

The background is a solid pink color. It is decorated with various hand-drawn geometric shapes in white and black. These include a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a black circle in the bottom right, a white circle in the bottom right, a black plus sign in the bottom left, a white triangle in the bottom center, a black circle in the bottom center, and a black plus sign in the bottom left.

# Welcome!

We'll get started shortly ...



# CS 49 Section


Week 4

Surajit A Bose





# Agenda

- Logistics and check-ins
  - Review of lecture concepts
    - Variables
    - Casting variables
    - Console, `input()` and `print()`
  - Problems
    - [Tiny Mad Libs](#)
    - [Squaring a number](#)
- 

# Logistics





# Polls



- Please fill out the anonymous Zoom poll about the time this class takes, including watching videos, reading the textbook, and working through the problems!
- Please keep an eye out for another anonymous Google survey asking how section can be more useful for you.





# Quick Check-in




Let's quickly go round the room! Pick any one of the following to complete the sentence: As the weeks go by, this class is getting:

- easier, because ...
- more difficult, because ....
- more fun, because ...
- more stressful, because ...





# How to get hold of me / get help<sub>+</sub>

- The [section forum](#), 24 hr turnaround
  - Email: [bozesurajit@fhda.edu](mailto:bozesurajit@fhda.edu), 24 hr turnaround
  - Office hours:
    - On campus: Tuesdays 12:00 noon to 1:30 pm, room 4218 in the STEM center
    - By appointment on Zoom
  - Other resources:
    - Contact Lane via Canvas
    - [Online](#) or [in-person](#) tutoring via the STEM center (Room 4213)
- 

# 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<sup>1</sup>
  - Must not be a reserved word such as **for** or **def**<sup>1</sup>
  - Should not replicate names for built-in functions like **print()**<sup>2</sup>
  - Should be in **snake\_case** if more than one word long<sup>3</sup>
  - Should be short but descriptive<sup>3</sup>
- <sup>1</sup>illegal in Python to violate this; <sup>2</sup>legal but highly inadvisable;  
<sup>3</sup>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**
  - 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
- What would the results be of the following cast operations?

```
int_val = 3
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
- What would the results be of the following cast operations?

```
int_val = 3
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 +

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')
```

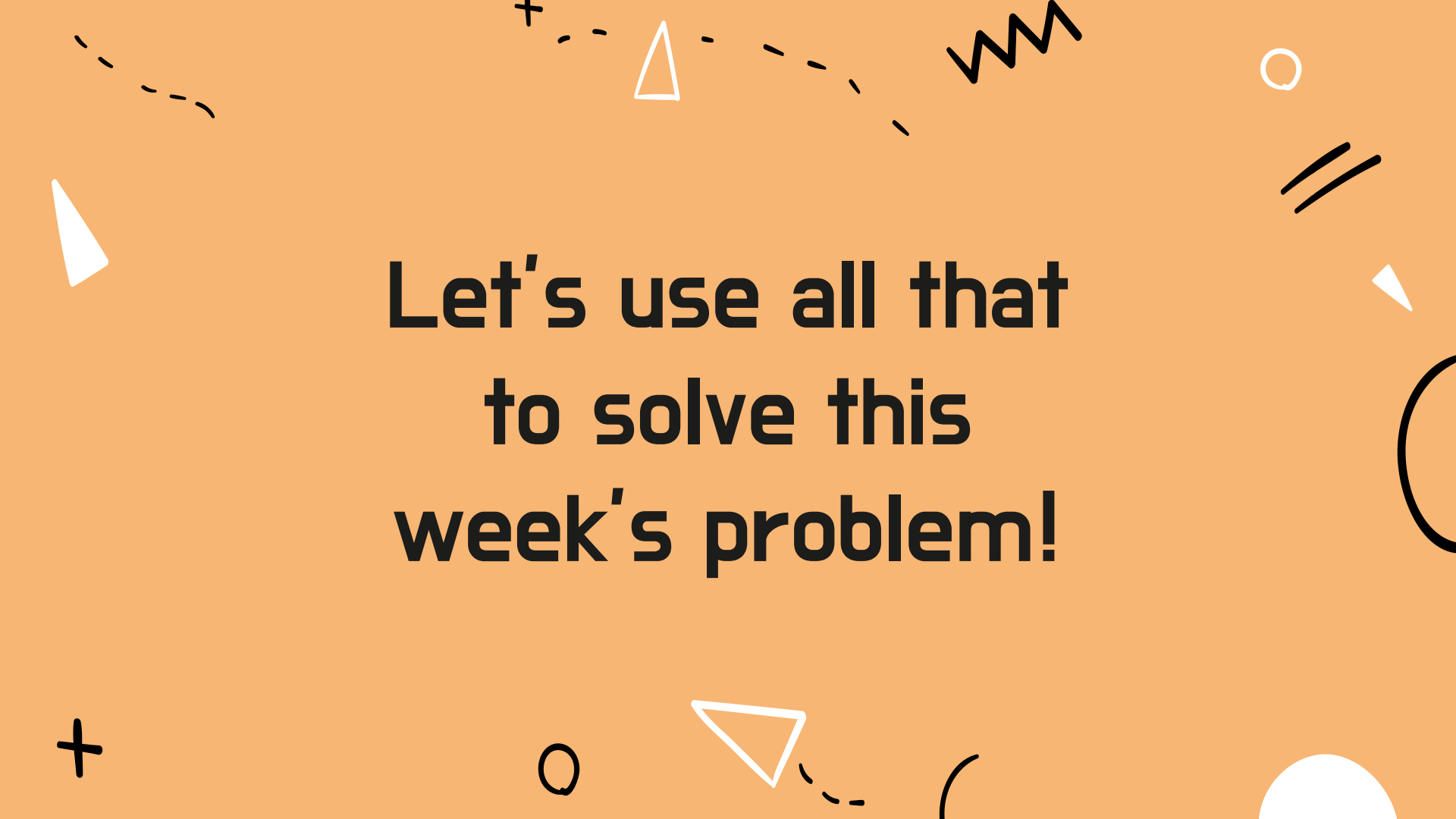
- Put the letter **f** before the open quotation mark
- Put all variable names in braces
- Python will cast all variables to string

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



The background is a solid orange color. It is decorated with various hand-drawn geometric shapes in white and black. These include a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a large black circle in the bottom right, a white circle in the bottom right, a black plus sign in the bottom left, a white circle in the bottom center, a white triangle in the bottom center, and a black plus sign in the bottom center.

**Any Questions?**

The background is a solid orange color. It is decorated with various white and black geometric shapes and lines. In the top left, there is a dashed line and a small white triangle. In the top center, there is a dashed line and a white triangle. In the top right, there is a dashed line, a black zigzag line, a small white circle, and two parallel black lines. In the middle left, there is a white triangle. In the middle right, there is a white triangle and a large black circle. In the bottom left, there is a black plus sign. In the bottom center, there is a black circle and a white triangle. In the bottom right, there is a black line and a white circle.

**Let's use all that  
to solve this  
week's problem!**



# Section problem 1: Tiny Mad Libs

<https://codeinplace.stanford.edu/foothill-cs49/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/foothill-cs49/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



The background is a solid pink color. It is decorated with various hand-drawn geometric shapes and symbols in white and black. These include a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a large black circle in the bottom right, a white circle in the bottom right, a black plus sign in the bottom left, a white circle in the bottom left, a white triangle in the bottom left, and a dashed line in the bottom left.

# That's all, folks!

Next up: Expressions!