Welcome to Class

Python for Economics
Morning, Aug 16, 2023
Richard Lawrence, Wesley Brashear, Zhenhua He, Josh Winchell
Course Outline

0. Welcome
1. Introduction
2. Elements of Code
3. Control Structures
4. Data Structures

5. Numpy
6. Matplotlib
7. Pandas
8. APIs

Today

Tomorrow
Learn Python

- Python has many topics.
- They build on each other.
Not all of Python

- A goal of this workshop is to do a project.
- We will learn only a few topics to make that possible.
Google Classroom

- Our teaching materials are hosted in a Google Classroom.
- Check your email for an invitation to classroom.google.com
- Log in with your TAMU Google Account.
Interactive Learning

- The Jupyter Notebook is a file you can edit.
- Colaboratory (provided Free from Google) is the editor.
- In a Notebook, you can try out new code right away!
Google Colaboratory

- Please try it now to make sure it works with your browser.
- Navigate to colab.research.google.com
- Log in with your TAMU Google account
Introduction to Python

Python for Economics
Morning, Aug 16, 2023
Richard Lawrence
What Computers Don’t

- Computers don’t think. They only follow instructions.
- Computers don’t understand what you want. They do what you say, literally.

Disney’s *Fantasia* (1940)
Where to get Python

- Windows, Mac, and Linux all have it in their app stores (for free)
- Python is also **online** from Google and others
  - [https://colab.research.google.com/](https://colab.research.google.com/)
  - [https://replit.com/](https://replit.com/)

*Note: stay away from Python 2 because Python 3 is much better.*
Starting Programming

Let’s get started with:

- Getting familiar with notebooks and Google Colab
- Writing your first Python program

→ Go to notebook to practice
Elements of Code

Python for Economics
Morning, Aug 16, 2023
Wesley Brashears, Richard Lawrence
This Module

1. Comments
2. Data Types
3. Operators
4. Variables
5. Functions
6. Tuples
7. Multi-line Statements
Some Basics

- **Comments:**
  Sometimes you need to write stuff that's not executed: annotations, documentation, or code that isn't used.

- **Data Types:**
  Different kinds of data are treated differently in Python. For example, you can do math with numbers, but you cannot do math with text. Python keeps track of which data “types” you’re working with to let you know if you can actually use them together.

→ Go to notebook to practice
Break Time Reminder Slide

10 minutes break
Operators: Math and Logic in Python

In Python you will see some of the same symbols you know from math (+, -, <, >, etc) and use them in some of the same ways:

- Integer arithmetic
- Order of operations
- Comparisons
- Logic

...But exactly what they do can depend on the data types involved.

→ Go to notebook to practice
Computers are able to store data in memory (a hardware component). This data is stored in the form of binary bits (0s and 1s) and each location in memory has an address which is also in the form of binary bits.

A programming language allows the programmer to refer to the data by a *label* and think abstractly. This is called a *Variable*. It is closely related to the math concept with the same name, but

\[ x = 5 \]

means something slightly different between math and programming.

→ Go to notebook to practice
10 minutes break
Functions: Reusing Code

- Code in general is predictable, so a program executed more than once will usually result in the same outcome.

- Of course, the results might change if we start with different data, but we very well might want to apply the same code to that new data.

- Separation of code and data makes code reusable. Reusable code is packaged as “functions”.

→ Go to notebook to practice
Tuples and Multi-line Statements

- Tuples let you work with multiple values at once!
- Parentheses (or brackets or braces) let you split up code over multiple lines.

```python
a, b, c = 1, 2, 3
print("a equals ", a, 
      "b equals ", b, 
      "c equals ", c)
```

- Go to notebook to practice
Lunch Break Reminder Slide

1 hour break
Control Structures

Python for Economics
Afternoon, Aug 16, 2023
Wesley Brashear
This Module

1. Indentation
2. FOR Loops
The amount of whitespace at the beginning a line is called the indentation.

Other programming languages often ignore whitespace...

*But Python does not! Whitespace is important!*

Common indentation levels: 2 spaces, 4 spaces, 8 spaces, etc

**Warning:** *Spaces and tabs* are both whitespace, but tabs don't look the same in every text editor so it can be a "gotcha".
Indentation: Blocks

In Python, programs are structured into blocks. A block is a group of statements that are executed together.

Statements in a block have the same indentation.

```
block 1
block 1
    block 2
    block 2
```
Indentation: Nested Blocks

Blocks can contain further blocks with greater indentation

Example (left):
- All the statements with no indentation are part of the main block (block 1)
- block 1 contains all the other blocks
Blocks can contain further blocks with greater indentation.

Example (left):

- **four** lines are part of block 2 because they're separated from each other by statements with *greater* indentation (blocks 3 and 4).
Indentation: Nested Blocks

Blocks can contain further blocks with greater indentation

Example (left):
- block 3 and block 4 are different blocks because they're separated by a statement with less indentation (block 2).
A block can be executed once, multiple times, or not at all.

A **control statement** determines when, why, and how this occurs.

Control statements *precede* the block and end in a colon "::":

```
block 1
block 1
control statement:
  block 2
  block 2
  control statement:
    block 3
    control statement:
      block 4
      block 2
block 1
```
FOR Loops: Anatomy of a Control Structure

We have already seen the `for` statement. This is an example of a control structure.

```python
for x in range():
    print()
```

Observations

- The `for` control statement ends with a colon `:`
- The next line is indented (some amount of space on the left)
The order in which statements are executed is called Flow. Control statements determine where flow goes next.

Each control statement can either

- send flow into its block
- pass to the statement after its block.

When flow reaches the end of a block, it returns to the control statement above that block.

FOR loops repeat their block a predetermined number of times.
Break Time Reminder Slide

10 minutes break
Nested Flow Control Diagram Example

block 1
block 1
control statement:
  block 2
  block 2
  control statement:
    block 3
  control statement:
    block 4
  block 2
block 1
Data Structures

Python for Economics
Afternoon, Aug 16, 2023
Josh Winchell
This Module

1. Lists and Dictionaries
2. Index and Key
3. Slicing
4. Methods
The Need for Data Structures

Say we have some data that should all be together—maybe in some order. Creating a bunch of individual variables would be a pain. Let’s instead use data structures.

Data structure means applying a label scheme to a group of data elements.

```
word1 = "once"
word2 = "upon"
word3 = "a"
word4 = "time"
```
Lists and Dictionaries

A **list** can store multiple values of any type:

```python
story = ["Chapter", 1, "Once", "upon", "a", "time"]
print(story)

['Chapter', 1, 'Once', 'upon', 'a', 'time']
```

A **dictionary** stores values (again of any type) with specified “keys”:

```python
story = {
    "Title": "Snow White",
    "Chapter": 1,
    "Text": "Once upon a time"
}
print(story)

{'Title': 'Snow White', 'Chapter': 1, 'Text': 'Once upon a time'}
```

→ Go to notebook to practice
Index and Key

Once we have data inside our data structures, we need to be able to access specific pieces of it:

- Lists and strings have an index (integers, starting at 0):
  
  \[ \text{story[0]} = \text{"Once"} \]

- Dictionaries have keys (multiple possible data types):
  
  \[ \text{story["first"] = \text{"Once"}} \]

→ Go to notebook to practice
Break Time Reminder Slide

10 minutes break
Slicing

We may want to get a range of values from a string or list:

```python
my_string = "eenie meenie minie moe"
print(my_string[6:18])

my_list = ["eenie", "meenie", "minie", "moe"]
print(my_list[1:3])

meenie minie
['meenie', 'minie']
```

→ Go to notebook to practice
Methods

Data structures (and other things) have their own built-in functions called *methods:*

```python
my_list = ["gizmo","whatchamacallit","doodad","thingamajig","maguffin"]
my_list.sort()
print(my_list)

['doodad', 'gizmo', 'maguffin', 'thingamajig', 'whatchamacallit']
```

→ Go to notebook to practice
End of Day!

Let us know if you have questions