Abstract

Each of these tasks should be coded in a separate text file. You can run various commands in the interpreter (the interactive terminal), but what should be submitted are a collection of text files that can be opened in Spyder and run. The files should be submitted via NYU Classes. The NYU Classes deadline is strict! There are no exceptions.[1]

1 Warm up

Recall the three exercises we discussed in class:

1. Print the chorus of Queen's "We Will Rock You" using only two strings. FYI: The chorus is "we will we will rock you rock you".

2. We saw that $1/2 = 0.5$, while $1//2 = 0$. Make changes to an expression that uses `/` so that it produces the same value as `//`. Play with different numerator and denominator values to confirm your results.

3. Write a program that rounds positive floating point numbers to the nearest integer. This code should be two lines: the first should assign a floating point number to a variable; the second should print the rounded integer version.

Implement these programs on your own.

2 Main event

Python has lots of string methods that you may find useful for these exercises; you can find them at the Python website.

2.1 Bob Barker

Create a variable containing the string "BOB! Come get Bob's bobblehead":

```
x = "BOB! Come get Bob's bobblehead"
```

Write a program that creates a new variable with all instances of "bob" in `x` replaced with "bob barker". Capitalization does not matter: "bob", "Bob", and "bOB" are identical.

2.2 String Report

Create another string as follows:

```
x = "The quick brown fox jumps over the lazy dog"
```

Using `x`,

- Print the lowercase version;
- Print the uppercase version;
- Print the string as if it were the title of a book;
- Count and print the number of vowels in the string.
- Which letter is the letter z? For example, is it the first character? Second? As was the case with the previous questions, there is a string method to help you out with this.

2.3 Madlibs

Make a sentence based on user input. Prompt the user for four inputs (a mix of words and numbers); create a two-line story based on the input. Example:

Input

```
Enter an adjective: purple
Enter another adjective: ancient
Enter a noun: dragon
Enter a number: 2
```

Output

```
The purple bear went into the ancient house.
There she saw a 2 year old dragon.
```

Your story should be different!

2.4 Interactive addition

Request four numbers from the user using four separate input statements:

```
Please enter the first number: 1
Please enter the second number: 2
Please enter the third number: 3
Please enter the fourth number: 4
```

Concatenate the first two numbers into a single value, and the second two numbers into a single value. Print the sum of these combined numbers. In the example above the correct answer would be 46, since $46 = 12 + 34$.

Note that this should work even if the user adds whitespace before or after their input; for example:

```
Please enter the first number:        1
Please enter the second number:   2
Please enter the third number:  3
Please enter the fourth number:          4
```

Should still print 46.

3 Additional challenge

Additional challenges are not required. They are designed to give you additional practice in the event that you have finished the Main event.

3.1 Time difference

Write a program that uses three input prompts to obtain the date (day, month, and year) then prints the number of days that have occurred between that date and today.

Python has a library to assist with date/time manipulations: `datetime`. You can use the library to get todays date as follows:

```
>>> import datetime
>>> today = datetime.datetime.today()
>>> another_day = datetime.datetime(2000, 2, 28)
```

`today` and `another_day` are objects[2] that happen to have subtraction defined on them:

```
>>> diff = today - another_day
>>> toseconds = diff.total_seconds()
```

`diff` is the difference between the two datetime objects; `toseconds` is that difference in seconds. Your job is to provide the user input (used when creating the variable `another_day`, and then to convert `toseconds` to years or days, whichever is more meaningful to you.

Thus, your program should produce something like this:

```
Please enter a month: 9
Please enter a day: 2
Please enter a year: 2014
That was 1 day ago
```

Pluralization of the output ("day" versus "days") is optional.

[1] To avoid missing a deadline, it is advised that you submit whatever you have prior to leaving class. NYU Classes is okay with multiple submissions, presenting the most recent submission to the course staff.

[2] A special data type that we'll learn about later.