Computer Organization

Kelly Rivers and Stephanie Rosenthal

15-110 Fall 2019

Last Time: Programming Basics

Programming is an instantiation of algorithms

Types: Integer, Float, String, Boolean
Variables
Syntax
Comments

Debugging – check for wrong answers and errors
- TypeError
- NameError
- IndentationError
- SyntaxError

Today's Learning Objectives

To understand how computers organize files and to be able to find files on your computer

To describe the process the computer performs to take abstract Python code and translate it into instructions the computer can execute

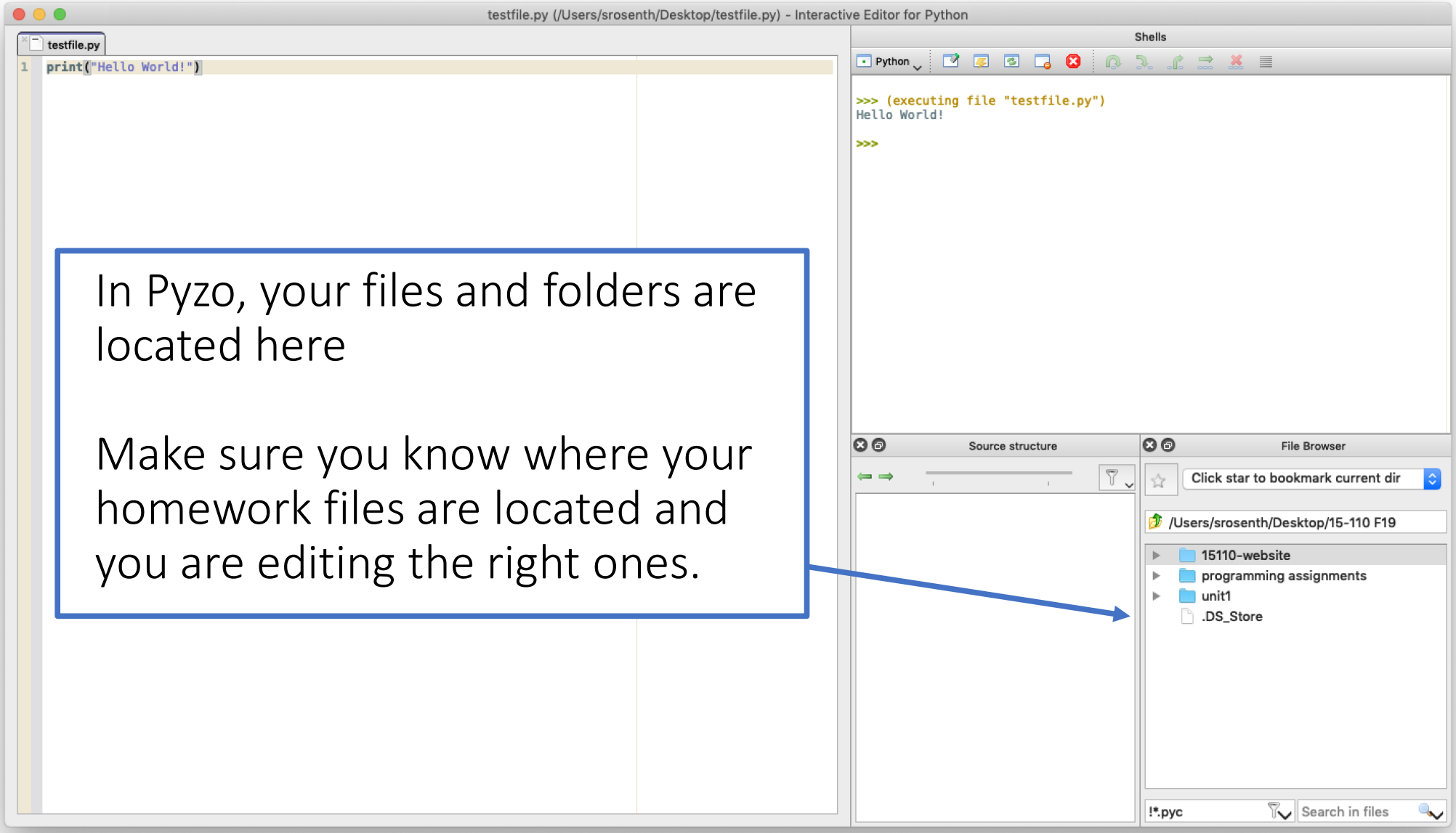
File Organization

Computers allow you to organize your data into files and folders

Files are the content that you or the computer creates/downloads/etc
documents, slides, spreadsheets, pdfs, python code, etc

Folders are the way to organize those files into meaningful sets
Folders can hold other folders or other files or both
A 15-110 folder could hold the syllabus and folders for slides,
and each of the homeworks
The folder of slides would hold all of the powerpoints
The folder of homeworks could hold a folder for each homework .py file

File Organization



In Pyzo, your files and folders are located here

Make sure you know where your homework files are located and you are editing the right ones.

Python as an Abstraction

A python .py file is a text file of the code you wrote

Python code and all human-readable code is an abstraction of what the computer can do

We don't tell the computer where to find the number 5 in memory
We don't tell the computer which points on the screen to print
We just tell it we need the number 5 and we need to print, and the Interpreter does the translation to tell the computer to do it

Python Interpreter

When you have Python3 loaded and run that file, the computer:

- opens an Interpreter that checks the file's syntax (whether it is valid Python code)
- translates that human-readable code into new bytecode that tells the computer exactly what to do
- tells the computer to run that bytecode sequentially (line by line)

Python Interpreter

When you have Python3 loaded and run that file, the computer:

- opens an Interpreter that checks the file's syntax (whether it is valid Python code)
- translates that human-readable code into new bytecode that tells the computer exactly what to do
- tells the computer to run that bytecode sequentially (line by line)

The Interpreter translates between something we understand and something the computer understands

In order to translate, the Interpreter defines the syntax of the language – the rules for how we should write code that it understands

Interpreter's Job: Tokenizing and Parsing Syntax

The Interpreter has to figure out what to do with each letter you write.
Is it part of a literal? A variable? A function? A keyword?

Tokenizing – splitting the text into meaningful chunks (e.g., words in English)

Parsing – do those tokens make sense in the order provided? Is there a meaning to the order of those tokens?

If the Interpreter cannot figure out how to tokenize or parse your file, it produces a SyntaxError (including IndentationError and EOL Error)

Examples of Errors using PythonBuddy.com

```
print(Hello)
```
NameError

```
print("Hello')
```
EOL literal (missing quotation)

```
   5/3
```
IndentationError: unexpected indentation

Interpreter Produces Bytecode

If the Interpreter can tokenize and parse all of the code, it can automatically create new code called **bytecode** that is more specific for the computer

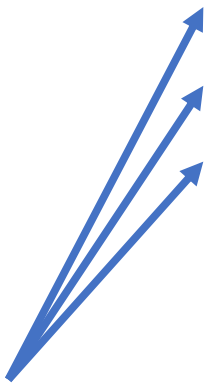
In the bytecode, there are:

a designated **set of instructions** that the computer knows how to run
(more info: https://docs.python.org/3/library/dis.html#bytecodes)

a table of all of the **literals** the code will use

a table of all of the **variables** the code will use

a table of all of the built in and defined **functions** (if any)
and what line of code they start on



Program Memory

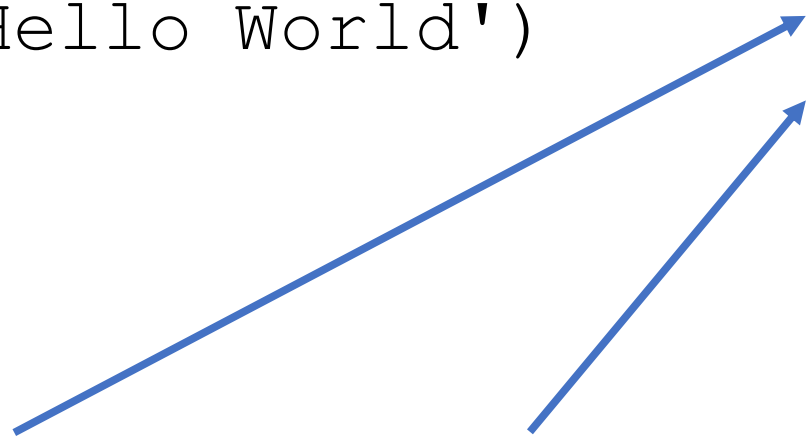
Python and Bytecode Example

Python:

```
print('Hello World')
```

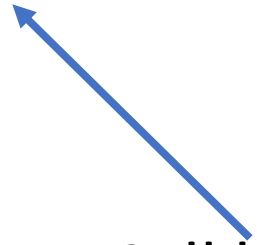
Bytecode:

```
LOAD_GLOBAL              0
LOAD_CONST               1
CALL_FUNCTION            1
```



Global Function **print** in Function Table location 0

Literal **'Hello World'** in Literals Table location 1



Call loaded print function with 1 constant **"Hello World"**

Python and Bytecode Example

Python:

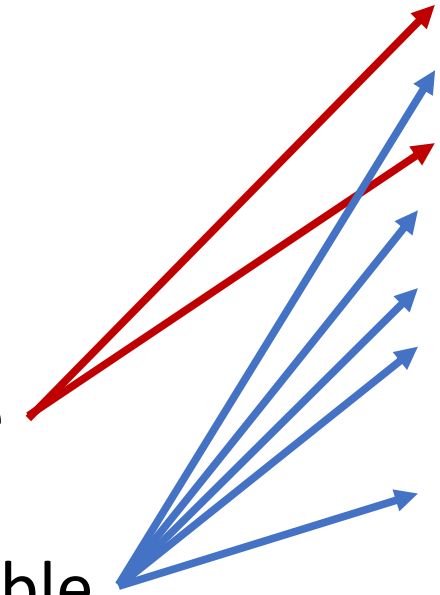
```
x = 5
y = 7
z = x+y
```

Bytecode:

```
LOAD_CONST          1  (5)
STORE_FAST          0  (x)
LOAD_CONST          2  (7)
STORE_FAST          1  (y)
LOAD_FAST           0  (x)
LOAD_FAST           1  (y)
BINARY_ADD
STORE_FAST          2  (z)
```

Literals in Literals Table

Variables in Variable Table



Memory and Bytecode

More about memory and representing information next week!

Interpreter and Errors

The Interpreter only finds Syntax Errors – errors in the python language rules

What about TypeErrors?

`print(5/3+" ")` unsupported operand for +: 'float' and 'str'

This code follows the rules, as long as 5/3 and " " are the same type

Python does not check types until the code runs

Runtime Errors – the code must run to find the error (e.g., TypeErrors)

Other errors?

Computers only do what you tell them

If you make an error in how the program works (use < instead of >) it is syntactically correct and has no runtime errors

Logical Errors – there is a problem with the code algorithm

High Level Concepts from Today

Computers allow you to store information in files and organize that information using folders

Python .py files are text files that the computer does not know how to run directly because the language is too abstract

Python has an interpreter that tokenizes and parses the text in order to check the syntax. It produces SyntaxErrors if it cannot tokenize/parse.

If the interpreter succeeds, then it generates bytecode automatically.

Syntax Error vs Runtime Error vs Logical Error

Announcements

- Get Python3 and Pyzo running on your computer
- Waitlist – Keep coming to class. We'll work with you to find space. Talk to us about access to the check-in assignment.
- Homework 1 Check-in due on Monday 9/2 noon
- Homework 1 is due Monday at 9/9 noon!

You should be able to do all of the check-in problems and more than half of the homework problems! Start early on both. Use office hours!