08 Exercise 3

Due: October 7, 2016. 5:00pm

This exercise is designed to get you writing functions using selection. In a file called `ex3.py`, create the two functions described below. As before, your file must not use `print`, `input` or `import`.

GPV Calculator

Create a function called `percent_to_gpv` that takes a percentage mark (`int`) as input, and returns the corresponding Grade Point Value (`float`). GPV is calculated as follows:

Table 1: default

Percentage Mark	GPV
90-100	4.0
85-89	4.0
80-84	3.7
77-79	3.3
73-76	3.0
70-72	2.7
67-69	2.3
63-66	2.0
60-62	1.7
57-59	1.3
53-56	1.0
50-52	0.7
0-49	0.0

You can safely assume that we will not give you values outside of the given ranges for testing, but your docstring should make it clear what values are acceptable

Card Naming

Write a function called `card_namer` that takes two single character strings, representing the value and suit of a card following the shorthand below, and returns the full name of the card. Some sneaky cheaters have been trying to slip in fake cards, like the 9 of triangles. So if the `suit` input isn't one of the recognized inputs the function should return `'CHEATER!'`[1]. You may assume that `value` will be one of the given inputs.

Input	Value
A	Ace
2 .. 9	2 .. 9
T	10
J	Jack
Q	Queen
K	King

Input	Suit
D	Diamonds
C	Clubs
H	Hearts
S	Spades

[1]HINT: There is an easy and a hard way to go about writing this function. If you're doing it properly, you should only need two if statements and no more than ~25 lines of code.

An example output of the function would be:

```
>>> card_namer('Q','D')
'Queen of Diamonds'
>>> card_namer('9','S')
'9 of Spades'
>>> card_namer('8','T')
'CHEATER!'
```

BONUS: Our Own `str()` Function

This one is a bonus. (We'll still run the auto-marker, but it won't count towards your mark)[2]. We've been using the `str()` function in class quite frequently to turn other data types into strings. However, I'd now like to build our own version, so that we can control exactly how the conversion happens.

Create a function called `my_str` that takes an `object` as input[3], and returns a string representation of that object. However, we don't want to use the boring old built-in representations, we're going to make our own.

- If the input is a string, just return it as it is
- If the object is a boolean, return `"YES"` (for `True`) or `"NO"` (for `False`)
- If the object is an integer:
  - If it's less than or equal to 10, return `"Small Number"`, for 11 - 99 return `"Medium Number"`, for anything larger return `"Large Number"`
- If the object is a float, return a standard string representation, but rounded to at most 2 decimal places
  - Try to do this with just a single call to `round()`
- If the object is any other data type, simply return the phrase `"I dunno"`

  Some sample output from the function[4]:

  ```
  >>> isinstance("Hello",str)
  True
  >>> isinstance(True,bool)
  True
  >>>
  >>> my_str("Hello")
  'Hello'
  >>> my_str(False)
  'NO'
  >>> my_str(42)
  'Medium Number'
  >>> my_str(42.0)
  '42.0'
  >>> my_str(3.1415926)
  '3.14'
  >>> my_str([1, 2, 3])
  'I dunno'
  ```

[2]It's okay if you submit a partial solution, but make sure it doesn't have any syntax errors that would prevent your other functions from running

[3]An object is a generic term for any piece of data regardless of type, you can use something like `(obj) -> str` as your type contract

[4]Note to self. Make sure not to accidentally copy/paste any extra lines of code that you typed before testing your function. Particularly not any lines that would be very helpful in completing the bonus question.