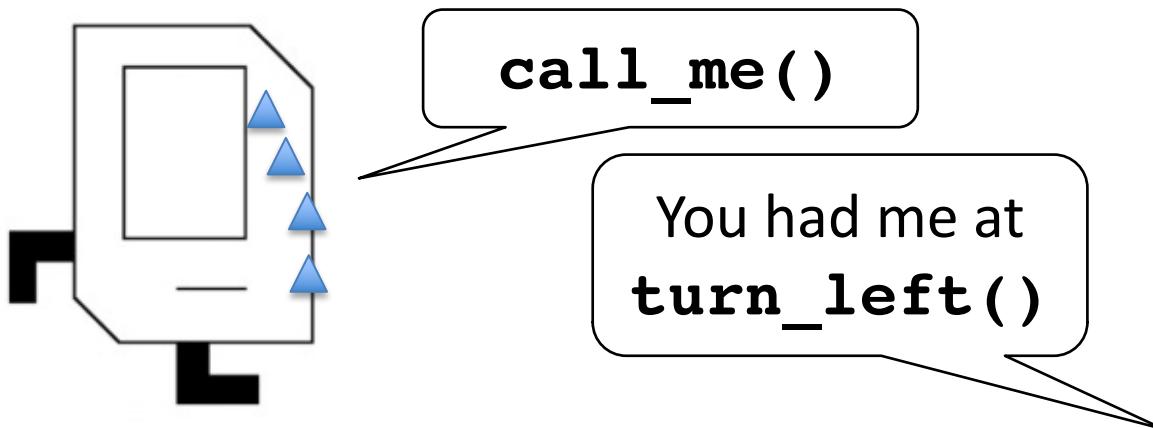




# Introduction to Python

# Bye, bye, Karel!



# More on Programming Style

....

File: SteepleChaseKarel.py

Karel runs a steeple chase that is 9 avenues long.  
Hurdles are of arbitrary height and placement.

....

To run a race that is 9 avenues long, we need to move forward or jump hurdles 8 times.

....

```
def main():
    for i in range(8):
        if front_is_clear():
            move()
        else:
            jump_hurdle()
```

Consistent  
indentation

Comments for program  
and *every* function

Decomposition principle:  
Each function should solve  
one step of problem

Pre-condition: Facing East at bottom of hurdle

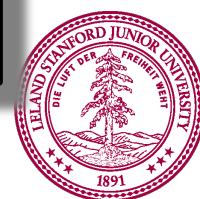
Post-condition: Facing East at bottom in next avenue after hurdle

....

```
def jump_hurdle():
    ascend_hurdle()
    move()
    descend_hurdle()
```

Short functions  
(usually 1-15 lines)

Descriptive *names*  
(snake\_case)



# What's Mozart Doing Now?



```
if mehran_teaching():  
    not_funny()
```

```
while mehran_teaching():  
    not_funny()
```

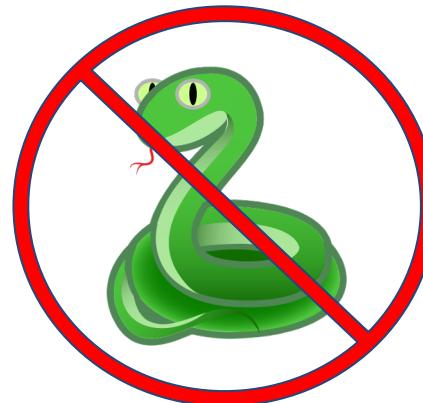


# Welcome to Python

Guido van Rossum  
(Creator of Python)

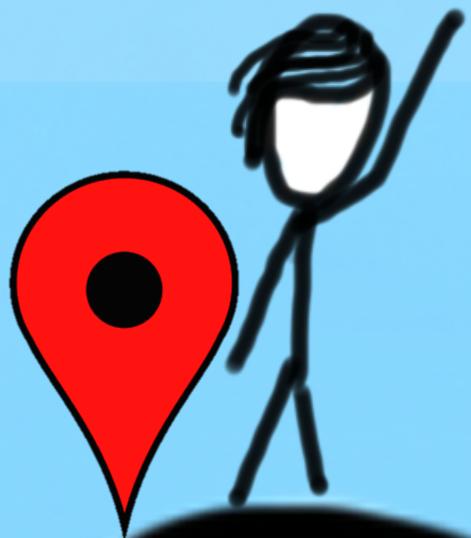


Monty Python's Flying Circus



# Today's Goal

1. Introduction to Python
2. Understanding variables



# Our First Python Program

```
"""
File: helloworld.py
-----
This is our first python program. It is customary to
have a programmer's first program write "hello world"
(inspired by the first program in Brian Kernighan and
Dennis Ritchie's classic book, 'The C Programming Language.')
"""
```

```
def main():
    print("hello, world!")
```

# Our First Python Program

The screenshot shows a web browser window for the edstem.org platform. The URL is [edstem.org/us/courses/10000/lessons/12840/slides/65572](https://edstem.org/us/courses/10000/lessons/12840/slides/65572). The page title is "ed Code in Place 2021 – Lessons". The main content area is titled "helloworld program". On the left sidebar, under "Lecture 4: Introduction to Python", the "helloworld program" slide is selected. The code editor displays the following Python script:

```
1 """"
2 File: helloworld.py
3 -----
4 This is our first python program...It is customary to
5 have a programmer's first program write "hello world"
6 (inspired by the first program in Brian Kernighan and
7 Dennis Ritchie's classic book, 'The C Programming Language.')
8 """
9
10
11 def main():
12     print("hello, world!")
13
14
15 # This provided line is required at the end of a Python file
16 # to call the main() function.
17 if __name__ == '__main__':
18     main()
```

The status bar at the bottom shows the file path "/home/helloworld.py", "Spaces: 4 (Auto)", and "All changes saved". Below the code editor is a "Terminal" section with a "Submit" button. A red box highlights the "Click here to activate the terminal" link.

# Our First Python Program

The screenshot shows a web-based code editor interface for a Python program titled "helloworld program". The interface includes a navigation bar with "ed Code in Place 2021 – Lessons", a search bar with the URL "edstem.org/us/courses/10000/lessons/12840/slides/65572", and various toolbar icons. The main area displays the code for "helloworld.py". The code is as follows:

```
1 """
2 File: helloworld.py
3 -----
4 This is our first python program...It is customary to
5 have a programmer's first program write "hello world"
6 (inspired by the first program in Brian Kernighan and
7 Dennis Ritchie's classic book, 'The C Programming Language.')
8 """
9
10
11 def main():
12     print("hello, world!")
13
14
15 # This provided line is required at the end of a Python file
16 # to call the main() function.
17 if __name__ == '__main__':
18     main()
```

The status bar at the bottom indicates the file path "/home/helloworld.py", "Spaces: 4 (Auto)", and "All changes saved". Below the code editor is a terminal window showing the command "[user@sahara ~]\$". There are "Submit" and "Reset" buttons for the terminal.

# Our First Python Program

The screenshot shows a web-based code editor interface from [edstem.org](https://edstem.org/us/courses/10000/lessons/12840/slides/65572). The title bar reads "ed Code in Place 2021 – Lessons". The main content area is titled "helloworld program". On the left sidebar, under "Lecture 4: Introduction to Python", the "helloworld program" is selected. The code editor displays the following Python script:

```
1 """
2 File: helloworld.py
3 -----
4 This is our first python program...It is customary to
5 have a programmer's first program write "hello world"
6 (inspired by the first program in Brian Kernighan and
7 Dennis Ritchie's classic book, 'The C Programming Language.')
8 """
9
10
11 def main():
12     print("hello, world!")
13
14
15 # This provided line is required at the end of a Python file
16 # to call the main() function.
17 if __name__ == '__main__':
18     main()
```

The status bar at the bottom indicates the file path is "/home/helloworld.py" and "Spaces: 4 (Auto)". A green dot next to "All changes saved" means the code is up-to-date. Below the code editor is a terminal window showing the command "[user@sahara ~]\$ python helloworld.py]". There are "Submit" and "Reset" buttons for the terminal.

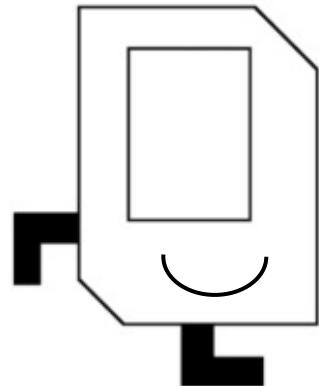
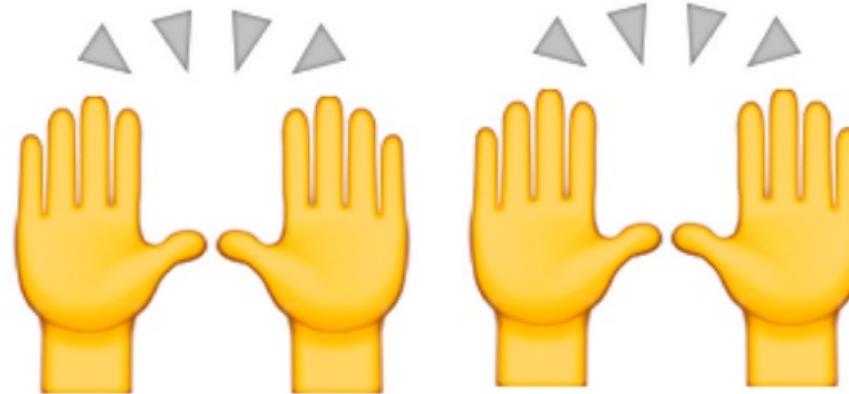
# Our First Python Program

The screenshot shows a web browser window for 'ed Code in Place 2021 – Lessons' at the URL [edstem.org/us/courses/10000/lessons/12840/slides/65572](https://edstem.org/us/courses/10000/lessons/12840/slides/65572). The page title is 'helloworld program'. The left sidebar shows 'Lecture 4: Introduction to Python' with 'helloworld program' selected. The main area displays the Python code for a 'helloworld.py' file:

```
1 """
2 File: helloworld.py
3 -----
4 This is our first python program...It is customary to
5 have a programmer's first program write "hello world"
6 (inspired by the first program in Brian Kernighan and
7 Dennis Ritchie's classic book, 'The C Programming Language.')
8 """
9
10
11 def main():
12     print("hello, world!")
13
14
15 # This provided line is required at the end of a Python file
16 # to call the main() function.
17 if __name__ == '__main__':
18     main()
```

The status bar indicates the file path is '/home/helloworld.py' and spaces are set to '4 (Auto)'. A green dot next to 'All changes saved' indicates no pending changes. Below the code editor is a terminal window showing the command `[user@sahara ~]$ python helloworld.py` and its output `hello, world!`. There are 'Submit' and 'Reset' buttons for the terminal.

# You're now all Python programmers!



hey\_that\_looks\_  
like\_what\_I\_  
taught\_them()

# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```



# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

This program adds two numbers.



# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

This program adds two numbers.  
Enter first number:



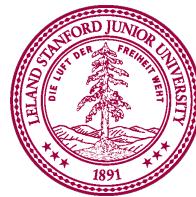
# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1 "9"

This program adds two numbers.

Enter first number: 9



# Another Program

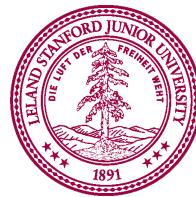
```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

This program adds two numbers.

Enter first number: 9



# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")  
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

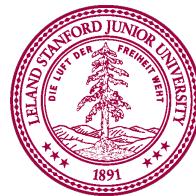
num1

9

This program adds two numbers.

Enter first number: 9

Enter second number:



# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ") (highlighted line)
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

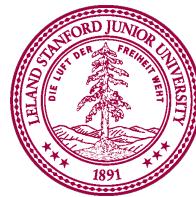
num2

"17"

This program adds two numbers.

Enter first number: 9

Enter second number: 17



# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

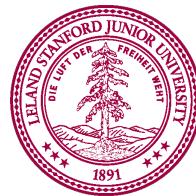
num2

17

This program adds two numbers.

Enter first number: 9

Enter second number: 17



# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

num2

17

total

26

This program adds two numbers.

Enter first number: 9

Enter second number: 17



# Another Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

num2

17

total

26

This program adds two numbers.

Enter first number: 9

Enter second number: 17

The total is 26.



# print function

```
print("This program adds two numbers.")
```

- **print** command prints text to the terminal
- Text printed is between double quotes ("text")
  - Can also be between single quotes ('text')
  - Choice of quotes depends on text you are printing
    - Double quotes when text contains single quotes  
`print("no, you didn't") → no, you didn't`
    - Single quotes when text contains double quotes  
`print('say "hi" Karel') → say "hi" Karel`



# input function

```
num1 = input("Enter first number: ")
```

- **input** command gets text input from the user
- Prints text specified in double/single quotes
  - Then waits for user input
  - Here, user input from **input** is put in a variable (**num1**)
  - The user input is considered text, even if user entered a number
- We'll talk more about **input** function later

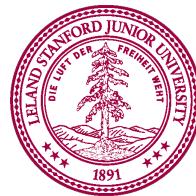


# What is a Variable?

x 10

- A **variable** is a place to store information in a program
- It associates a **name** with a **value**
- You can create a new variable by **assigning** a value:

**x = 10**



# What is a Variable?

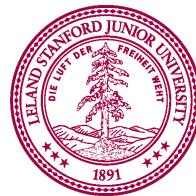


- A **variable** is a place to store information in a program
- It associates a **name** with a **value**
- You can create a new variable by assigning a value:

`x = 10`

- The value can change with a new assignment

`x = 5`



# What is a Variable?

x 12

- A **variable** is a place to store information in a program
- It associates a **name** with a **value**
- You can create a new variable by assigning a value:

`x = 10`

- The value can change with a new assignment

`x = 5`

- You can set the value using mathematical expressions

`x = 5 + 7`

- More about expressions next class



# Variable Assignment

- You use the equal sign (=) to assign to a variable
  - The first time you assign a value to a variable, you create it
  - Subsequent assignments give the variable a new value
- Assignment is not the same as "equals" in math
  - Assignment: first evaluate right-hand side, then assign to the variable on the left-hand side
  - Consider the following code:

```
total = 5
total = total + 1
```
- Variables are only visible inside the function in which they are created (called "scope" of variable)
  - If you create a variable in `main()`, its only visible in `main()`
  - More on that next class



# Variable Names

- Variable names must:
  - Start with a letter or an underscore ( `_` )
  - Contain only letters, digits, or underscores
  - Cannot be a "built in" command in Python (e.g., `for`)
- Variable names are case sensitive
  - **Hello** is not the name as **hello**
- Variable names should:
  - Be descriptive of the value they refer to
    - E.g., `x` is only a good name if it's a coordinate
  - Be in snake case (e.g., `num_students`)



# Suitcase Analogy

x 12

- When you store information in a variable, it becomes a Python *object*
  - Objects come in different sizes and types
- Think about a Python object as a suitcase stored in your computer's memory
  - Object take up different amounts of RAM depending on what you're storing.



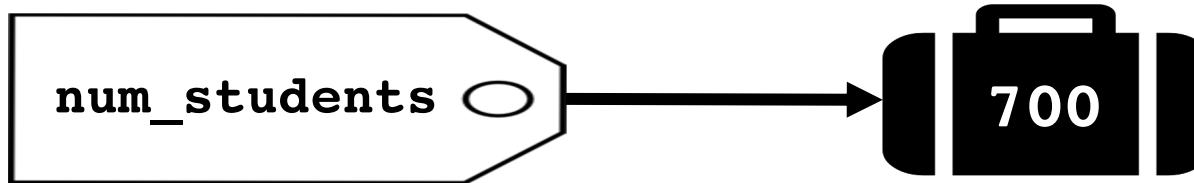
You have space for millions on suitcases!

# Suitcase Analogy

- Variable is a luggage tag that gives a *name* to suitcase

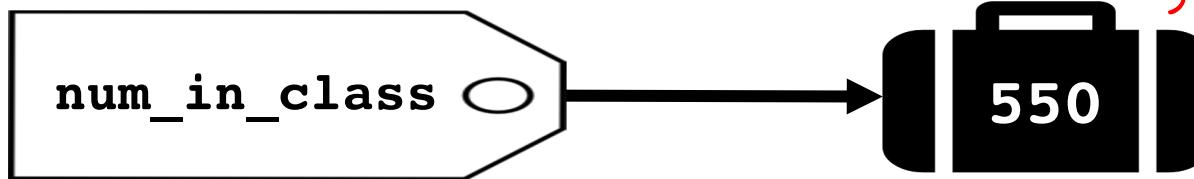
`num_students = 700`

- *Value* is what is stored in the suitcase
- Create the tag/suitcase the first time you assign to variable

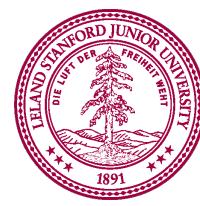
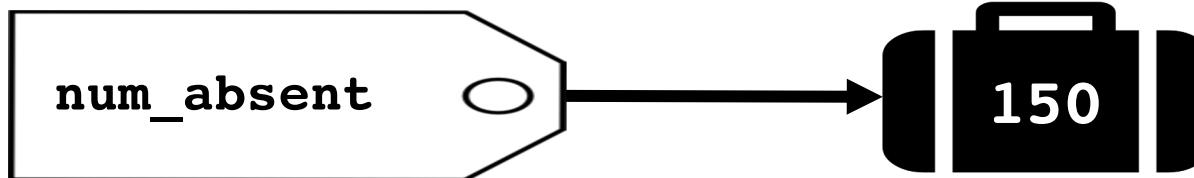


`num_in_class = 550`

Python handles the  
baggage for you!



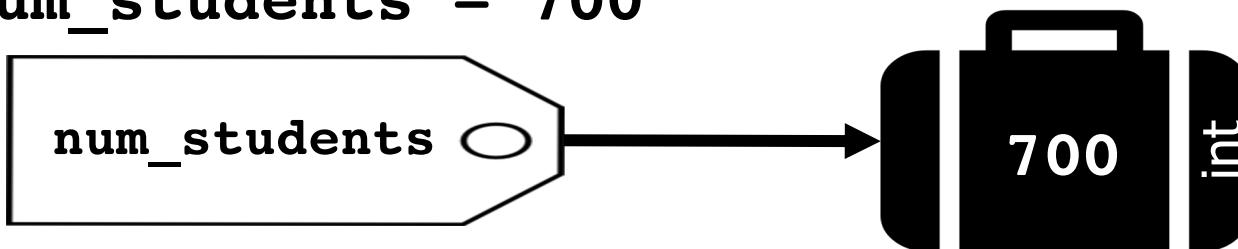
`num_absent = num_students - num_in_class`



# Types

- Each suitcase knows what **type** of information it carries

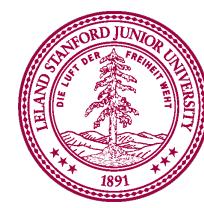
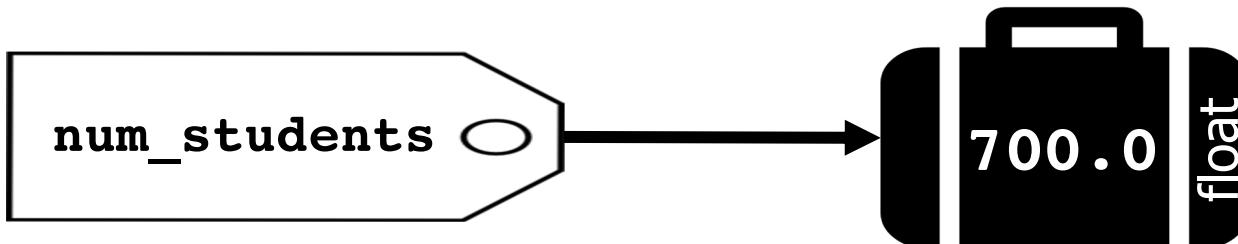
```
num_students = 700
```



- Value stored in suitcase is an integer (called an **int** in Python)
- Suitcase keeps track of **type** of data that is stored there

```
num_students = 700.0      # note decimal point
```

- Now, value stored is a real number (called a **float** in Python)



# Some Types in Python

- **int:** integer value (no decimal point)

`x = 10`      `y = -2`

- **float:** real number value (has decimal point)

`x = 5.0`      `y = -3.7`

- **string:** text characters (between single/double quotes)

`x = "hello"`   `y = '10'`

– Note: the string "5" is **not** the same as the integer 5

- **bool:** Boolean logical values (**True/False**)

`x = True`      `y = False`

- More on strings and bools in a few days



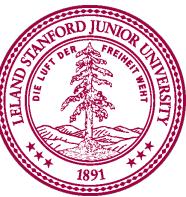
# Why Do We Have int and float?

- How much do I weigh?
  - Answer can be a real valued number
  - There is no "next" number
  - This would be a float
- How many children do I have?
  - Answer is an integer
  - There is a well-defined "next" number
  - This would be an int



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

This program adds two numbers.

- **print** command is displaying a **string**



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1 "9"

This program adds two numbers.  
Enter first number: 9

- **input** command gives you back a **string**
  - Even if the user types in a number



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

This program adds two numbers.

Enter first number: 9

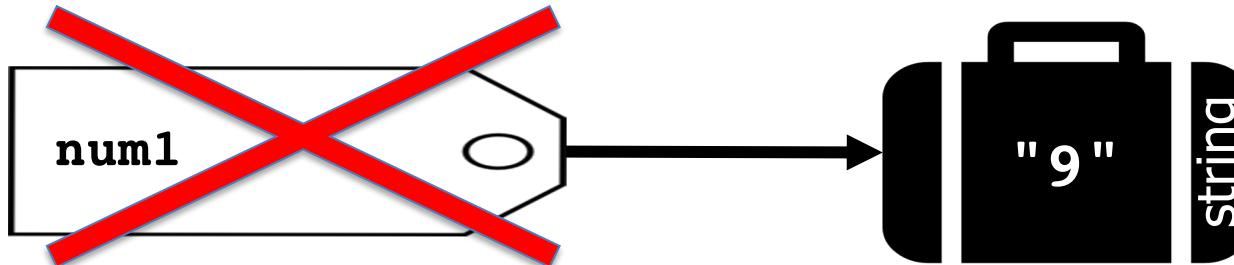
- Create **int** version of **string** and assign it back to **num1**



# Show Me The Luggage!

- **input** command gives you back a **string**

```
num1 = input("Enter first number: ")
```

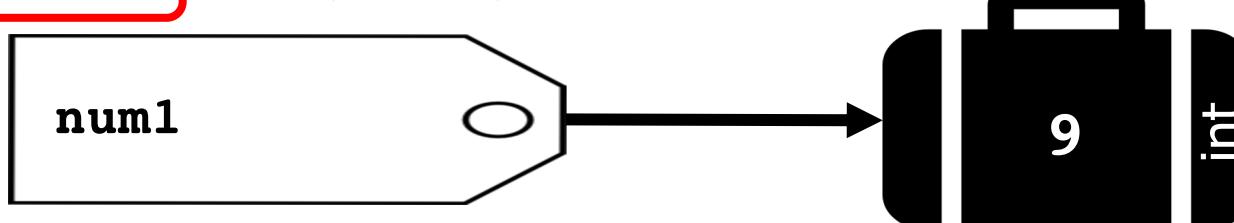


- We create an integer version of **num1**

```
num1 = int(num1)
```

- Create a new suitcase that has **int** version of **num1**
- Then assign the tag num1 to that piece of luggage

```
num1 = int(num1)
```



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

This program adds two numbers.

Enter first number: 9

- Create **int** version of **string** and assign it back to **num1**



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ") (highlighted)
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

This program adds two numbers.

Enter first number: 9

Enter second number:



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ") num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

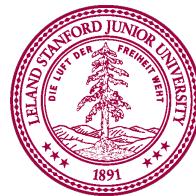
num2

"17"

This program adds two numbers.

Enter first number: 9

Enter second number: 17



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

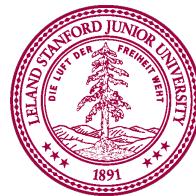
num2

17

This program adds two numbers.

Enter first number: 9

Enter second number: 17



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

num2

17

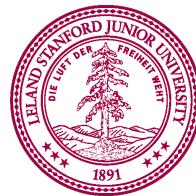
total

26

This program adds two numbers.

Enter first number: 9

Enter second number: 17



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

num2

17

total

26

This program adds two numbers.

Enter first number: 9

Enter second number: 17

The total is 26.



# What's Going on With `print`

- Adding strings in `print` command?!

```
print("The total is " + str(total) + ".")
```

- The `+` operator concatenates strings together

```
str1 = "hi"
```

```
str2 = " "
```

```
str3 = "there"
```

```
str4 = str1 + str2 + str3
```

- `total` is integer, so we need to create a string version

```
str(total)
```

- String version of `total` is a new value that is concatenated to produce final string that is printed
- Original variable `total` is still an `int`



# Recall, Our Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

num1

9

num2

17

total

26

This program adds two numbers.

Enter first number: 9

Enter second number: 17

The total is 26.



# Side note about `print`

- You can `print` numbers by themselves directly
  - Only need to create string version of numbers when printing other text (strings) with them

```
def main():
    x = 10
    y = 3.5
    print(x)
    print(y)
    print("x = " + str(x))
```

```
10
3.5
x = 10
```



# Multiple values in print

- You can also **print** multiple items separating them with commas
  - By default, a space is printed between each item

```
def main():
    x = 4
    y = 0.2
    print(x, y)
    print("x =", x, "and y =", y)
```

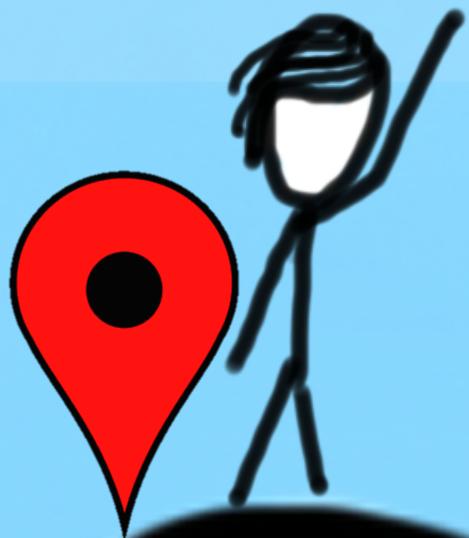
```
4 0.2
x = 4 and y = 0.2
```



You just wrote your first  
Python program and learned  
about variables!

# Today's Goal

1. Introduction to Python
2. Understanding variables



add2numbers.py