

CS 116 TUTORIAL

3





STRINGS, OUTPUT/INPUT


REMINDER

- Assignment 03 due next Wed., Feb. 5 at 10:00am.
- Midterm is on Mar. 2nd starting at 7 PM.

REVIEW

 = "SsSSsS"

.upper() => 'SSSSSS'


.lower() => 'ssssss'

- String operations
- Print
- Input and output
- Formatted strings and placeholder

COMMON STRING OPERATIONS

```
s = "string"
```

```
t = "another_string"
```

- + → concatenate strings
- len(s) → length of the string s
- s[i:j] → slicing from s[i] to s[j-1]
- s[i:j:k] → slicing from s[i] to s[j-1], stepping by k (stopping before j)
- **Some common str methods:**
 - s.find(value, index1, index2)
 - s.isalpha() 
 - s.replace(a,b)
 - dir(str) - See Module 03 Slide 8 for list of the str functions

Remember: indexing starts at 0, not 1!

PRINT

```
print (value)
```

- Returns None!
 - Use `return` to return something besides None
- Has an **effect**: prints to the screen
- Great tool for debugging!
 - Remove or comment them out before submitting your code

EQ 1

What is the value of s5?

```
s1 = "The sky is blue"
```

```
s2 = "The grass is green"
```

```
s3 = s1[:7]
```

```
s4 = s2[9:18]
```

```
s5 = s3 + s4
```

- A. "The sky is blue"
- B. "The grass is green"
- C. "The grass is blue"
- D. "The sky is green"

INPUT

```
input_var = input("Message here: ")
```

- Allows the user to enter something into the program
- The value entered is now the value of `input_var`
- Input always returns a **string**
- Has an **effect**:
 - value is being read in
 - Reads input from keyboard

UPDATES TO DESIGN RECIPE

Effects:

- Short and concise
- One for input and one for printing

Tests:

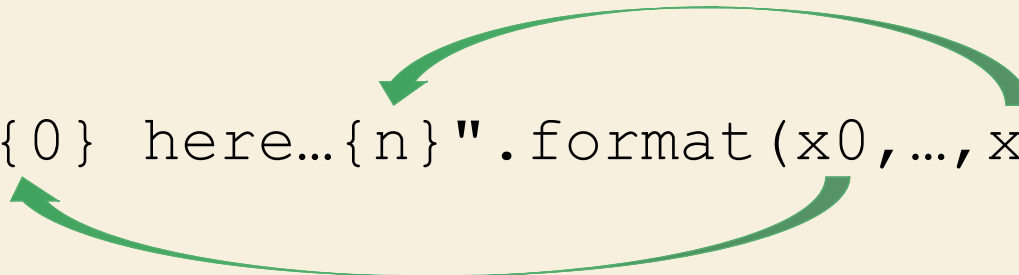
- If function returns nothing, put `None` in `check.expect`
- Three new functions:
 - `Check.set_screen`
 - `Check.set_print_exact`
 - `Check.set_input`
- All of these check functions only consume only `Str` for parameters

UPDATES TO DESIGN RECIPE

- `check.set_input(inp1, ..., inpN)`
 - Consume a set of parameters that are the expected input
 - Order is input
- `Check.set_screen(desc)`
 - Provides a **concise** description for strings printed
 - Includes input prompts
- `check.set_print_exact(str1, ..., strN)`
 - Check for accuracy in printing
 - Ignores input prompts

FORMATTING STRINGS

```
"Text {0} here...{n} ".format(x0, ..., xn)
```



- Allows you to incorporate data inside the string
- Returns a new string, like the original, but with some changes
- The symbols `{#}` are changed with the [evaluated] value of `x#`
 - Order for `format(x0, ..., xn)` matters!

CQ 2

Pretending the assignment for `animal_1` is all on one line.

What is the value of `animal`?

```
animal_1 = "Some people like {2}. Other people  
like {0}. But everyone knows that {1} get eaten  
by {2} and {2} don't like {0}. "
```

```
animal = animal_1.format("dogs", "mice", "cats")
```

- A. "Some people like cats. Other people like dogs. But everyone knows that mice get eaten by cats and cats don't like dogs. "
- B. "Some people like mice. Other people like cats. But everyone knows that dogs get eaten by mice and mice don't like cats. "
- C. "Some people like cats. Other people like mice. But everyone knows that cats get eaten by dogs and dogs don't like mice. "
- D. Error

QUESTION 1

Write a function `closest_integer` that has no parameters, but instead reads in a floating point number from console input with a prompt "What's the number?", and returns the closest integer to that number. The read-in floating point number has at most 10 digits after decimal point.

This function rounds ties up, so:

```
closest_integer()  
What's the number?: 0.5  
=> 1
```

```
closest_integer()  
What's the number?: -0.5  
=> 0
```

DO NOT use `math.ceil`, `math.floor` or `round` in your solution

QUESTION 2

Write a function `create_date` that consumes nothing, but takes keyboard input. The program has three prompts: "Enter the year: ", "Enter the month: " and "Enter the day: ". The function then returns a date in the form "dd/mm/yyyy", where dd is a 2-digit integer (between 01 and 31, depending on the month), mm is a 2-digit integer (between 01 and 12), and yyyy is a 4-digit integer.

Use string methods and string formatting (using `{}`) to complete this question.

For example,

```
create_date()  
Enter the year: 1996  
Enter the month: 06  
Enter the day: 17  
=> "17/06/1996"
```

QUESTION 3

Write a function `fill_the_string` that consumes a non-empty string `s` and a positive integer `n`, and returns a string of length `n`, created from multiple copies of `s`, where the last one is perhaps a partial copy. Assume `n >= len(s)`.

For example,

```
fill_the_string("love", 12) => "lovelovelove"
```

```
fill_the_string("truth", 12) => "truthtruthtr"
```

QUESTION 4

Write a recursive function `sum_up` that has no parameters but reads input from the keyboard. This function prompts the user with "Enter the amount of numbers to sum: ", followed by "Enter an integer: " which will read input the number of times as the number entered before. The function then prints a message "The sum is n.", where n is the sum of all the number.

For Example:

```
Enter the amount of numbers to sum: 4
```

```
Enter an integer: 3
```

```
Enter an integer: 56
```

```
Enter an integer: 7
```

```
Enter an integer: 8
```

```
The sum is 74.
```

QUESTION 5

We've seen the string function `str.count` in lectures. Using recursion, implement a version of this function, called `my_string_count(s, c)`, where `s` is any string, and `c` is a string of length one. `my_string_count` will return the number of times that the character `c` appears in the string `s`.

```
my_string_count("hello world", "l") => 3
```

```
my_string_count("abracadabra", "e") => 0
```

```
my_string_count("", "e") => 0
```