Welcome!

We'll get started shortly …

CS 49 Section

Week 4

Surajit A Bose

Agenda

- Logistics and Zoom poll
- Review of lecture concepts
  - Variables
  - Casting variables
  - Console, **`input()`** and **`print()`**
- Problems
  - Tiny Mad Libs
  - Squaring a number

Logistics

How to get hold of me / get help

- The class forum. Feel free not only to ask, but also to answer questions!
- Surajit's office hours:
  - Fridays 12 noon–1p, directly after section
  - By appointment on Zoom
- Lane's office hours
- Canvas inbox or Pronto inbox for Lane
- Canvas inbox (preferred) or Pronto for Surajit
- Sina's support section, Fridays 2p–3p on Zoom
- Email bosesurajit@fhda.edu, 24 hr turnaround
- Online or in-person tutoring via the STEM center (Room 4213)
- The section GitHub repo has lecture and section slides and solutions

Any Questions?

Lecture Review: Variables

Variables

- A **named** place in memory that holds a **value** of a particular **type**
- A variable is a **location in memory**
- Name, value, and type are the three key characteristics of a variable
  - The variable **is** or **has** a **name**. The variable name is an identifier or tag that specifies the memory location
  - The variable **has** or **holds** a **value**, such as 8.04, -93, or "steve@apple.com"
  - The value **is of** or **has** a **type**, such as **`float`**, **`int`**, or **`str`**
- Let's explore each of name, value, and type in the next few slides

Variables: Names

- A **named** place in memory that holds a **value** of a particular **type**
- Names:
  - Are case-sensitive: `cumulative_GPA` vs `cumulative_gpa`
  - Must begin with a letter or underscore
  - Must not be a reserved word such as `for` or `def`
  - Should not replicate names for built-in functions like `print()`
  - Should be in `snake_case` if more than one word long
  - Should be short but descriptive
- "Must" indicates what is enforced by Python
- "Should" indicates best practices

Variables: Names

- Which of the following variable names meet the specified standards?
  - Must begin with a letter or underscore[1]
  - Must not be a reserved word such as **`for`** or **`def`**[1]
  - Should not replicate names for built-in functions like **`print()`**[2]
  - Should be in **`snake_case`** if more than one word long[3]
  - Should be short but descriptive[3]
- [1]illegal in Python to violate this; [2]legal but highly inadvisable; [3]recommended by convention

```
result    101_dalmatians   num_students   pass   input
numStudents   longitude   y   total   main   1atitude
```

Variables: Names

- Which of the following variable names meet the specified standards?
    - Must begin with a letter or underscore
    - Must not be a reserved word such as **`for`** or **`def`**
    - Should not replicate names for built-in functions like **`print()`**
    - Should be in **`snake_case`** if more than one word long
    - Should be short but descriptive
- **`red`** : illegal in Python to violate this, **`purple`** : legal but highly inadvisable, **`yellow`** : legal but against convention, **`green`** : legal

```
result    101_dalmatians    num_students    pass    input
numStudents    longitude    y    total    main    1atitude
```

Variables: Value

- A **named** place in memory that holds a **value** of a particular **type**
- Value:
    - Assigned with the equals sign, e.g. **`answer = 42`**
    - This is also called "binding": the value **`42`** is bound to the variable **`answer`**
    - Can be the result of an expression, e.g. **`gpa = qual_points / num_credits`**
    - The right hand side of the equals sign is **evaluated** , then the result is placed into the variable on the left hand side
    - This means we can have commands like **`balance = balance + interest`**

Variables: Type

- A **named** place in memory that holds a **value** of a particular **type**
- Primitive types:
  - **`int`**, for numbers without a decimal point ("How many?"), e.g. -3
  - **`float`**, for numbers with a decimal point ("How much?"), e.g. 3.14159
  - **`str`**, for text characters including letters, numerals, punctuation, etc.
  - **`bool`**, for two specific values **`True`** and **`False`**
- What are the types of the following values? Some types might not be determinable from just the information given. And remember Karel!

```
35   qual_points / credits    "front_is_clear()"    0.0
front_is_clear()    '8 + 11'    wage_rate * hours_worked
```

Variables: Type

- A **named** place in memory that holds a **value** of a particular **type**
- Primitive types:
  - `int`, for numbers without a decimal point ("How many?"), e.g. -3
  - `float`, for numbers with a decimal point ("How much?"), e.g. 3.14159
  - `str`, for text characters including letters, numerals, punctuation, etc.
  - `bool`, for two specific values `True` and `False`
- What are the types of the following values? `red` : string, `green` : int, `purple` : float, `magenta` : bool, `yellow` : indeterminate

```
35   qual_points / credits    "front_is_clear()"    0.0
front_is_clear()    '8 + 11'    wage_rate * hours_worked
```

Any Questions?

Lecture Review: Type Casting

Type casting

- A value of one type can be changed into a value of another type
- This is called type casting
- Given **`int_val = 3`**, what would the results be of the following cast operations?

```
float_val = float(int_val)
str_val = str(int_val)
str_float_val = str(float_val)
int_val_2 = int(str_float_val)

e = 2.7183
int_e = int(e)
```

Type casting

- A value of one type can be changed into a value of another type
- This is called type casting
- Given **`int_val = 3`**, what would the results be of the following cast operations?

```
float_val = float(int_val)          # result: 3.0
str_val = str(int_val)              # result: '3'
str_float_val = str(float_val)      # result: '3.0'
int_val_2 = int(str_float_val)      # result: error!

e = 2.7183
int_e = int(e)                      # result: 2
```

Lecture Review: Console I/O

The console, aka the terminal

```
Terminal
%
```

The console, aka the terminal

- Where a Python program displays its output (via the **`print()`** function)
- Where the program gets its user input (via the **`input()`** function)
- Console input and output is always in the form of strings
  - Anything printed to the console must be a string
  - Any variable must be cast into the appropriate type before outputting it, e.g., an **`int`** must be cast into its **`str`** representation
  - Anything brought into the program from the console will be a string
  - Any variable from console input must be cast into the appropriate type such as **`int`** or **`float`** before it can be used in the program

Printing to the console: old style

Assume that the variable **`num_classes`** has some value of type **`int`**. Here are two somewhat outmoded ways of printing to the console (i.e., of constructing a string to send to the screen).

- String concatenation, need to cast variables to **`str`** and include spaces

```
print('I have ' + str(num_classes) + ' classes this quarter')
```

- Commas, will automatically cast variables and add spaces

```
print('I have', num_classes, 'classes this quarter')
```

These are fine, but the cool kids have moved on to ...

Printing to the console: f-strings

… **`format strings`**, aka f-strings.

```
print(f'I have {num_classes} classes this quarter')
```

- Put the letter **f** before the open quotation mark
- Put all variable names in braces
- Python will cast all variables to string

These are all ways of constructing a new string by putting together various elements such as smaller strings or variables cast to string.

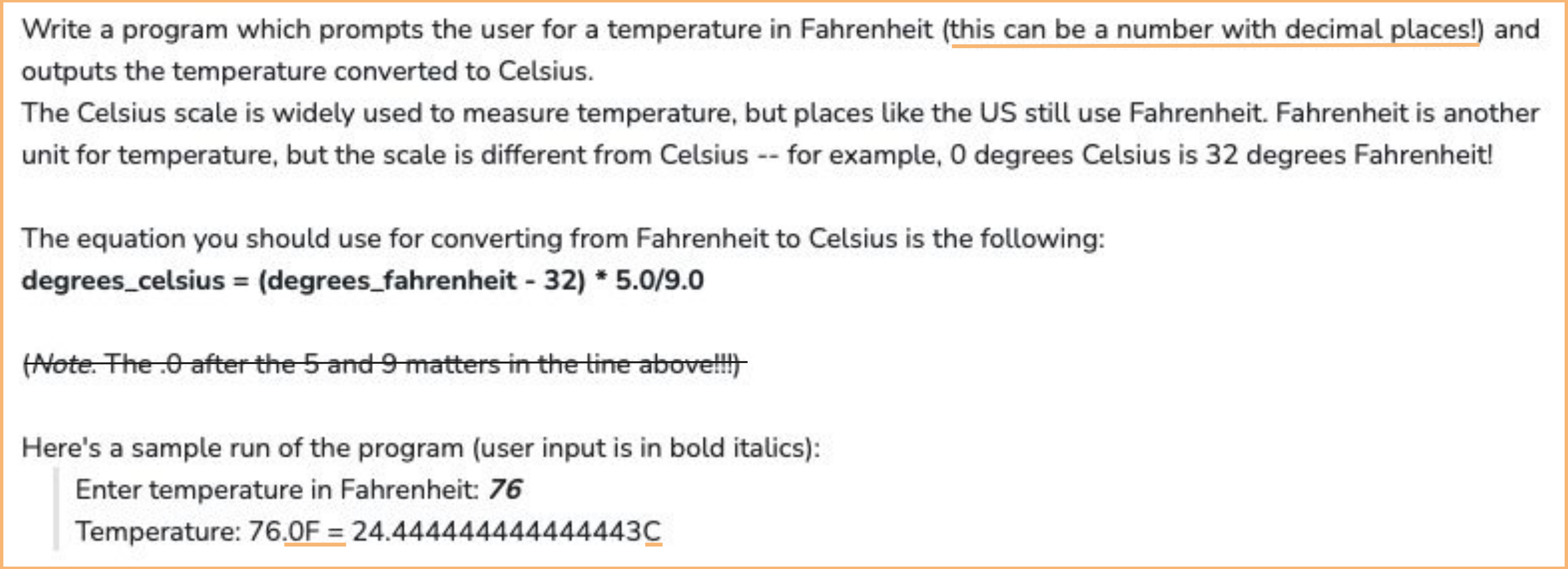
Any Questions?

Let's use all that to solve this week's problems!

Example problem: Fahrenheit to Celsius

https://codeinplace.stanford.edu/cs49-w24/ide/a/ftoc

Fahrenheit to Celsius



Write a program which prompts the user for a temperature in Fahrenheit (this can be a number with decimal places!) and outputs the temperature converted to Celsius.
The Celsius scale is widely used to measure temperature, but places like the US still use Fahrenheit. Fahrenheit is another unit for temperature, but the scale is different from Celsius -- for example, 0 degrees Celsius is 32 degrees Fahrenheit!

The equation you should use for converting from Fahrenheit to Celsius is the following:
degrees_celsius = (degrees_fahrenheit - 32) * 5.0/9.0

~~(*Note*. The .0 after the 5 and 9 matters in the line above!!!)~~

Here's a sample run of the program (user input is in bold italics):

```
Enter temperature in Fahrenheit: 76
Temperature: 76.0F = 24.444444444444443C
```

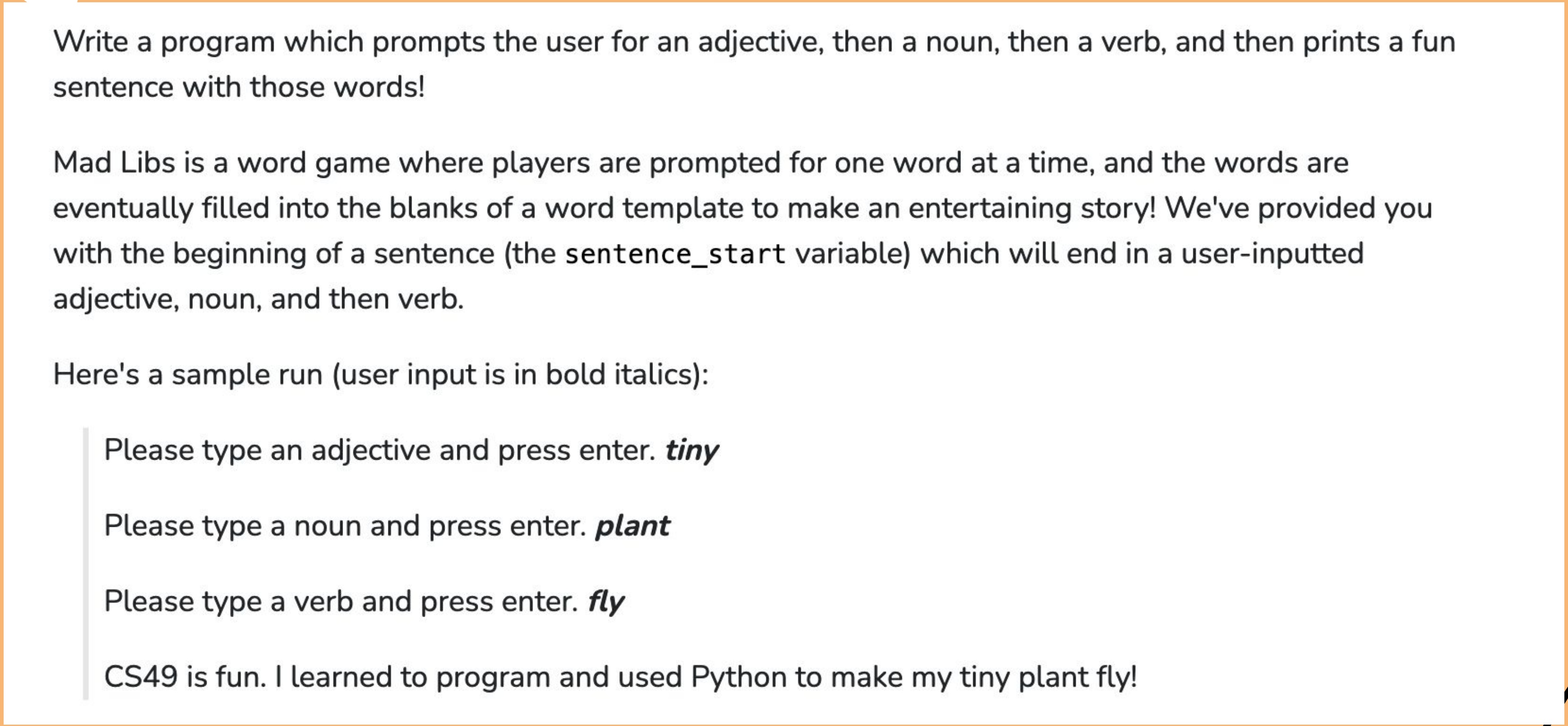
Steps to Solution

- Get input from user and save it in a variable
  - What type will the input be?
  - As what type should it be cast?
  - What would be a good variable name?
- Apply the conversion formula and save the result in a variable
  - What would be a good variable name?
- Print the result to the screen **using an f-string**

Section problem 1:
Tiny Mad Libs

https://codeinplace.stanford.edu/cs49-w24/ide/a/tinymadlibs

Tiny Mad Libs



Write a program which prompts the user for an adjective, then a noun, then a verb, and then prints a fun sentence with those words!

Mad Libs is a word game where players are prompted for one word at a time, and the words are eventually filled into the blanks of a word template to make an entertaining story! We've provided you with the beginning of a sentence (the `sentence_start` variable) which will end in a user-inputted adjective, noun, and then verb.

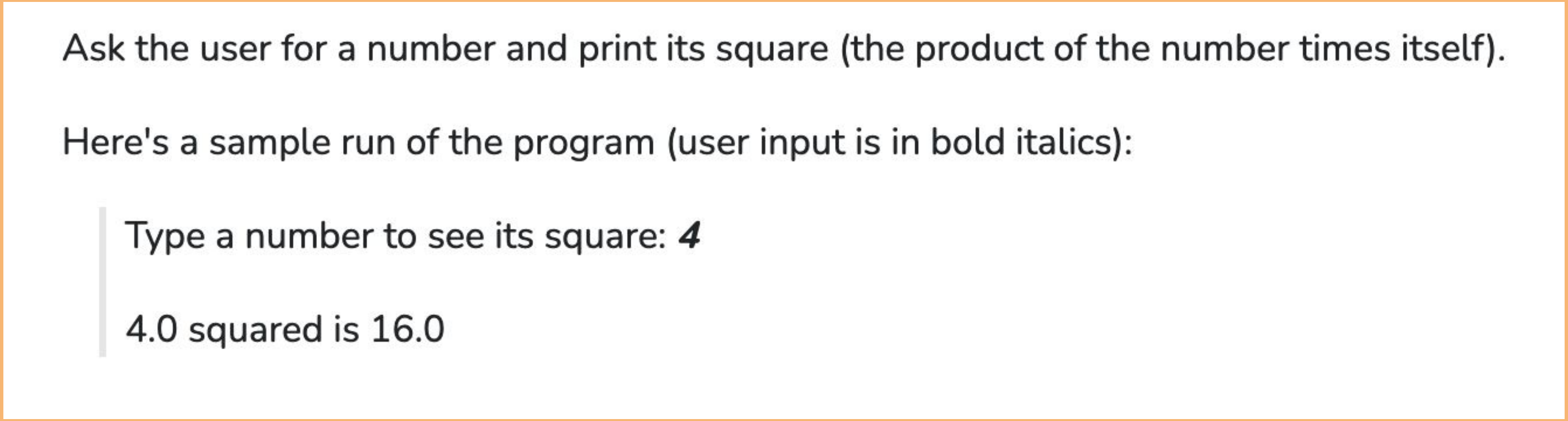
Here's a sample run (user input is in bold italics):

> Please type an adjective and press enter. ***tiny***
>
> Please type a noun and press enter. ***plant***
>
> Please type a verb and press enter. ***fly***
>
> CS49 is fun. I learned to program and used Python to make my tiny plant fly!

Section problem 2: Square Number

https://codeinplace.stanford.edu/cs49-w24/ide/a/squarenum

Square Number



Ask the user for a number and print its square (the product of the number times itself).

Here's a sample run of the program (user input is in bold italics):

> Type a number to see its square: ***4***
>
> 4.0 squared is 16.0

That's all, folks!

Next up: Expressions!