Programming language components

ATLS 1300
Tues, Jan 21

Announcements

- Outside code
  - <= 20% of your program
- DON'T PLAGIARIZE
  - If you did not write the code, if it was not made in this class you
  - MUST COMMENT WITH WHERE YOU GOT THE CODE FROM
    - Your LA's name or my name (dates given are a plus)
    - The website URL
    - The course/program/date you made it outside the class
  - You 1. Will not get credit on the assignment, 2. May fail the class, 3.
- CHANGE YOUR CODE AND RESUBMIT IF YOU DID THIS.

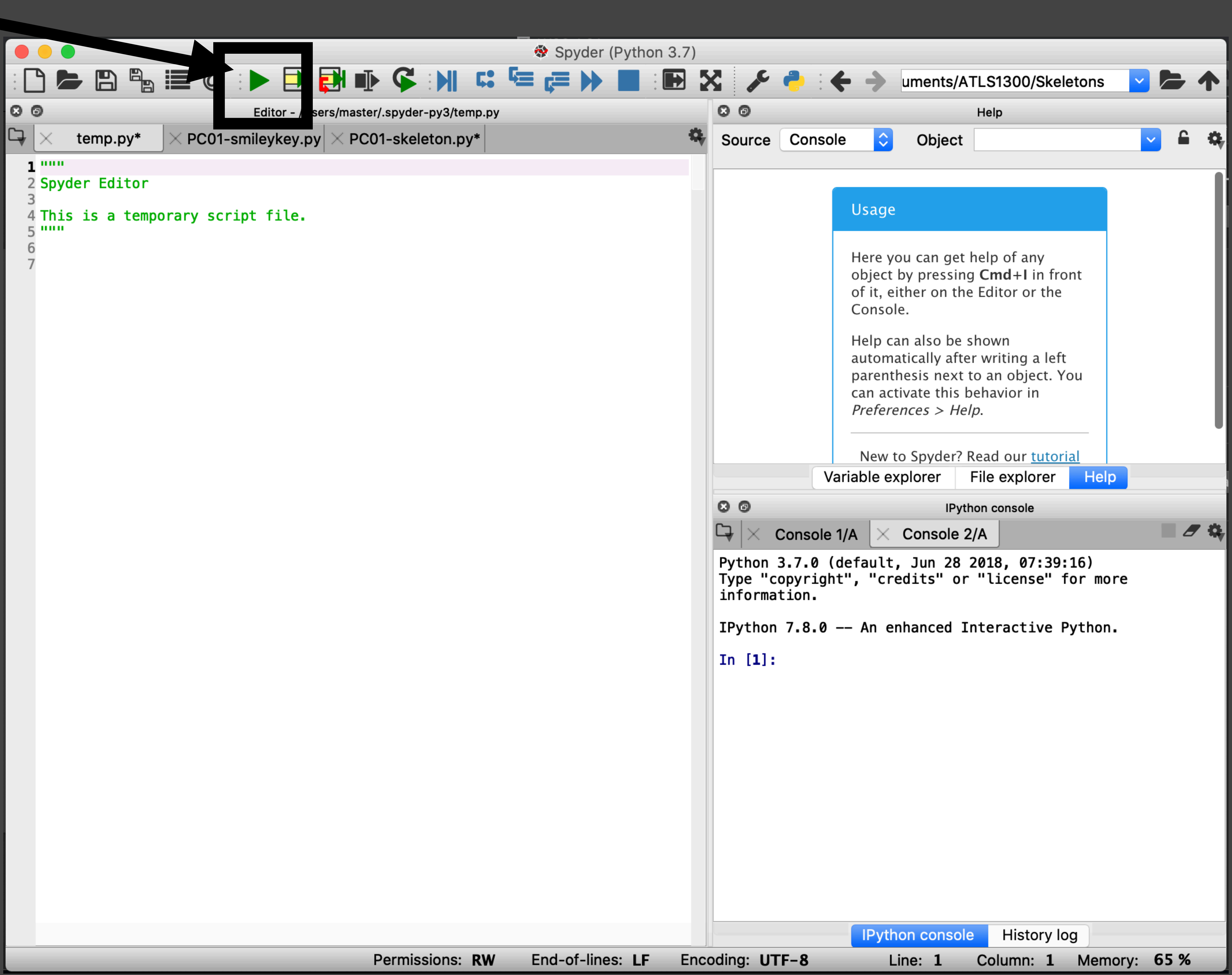
Spyder interface

Run button

Runs your code

Text area

Write your script here



```
"""
Spyder Editor

This is a temporary script file.
"""
```

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type "copyright", "credits" or "license" for more
information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]:
```

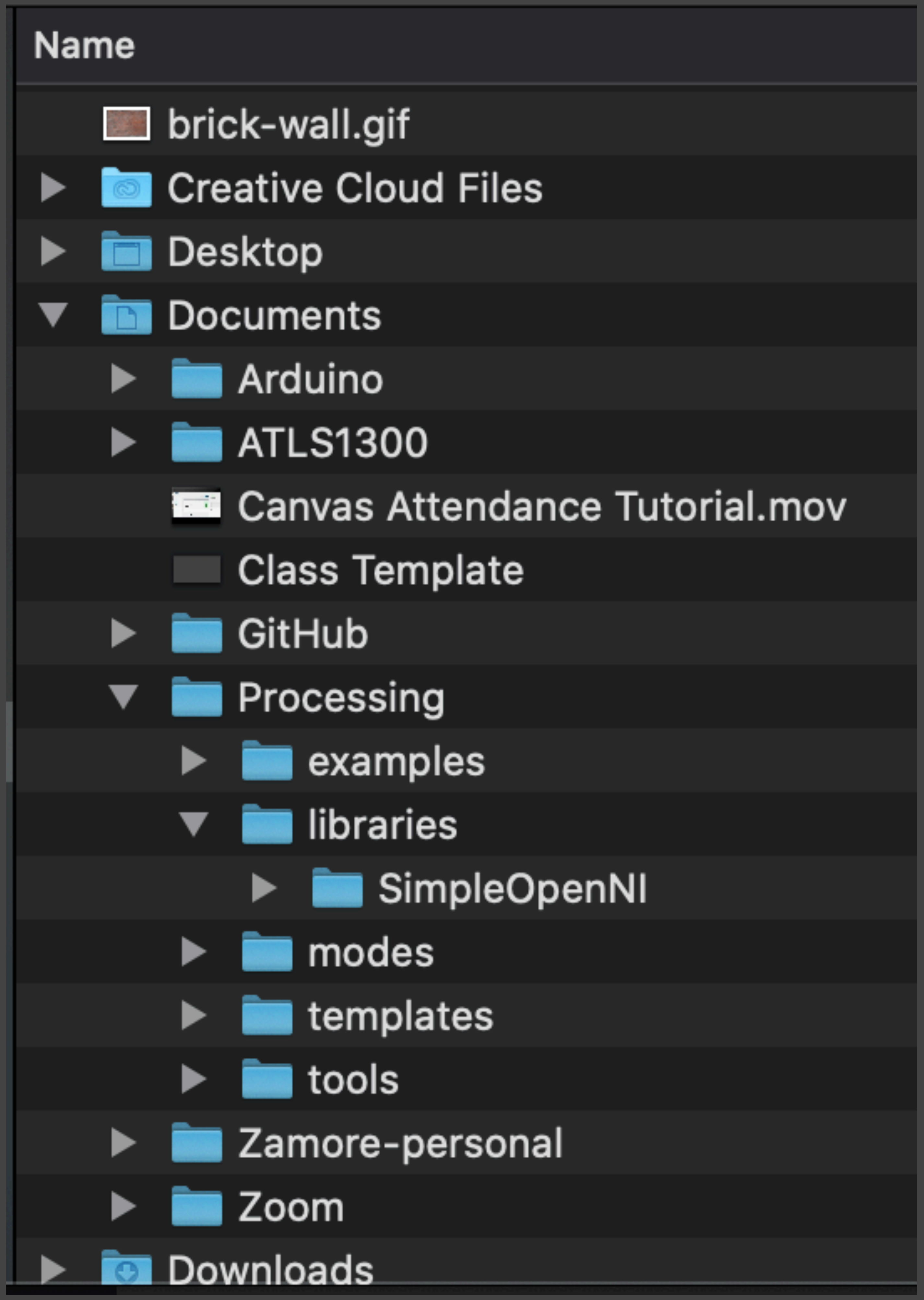
Permissions: RW   End-of-lines: LF   Encoding: UTF-8   Line: 1   Column: 1   Memory: 65 %

Command Line

Test commands, try things
Outputs the run script

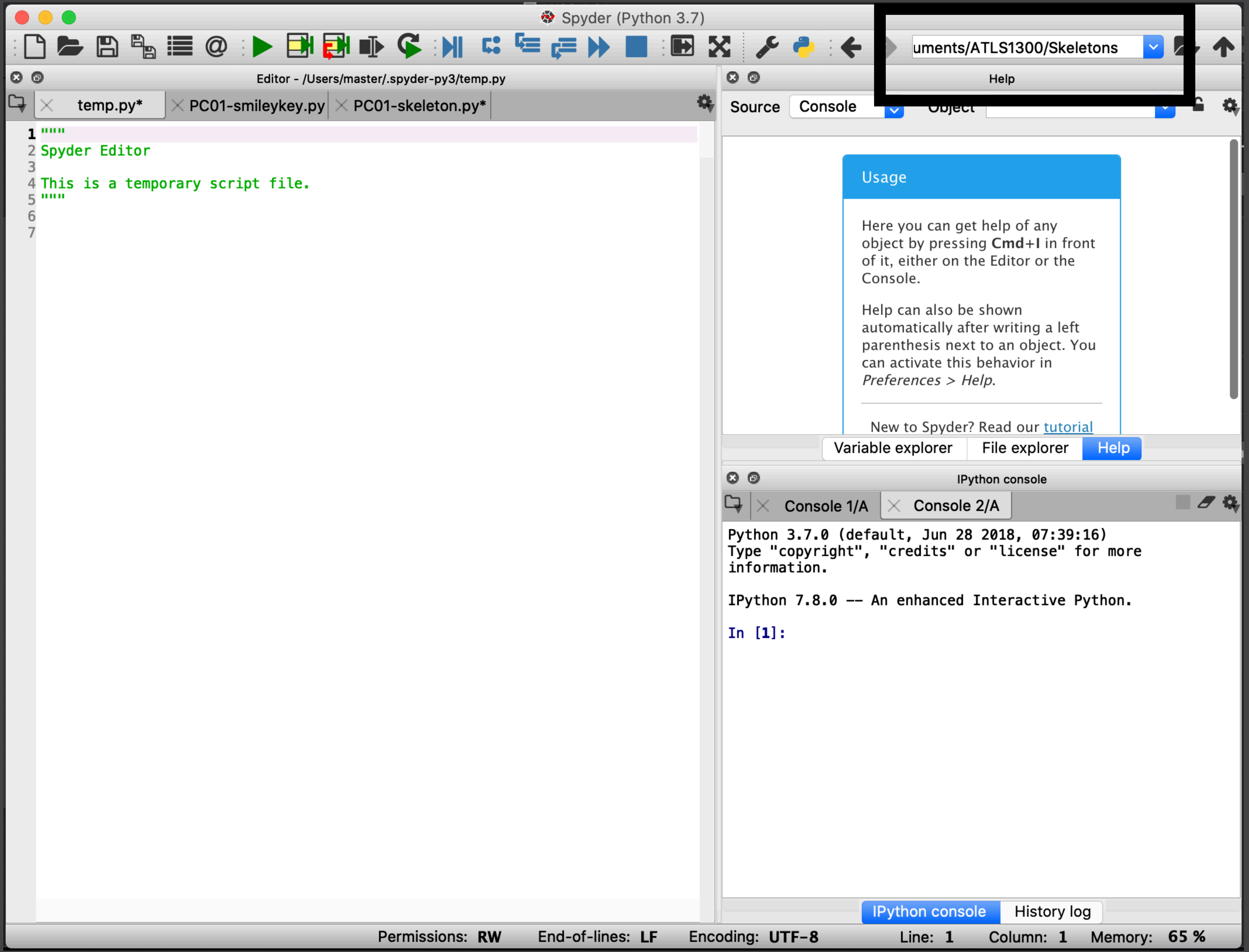
File Structures and You

- Your computer is organized in a collection of folders
- Folders can be **nested**
- You should create a folder for this class
- Inside this folder, you should create folders for EACH assignment.



```
Name
    brick-wall.gif
▶   Creative Cloud Files
▶   Desktop
▼   Documents
    ▶   Arduino
    ▶   ATLS1300
        Canvas Attendance Tutorial.mov
        Class Template
    ▶   GitHub
    ▼   Processing
        ▶   examples
        ▼   libraries
            ▶   SimpleOpenNI
        ▶   modes
        ▶   templates
        ▶   tools
    ▶   Zamore-personal
    ▶   Zoom
▶   Downloads
```

Spyder interface


Spyder (Python 3.7)

uments/ATLS1300/Skeletons

Editor - /Users/master/.spyder-py3/temp.py

temp.py* | PC01-smileykey.py | PC01-skeleton.py*

```
"""
Spyder Editor

This is a temporary script file.
"""
```

Help

Source: Console | Object

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

IPython console

Console 1/A | Console 2/A

```
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type "copyright", "credits" or "license" for more
information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]:
```

Permissions: RW   End-of-lines: LF   Encoding: UTF-8   Line: 1   Column: 1   Memory: 65 %

Working folder

Where your script will save
Where you need to put files you want your script to access

Course Trajectory

- **This week**-
  - *Tech focus*- Programming languages, common operators, variables, building programs
  - *Creative focus*- programming as a canvas
- **Next week**- More variables, debugging, and writing a program

READING (due Friday): Python Crash Course, Ch. 2 - Variables and Simple Data Types

Class Objectives

- Components of programming languages
- Breaking down **Python** like it's **English**
  - Grammar and vocabulary
  - Naming rules and conventions
- Turtles!

Programming Languages

- **Programming languages** (especially high level languages) are programs themselves
- You can implement the language using an **IDE**, sometimes this is referred to as a **programming platform**.
- Platforms (language + IDE) come with functions and tools,

Components of programming language

- Programming languages translate human desire into computer function
- **Vocabulary**
  - $\boxed{\text{nouns - variables}}$
  - **verbs - built-in functions**
  - **adverbs - keywords**
- **Grammar**
  - syntax - rules for how the program is written

Data Types

- **Variables** are named containers for storing data values
  - **Integer** - `int` - whole number values

```
1, 2, 8, 1e10
```

  - **Float** - `float` - floating point numbers,

```
9., 3.14, 83261429019.777777777777778
```

    - no fixed number of digits before or after decimal point

Nouns

Data Types

- **Variables** are named containers for storing data values

  - **Boolean** - `bool` - Binary logic variable

    `True, False`

  - **String** - `str` - sequence of characters

    `“fleas”, ‘a’, ‘3’`

Creating variables in Python

- **Variables** - stored, named data
- Data are values
- Name is whatever you want to call your variable. Go crazy!
- Variables are assigned **left to right**

Name		**Value**
`A`	`=`	`3**2`
`name`	`=`	`“Dr. Z”`
`date`	`=`	`200121`
`imaBool`	`=`	`False`

Create some variables!

- Open Spyder
- Create variables of different data types:
    - `Float (floating point numbers)`
    - `Bool (True or False values)`
    - `String (Text inside quotes)`

Vocabulary

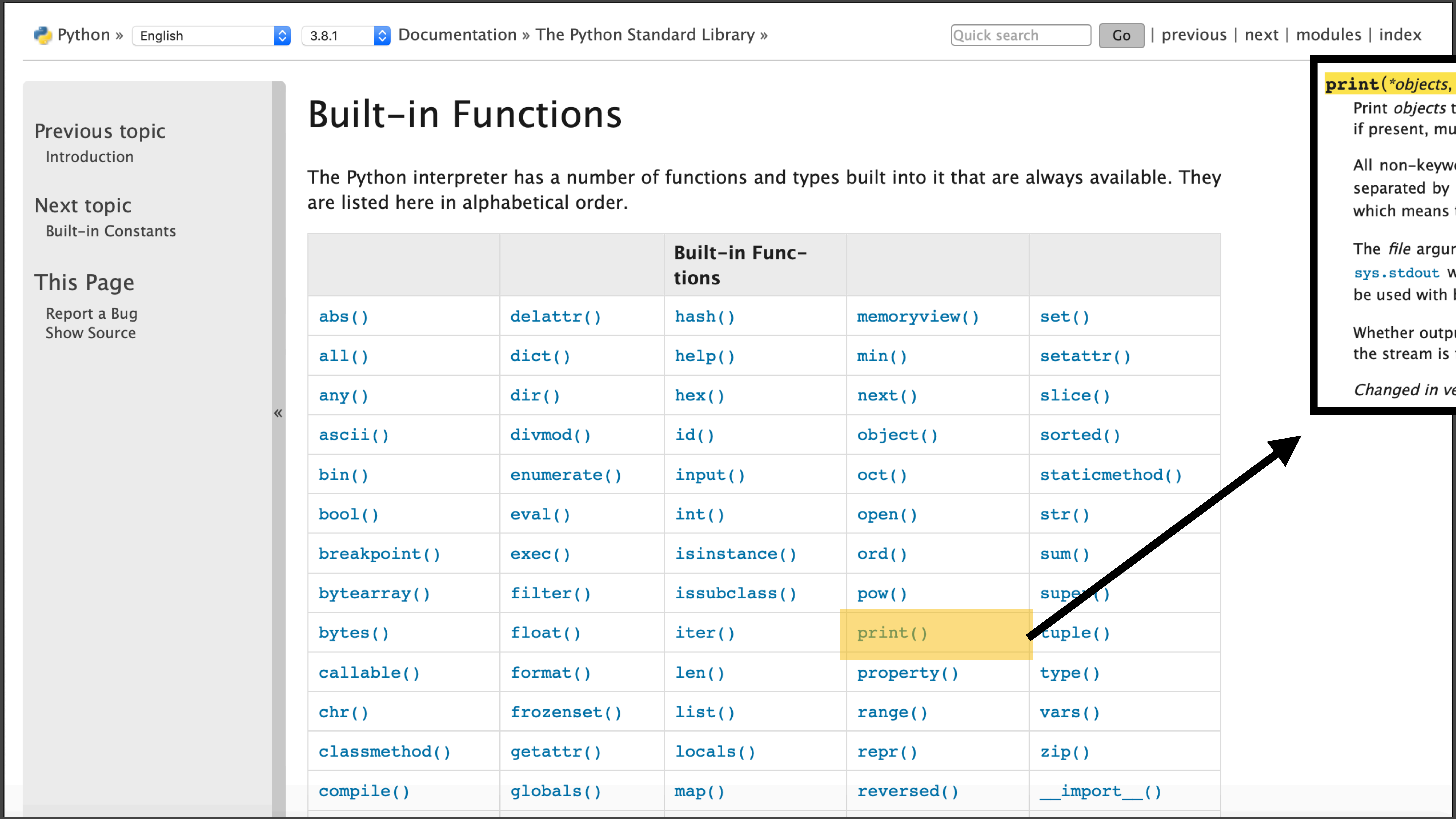
- **Variables**
  - Named containers for storing data values
- **Keywords and functions**
  - Do basic tasks, like provide a user input
  - Retrieve more programs, run functions from them
  - Build more complex or elegant tasks

Components of programming language

- Programming languages translate human desire into computer function
- Vocabulary
  - **nouns** - variables
  - **verbs - built-in functions**
  - **adverbs** - keywords
- Grammar
  - syntax - rules for how the program is written

Built-in functions	Keywords
`input()`	`True`
`print()`	`False`
`str()`	`None`
`bool()`	`del`
`int()`	`if, elif, else`
`float()`	`for, while`

More info in Documentation



Python » English 3.8.1 Documentation » The Python Standard Library »

Quick search Go | previous | next | modules | index

Previous topic
Introduction

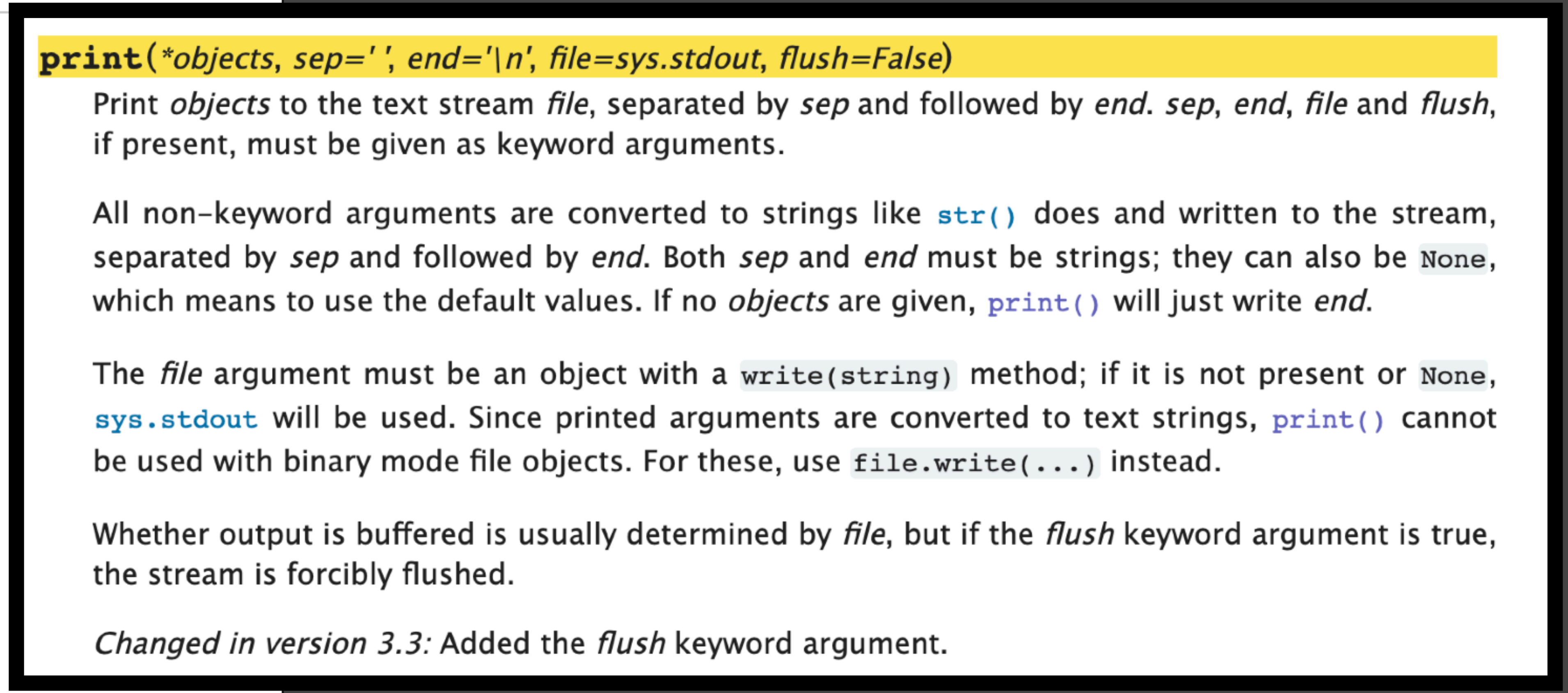
Next topic
Built-in Constants

This Page
Report a Bug
Show Source

Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()



print(**objects, sep=' ', end='\n', file=sys.stdout, flush=False*)

Print *objects* to the text stream *file*, separated by *sep* and followed by *end*. *sep*, *end*, *file* and *flush*, if present, must be given as keyword arguments.

All non-keyword arguments are converted to strings like `str()` does and written to the stream, separated by *sep* and followed by *end*. Both *sep* and *end* must be strings; they can also be `None`, which means to use the default values. If no *objects* are given, `print()` will just write *end*.

The *file* argument must be an object with a `write(string)` method; if it is not present or `None`, `sys.stdout` will be used. Since printed arguments are converted to text strings, `print()` cannot be used with binary mode file objects. For these, use `file.write(...)` instead.

Whether output is buffered is usually determined by *file*, but if the *flush* keyword argument is true, the stream is forcibly flushed.

Changed in version 3.3: Added the *flush* keyword argument.

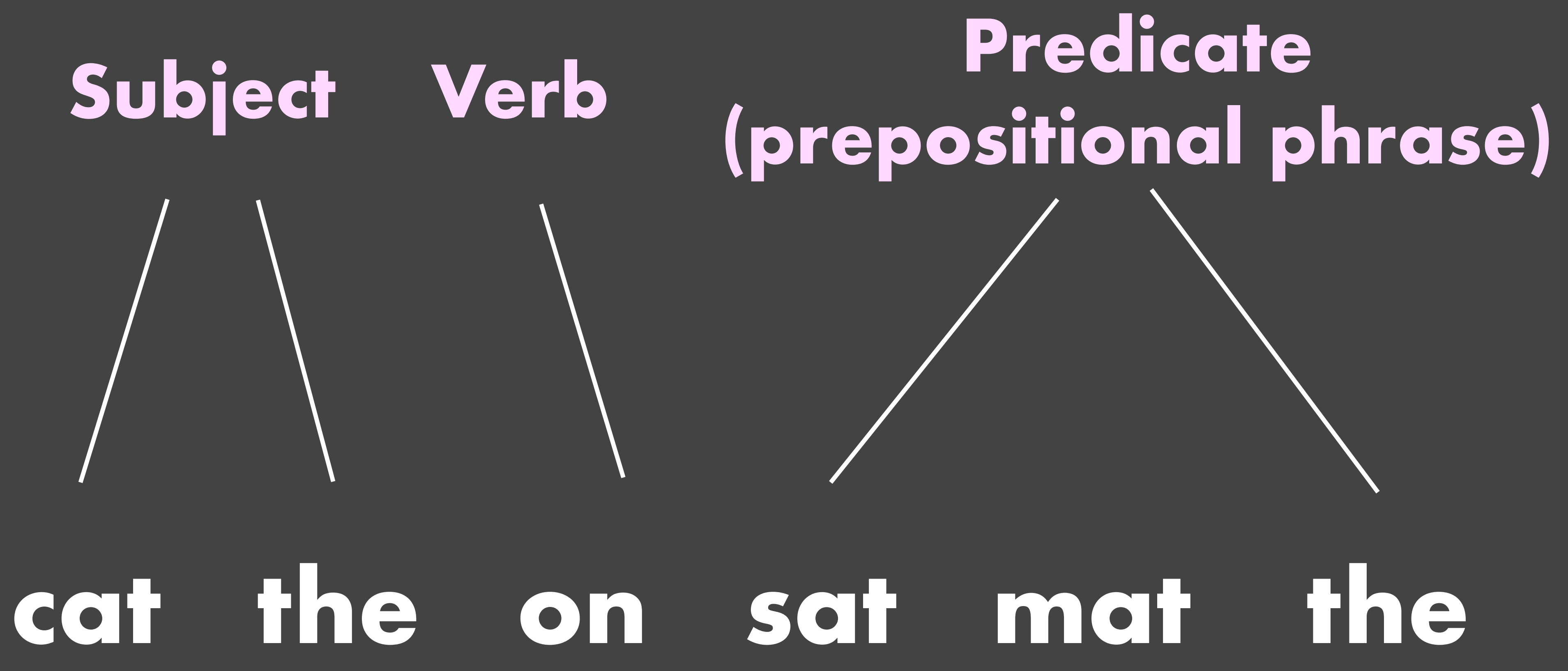
Linked in this week's module on Canvas

Components of programming language

- Programming languages translate human desire into computer function
- Vocabulary
  - **nouns** - **variables**
  - **verbs - built-in functions**
  - **adverbs** - **keywords**
- Grammar
  - **syntax** - rules for how the program is written

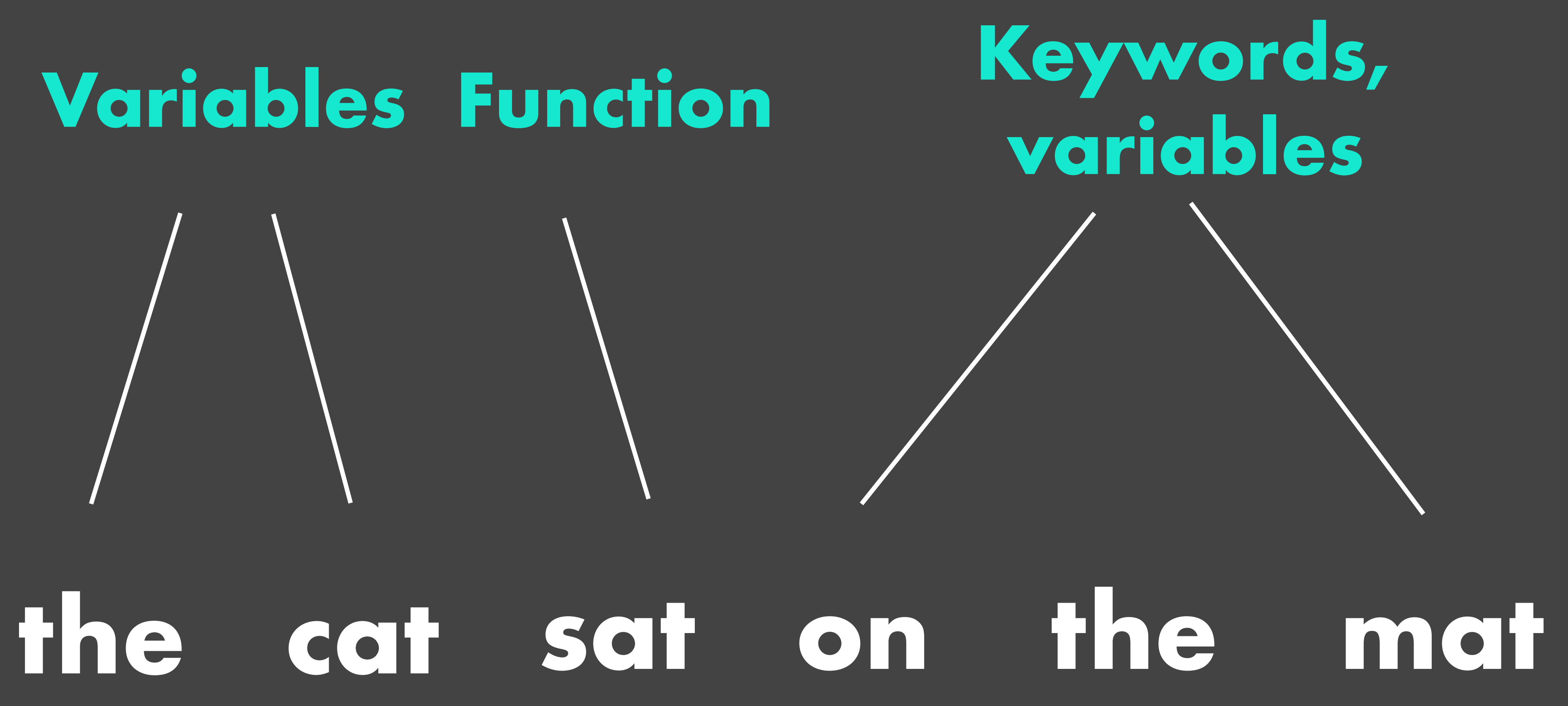
Syntax

The order and format that values, variables, keywords and functions have to be presented



Syntax

The order and format that values, variables, keywords and functions have to be presented



Python has a syntax just like any language!

Python Syntax

Comments marked with #

```
# I’m a comment!
```

Call functions, group math with **parentheses**

```
2 /  (6 + 4)

print (‘yo waddup’)
```

Indent with loops (more on loops later)

```
if 0 < 1:

     print(‘Mathematical!’)
```

Python syntax

Case sensitive

Enter these lines in your command line:

```
Name = 0

name
```

Indentation matters!

- This will come up more when we get to loops…

Exploring syntax with turtles

In your command line type:

```
from turtle import *

Turtle()
```

Imports a set of functions for you to use

What happens?

How would you set the output of the turtle command (`Turtle()`) to a name (like `Ari`)?

Hint: remember ***left-to-right notation***

Exploring syntax with turtles

- Now let’s make things interesting.
- Our turtle, `Ari`, has a bunch of functions attached to it.
- Change the arrow shape to a turtle by using the command shape(). To call this command, we’ll use dot notation

1. Try:
   `Ari.shape(‘turtle’)`
   - You can also input: `“circle”, “square”, “triangle”`, or `“classic”`
2. Now try the entering the command `forward(x)`, where x is the numerical distance you want the turtle to go in pixels.

- Which turtle moved? How do you get the other turtle to move?

Recap

- **Programming languages** and **IDEs** are both examples of _________.
- Languages are made up of vocabulary and grammar. In programming:
  - Vocabulary is _________, ________, and ________.
  - Grammar is __________, the rules that determine how the program is written.
- Python variables are assigned **left to right**, with the _______ on the left, and the ________ on the right.
- Functions are called using ____________.

Thursday

- Documentation
- Composition & color
- Drawing with turtles
- Look up creative coding art (#creativecoding, #computationalthinking)

READ CH 2!