Python Review EXERCISES

Work with your neighbor.

Review: variables, assignment, operators, strings (indexing and slicing, concatenation, basic operations)

1. What is the value of `x`?

```
x = 15 % 4 - 1 * 8
```

2. Given a variable `x`, write an expression that evaluates to `True` if `x` is even and `False` otherwise. (No `if` statements allowed!)

3. Given a variable `x`, write an expression that evaluates to `True` if `x` is even and divisible by 5 and `False` otherwise. (No `if` statements allowed! You may use `and`, `or` or `not`.

4. Assume that string `s` has length greater than 2. Show two ways to produce the second to the last character of `s`. That is if, `s = "abcde",` the expression should produce `'d'`.

5. Given the expression

```
text = "Hello World!"
```

Write the result of the following expressions:

```
text[2:5]
```

```
text[:3]
```

```
text * 2
```

6. Write an expression that takes a string `text` and creates a new string where the first and last characters of `text` are swapped. Assume that `text` has a length of 2 or greater. (Use concatenation and slicing.)

7. Suppose you are given a string variable `text` that consists of arbitrary characters and a single "`-`" character. An example would be the following:

```
text = "ababab-ccc"
```

Create a new string from `text` that swaps all the characters before and after the "`-`" and removes the hyphen. Given the definition of `text` above, the created string would be

```
"cccababab"
```

Assume that the hyphen is not the last character of `text`.

Hint: consult the reference page for a string method that will help in solving this problem.

8. Given the expression

```
text = "?!?!?!?!!?!!??!how are you?!?!!??!?!?!"
```

Write an expression that operates on `text` and evaluates to

```
"HOW ARE YOU"
```

Hint: consult the reference page for a string method that will help in solving this problem.

Reference Sheet

Arithmetic expressions

Python supports the usual arithmetic operators such as `+, -, *, %, /,` and `//`. The arithmetic operators follow the usual PEMDAS rules of precedence.

Logical operators

Python supports the usual logical operators: `and`, `or`, and `not`.

Strings

In Python, strings are denoted by characters enclosed in single or double quotes. For example,

```
s = "abcdefg"

text = 'Hello all!'
```

Strings are indexed (or subscripted) from position 0 and are indexed using brackets so that `s[i]` results in the `i`th element of string `s.`

Given the assignments above,

```
s[3]
```

is the string `'d'.` Negative indexing starts at the end of the string.

The following table lists a few of the Python string operations.

Operation or method	Description
***s*`.find(substring)`	Returns the index where the start of the given substring appears in string ***s*** (Returns -1 if not found.)
***s*`[index1: index2]`	Returns the characters in string ***s*** from index1 (inclusive) to index2 (exclusive); if index2 is omitted, grabs till end of string
***s*`.lower(),` ***s*`.upper()`	Returns a new string with characters in ***s*** converted to lowercase or uppercase letters
***s*`.startswith(substring)`	Returns True if ***s*** starts with string substring and False otherwise
***s*`.join(substring)`	Returns a string which is the concatenation of the individual single length substrings in `substring`.
***s*`.strip(substring)`	Returns a new string with any characters in `substring` removed from both ends of `s`.
s1 + s2	Concatenates strings ***s1*** and ***s2***
len(s)	Returns the length of string ***s***