



CSCI-UA-0002

# **Intro to Computer Programming (No Prior Experience)**

## **Module 1: Variables, Statements, Etc...**

**Professor Emily Zhao**

Section 008

T/R 12:30-1:45PM

Section 012

T/R 4:55-6:10PM



## Agenda

- Classroom Agreements
- Review Questions
- **Module 1 Review**
- Pseudo-Code/Commenting
- **Introduce Assignment 1**

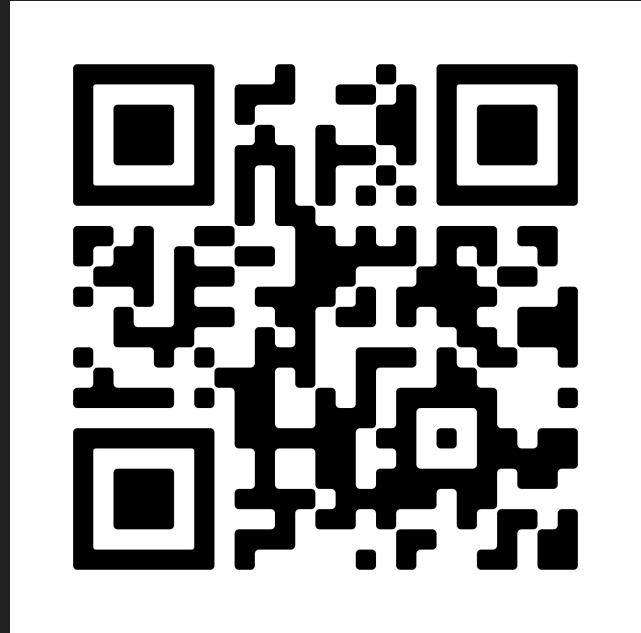
## Classroom Agreements

For both the teacher and the students:

- Be engaged and communicate your needs.
- Don't be afraid to ask for help and offer help when warranted.
- Be timely.
- Put forth your best effort.
- Practice respect and non-judgment. We come from many different backgrounds, are starting in different places, go at unique paces, and each have our own personal lives.
- One mic: one person speaks at a time. Listen without interruption.
- Any and all questions are valid – there is no such thing as a "stupid" question.

## Class Website

<http://bit.ly/python-with-emily>



## Module 1

- Setting up IDLE
- Numeric vs string literals
- Variables
- Functions + Function Calls
- The `print()` function
- The `input()` function
- Math operators

- *What are the different modes?*
- *What are the differences between them?*
- *What are the rules for naming them?*
- *What does it mean to "call" one?*
- *What is `end` and `sep`?!*
- *What does the input function return?*
- *Where do you do your math operations?*

## Additional question(s)

→ *What (the heck) is `\n`?*

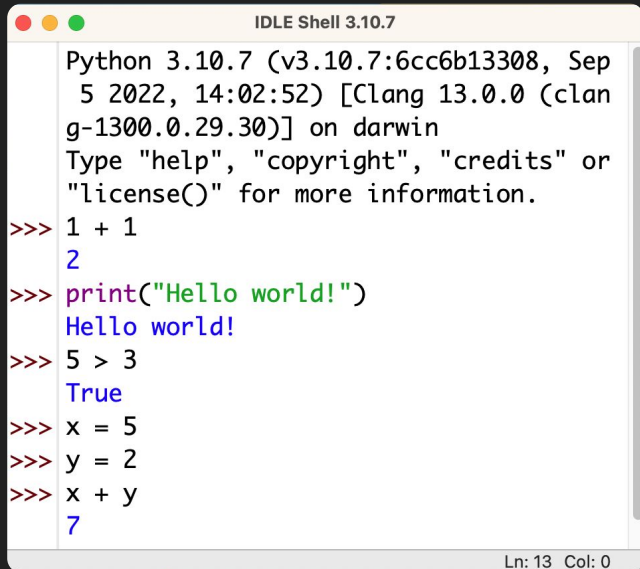
→ *What is the difference between `\` and `/`?*

# Installing Python/IDLE

# IDLE (Integrated DeveLopment Environment)

## Interactive Mode

Commands are immediately processed as they are received

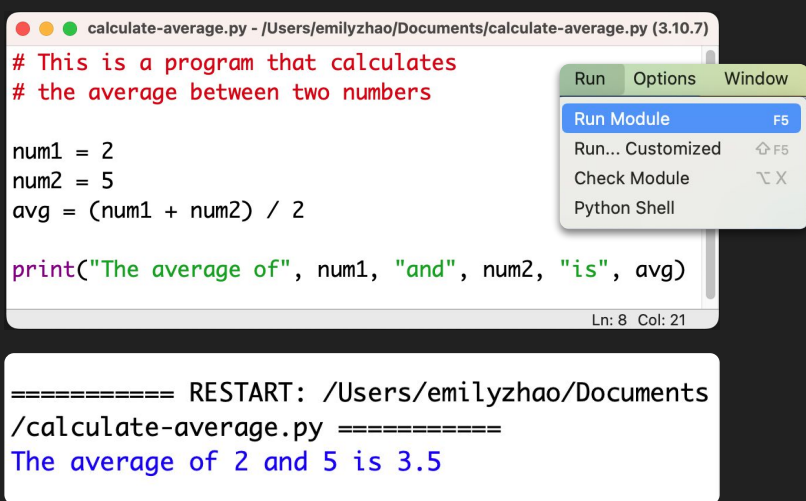


The screenshot shows the IDLE Shell 3.10.7 window. The title bar reads 'IDLE Shell 3.10.7'. The text area contains the following content: Python 3.10.7 (v3.10.7:6cc6b13308, Sep 5 2022, 14:02:52) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin. Type "help", "copyright", "credits" or "license()" for more information. Below this, several interactive commands and their outputs are shown: '>>> 1 + 1' followed by '2', '>>> print("Hello world!")' followed by 'Hello world!', '>>> 5 > 3' followed by 'True', and '>>> x = 5', '>>> y = 2', '>>> x + y' followed by '7'. The status bar at the bottom right indicates 'Ln: 13 Col: 0'.

```
Python 3.10.7 (v3.10.7:6cc6b13308, Sep 5 2022, 14:02:52) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> 1 + 1
2
>>> print("Hello world!")
Hello world!
>>> 5 > 3
True
>>> x = 5
>>> y = 2
>>> x + y
7
```

## Script Mode\*

Write a program (save as a "text file" on your computer) and run it whenever you like



The screenshot shows the IDLE Script Mode window for a file named 'calculate-average.py'. The title bar reads 'calculate-average.py - /Users/emilyzhao/Documents/calculate-average.py (3.10.7)'. The text area contains the following code: '# This is a program that calculates', '# the average between two numbers', 'num1 = 2', 'num2 = 5', 'avg = (num1 + num2) / 2', and 'print("The average of", num1, "and", num2, "is", avg)'. A context menu is open over the code, showing options: 'Run' (highlighted), 'Options', 'Window', 'Run Module' (with a keyboard shortcut 'F5'), 'Run... Customized' (with a keyboard shortcut '⇧ F5'), 'Check Module' (with a keyboard shortcut '⌘ X'), and 'Python Shell'. The status bar at the bottom right indicates 'Ln: 8 Col: 21'. Below the code editor, a separate box shows the output of running the script: '===== RESTART: /Users/emilyzhao/Documents /calculate-average.py =====' followed by 'The average of 2 and 5 is 3.5'.

```
# This is a program that calculates
# the average between two numbers

num1 = 2
num2 = 5
avg = (num1 + num2) / 2

print("The average of", num1, "and", num2, "is", avg)
```

===== RESTART: /Users/emilyzhao/Documents /calculate-average.py =====  
The average of 2 and 5 is 3.5

\* we will mostly be using script mode



## **Numeric vs String literals**

# String Literals

a sequence of characters that can contain letters, numbers, symbols and even spaces

must be enclosed in matching *delimiters*

## # Examples of valid strings

```
greeting = 'hello'
```

```
greeting2 = "hola"
```

```
greeting3 = '''bonjour'''
```

# Numeric Literals

used to represent numbers in a program (i.e. integers, floating point numbers and complex numbers)

```
# Examples of valid numeric literals
```

```
x = 5
```

```
PI = 3.14
```

# Variables

# Variables

“Buckets” that store information in your computer’s memory

```
speed = 5
```

```
name = "Emily"
```

## Naming Variables

- Can't contain spaces (can use "\_" in place) or special characters (!@#\$%^&\*)
- Can only start with a letter or underscore; can be followed by any alphanumeric character after that
- Can't use Python's "reserved" words

## Python's Reserved Words

False	None	True	and	as	assert
break	class	continue	def	del	elif
else	except	finally	for	from	global
if	import	in	is	lambda	nonlocal
not	or	pass	raise	return	try
while	with	yield			

## Legal or Illegal variable name?

```
class = 2
```

```
class_avg = 70
```

```
classAvg = 99
```

```
_class_avg = 99
```

```
2ndclassavg = 88
```

```
classavg! = 99
```



## Legal or Illegal variable name?

```
class = 2
```



`class` is reserved word

```
class_avg = 70
```



```
classAvg = 99
```



```
_class_avg = 99
```



```
2ndclassavg = 88
```



cannot start with number

```
classavg! = 99
```



alphanumeric only

## Common Variable Naming Conventions

```
rockettopspeed = 100
```

→ valid, but hard to read

```
rocket_top_speed = 100
```

→ underscored

```
rocketTopSpeed = 100
```

→ camelCase

# Functions

# Functions

You can think of functions like verbs!

- 1) They DO things
  - 2) They RETURN things
- We will learn how to write functions later in class
  - For now, we will "call" or use Python's pre-written functions

## Anatomy of a function you call

```
functionName(<arguments>)
```

```
print()
```

```
print(value, ..., sep=" ", end="\n")
```

```
input(prompt)
```

## The `print()` function

**What it does:** prints objects to the shell

**What it returns:** nothing

Defaults:

- Separates objects with a space (`sep=" "`)
- Ends each line with a new line (`end="\n"`)

## Examples of `print()`

```
print()  
# prints a new line  
  
print("hello", end="")  
print("there", end="")  
# prints "hellothere"  
  
print("hello", "there", sep="*", end="!")  
print("goodbye", "now", sep="*", end="!")  
  
# prints hello*there!goodbye*now!
```

## The `input()` function

**What it does:** asks the user for input with prompt

**What it returns:** the user input as a string

```
name = input("What's your name? ")  
print("Hello,", name, end="!")  
-----
```

```
>>> What's your name? Emily
```

```
>>> Hello, Emily!
```



# Functions

You can think of functions like verbs!

- 1) They DO things
- 2) They RETURN things

**If your function returns a value, you must store the value!**

```
>>> print_return = print()

>>> input_return = input("Enter your name: ")
Enter your name: Emily
>>> print(print_return)
None
>>> print(input_return)
Emily
```

## Escape Characters

- An “escape character” allows you to perform special actions inside the confines of a delimiter
- In Python, the escape character is `\`
- It causes Python to treat the next character as “special”

```
print('Hi, I\'m Harry Potter, a wizard.')
```

## Escape Characters

- There are a number of special characters you can use in conjunction with the escape character to perform special string operations
- `\n` forces a line break
- `\t` creates a tab

```
print ("line 1\n\tline 2\nline 3\n")
```

```
# line 1  
#   line 2  
# line 3
```

## Line Continuation

- Sometimes the code you write can get very long
- You can use the `\` symbol to indicate to Python that you would like to continue your code onto another line

```
1 print("Once upon a time, there was a king; who used to wear a single \  
2     horned crown. He had a lavish palace, three beautiful wives, \  
3     and seven children; all well qualified in their respective fields. \  
4     The king was reaching the retirement age, so he asked his elder son \  
5     to lead his empire so that he could undergo seclusion.")
```

## Programming Challenge

```
item1 = Bread  
item2 = Eggs  
price1 = $2.99  
price2 = $1.99
```

```
# Desired Output:  
# Item: Bread, Price: $2.99  
# Item: Eggs, Price: $1.99
```

## Programming Challenge: Make a Mad Lib

Write a program that asks the user to type in 4 different words using the following prompts:

- Enter a noun:
- Enter a verb:
- Enter an adjective:
- Enter an adverb:

Output the following text, replacing the blanks with the user input.

The [adjective] [noun] was very hungry so it decided to [adverb] [verb] to the nearest restaurant.

# Commenting



## **Homework**

- Self-Paced Learning Module #2 (due next class)
- Quiz #2 (due next class by 12:30PM)
- Ask a question on Ed
- Assignment #1 (due in one week)