REMINDER

• Assignment 03 due next Wednesday, January 31st at 10 AM
• Midterm is on February 26th starting at 7 PM
REVIEW

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

\[ \text{\texttt{\textasciitilde}} = \text{‘Sssss’} \]

\[ \text{str.upper(\texttt{\textasciitilde})} \Rightarrow \text{‘SSSSSS’} \]
COMMON STRING OPERATIONS

\[ s = \text{“string”} \]
\[ t = \text{“another_string”} \]

• + \rightarrow concatenate strings
• \text{len}(s) \rightarrow length of the string s
• \text{s[i:j]} \rightarrow slicing from i to j-1
• \text{s[i:j:k]} \rightarrow slicing from i to j, stepping by k (stopping before j)

• \text{str} methods:
  – \text{s.find(value, index1, index2)}
  – \text{t.join(s)}
  – \text{s.split(t)}
  – \text{dir(str)} – See Module 03 Slide 8 for list of the \text{str} functions

Remember: indexing starts at 0, not 1!
print(value)

• **Returns** None!
  – **Use** `return` **to return something besides** None
• **Has an effect** – information is printed
• **Great tool for debugging**!
  – **But remove them before submitting your code**
user_input = input("Message here: ")

• Allows the user to enter something into the program
• The value entered is now the value of user_input
• Input always returns a string
• Has an effect – value is being read in
**FORMATTING STRINGS**

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

- Allows you to input 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!
QUESTION 1

Write a function closest_integer that has no argument, 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 or round in your solution
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.

For example,

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

Use string methods and string formatting (using `{}`) to complete this question.
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"
```
Write a recursive function `sum_up` that has no parameters but reads input from the keyboard. This function prompts the user with "Enter an integer or 'stop' to print sum: " and reads in a series of integers until the user types "stop". The function then prints a message "The sum is n", where n is the sum of all the numbers entered.

For Example:

Enter an integer or 'stop' to print sum: 3
Enter an integer or 'stop' to print sum: 56
Enter an integer or 'stop' to print sum: 7
Enter an integer or 'stop' to print sum: 8
Enter an integer or 'stop' to print sum: stop

The sum is 74