Learn Python

Learning Python has been something I've postponed over the years. I've run scripts here and there and created small scripts but nothing serious. I'm dealing with a scenario where I need to expand my knowledge of it. I'm excited about the opportunity, so here we go!

This article assumes a working knowledge of the command line and at least one programming language.

The Basics

Python is an object-oriented, interpreted, and easy to learn language. The syntax is concise and approachable to beginners. I've used it in the past off and on and

have found myself saying, “That’s it? Wow!”. You’ll need to install python to follow along.

Similar to Ruby, there is a pyenv utility for managing your python versions. It’s best to leave your system Python versions alone to avoid trouble with system dependencies.

```
#!/usr/bin/env python3
import pandas as pd

filename = "example.csv"
print(f"Chunking records for {filename}")
print("\n")
chunksize = 3
for chunk in pd.read_csv(filename, chunksize=chunksize):
    print(chunk)
```

Running this code requires a little more work. You’ll need to install pip so you can install the pandas library for data analysis tasks. I’m on macOS, so I need to run this command.

```
python3 -m pip install --user --upgrade pip
```

I need to install the pandas package.

```
pip3 install pandas
```

Download these files to run the code above and put them in the same directory.

example.csv & analyze-csv.py & .gitignore

Run this command in your terminal to make the program executable.

```
chmod +x ./analyze-csv.py
```

Run this command to list all your files.

```
ls -w
```

You should see output like this.

```
$ ls -w
analyze-csv.py          example.csv
```

Now execute the file.

```
./analyze-csv.py
```

You should see the following output.

```
$ ./analyze-csv.py
          store|sales
Office A            7
Office B            3
Office C            9
          store|sales
Office D          100
Office E            4
Office F           96
          store|sales
Office G           56
```

```
Office H           34
Office I           37
          store|sales
```

It didn't take much to create a program that can process CSV files in chunks.

Virtual Environments

I've learned that using a virtual environment is preferred over using the operating system's interpreter. The modern method to create a virtual environment is to use the venv package. Run this command in the directory you downloaded the **analyze-csv.py** file to. This will initialize your virtual environment.

```
python3 -m venv venv/
```

I ran into issues running this on a network drive. If the command hangs, try moving your folder to a physical disk.

Run this command to activate the environment.

```
source venv/bin/activate
```

You should see a shell prompt like this if it was successful. Note the (env) text indicating that the virtual environment is activated.

```
(venv) Jeffs-Laptop:python jeffbailey$
```

Now your program is isolated from the system. We can rest easy with the

knowledge that our program will more reliably run in a virtual environment. Virtual environments save users of your program headaches. Users are not required to align their package versions with your required packages.

To leave the virtual environment run this command in your directory.

```
deactivate
```

Packages

Now that we have a virtual environment, we can install the pandas package but this time in the virtual environment.

```
pip install pandas
```

We can run this nifty command to create a requirements.txt file. This file tells users of our program what package versions to use.

```
pip freeze > requirements.txt
```

The freeze command will create a file that contains the dependencies you'll need to run the **analyze-csv.py** file. Run this command to view the contents.

```
cat requirements.txt
```

You see an output like this.

```
$ cat requirements.txt
numpy==1.18.4
pandas==1.0.3
python-dateutil==2.8.1
pytz==2020.1
six==1.15.0
```

The pip freeze command lists all packages installed on your system. We've now defined which dependencies are required for the program to run. Users of your application can run this command to install the necessary modules.

```
pip install -r requirements.txt
```

The virtual environment will keep us from having to use our system-wide installation of python, which may have *issues.*

Virtual environments will make our programs happy little clams.

Pushing It Up

If you like GitHub, you can create a GitHub repository called **learn-python** and push your new code to it. If you're in a hurry and already know all this stuff, you can clone my repository on GitHub and play around with it.

Run these commands to initialize a new repository and get it ready for the push.

```
git add . && git commit -am "Initial Commit"
```

Now let's push to our newly created repository. You'll want to replace my username with your GitHub username.

```
git remote add origin git@github.com:jeffabailey/learn-python.git
git push -u origin master
```

And that's that, you now have a program you can extend and have fun learning with.

Fundamentals

Learn Python without considering the fundamental programming constructs of Python? That's crazy talk.

Variables

Variable assignment is simple enough, no magic here, just the way I like it.

The code is following the naming section of Google's style guide. I am using code in the style guide source code as an example. Snake case variable naming is common when looking at Python code.

Comments

Comments for sanity. Tell your future self and your friends why you did something weird once upon a time.

Control Structures

We saw a for loop in our CSV analysis program. Here’s another example.

A while loop

An if else statement

Functions

Functions are the bread and butter of any programming language. I also threw in an f-string, which allows you to format your strings with variables elegantly.

Classes and Objects

Classes and objects are the nuts and bolts.

Lambdas

Lambdas smooth out the rough edges. Use them sparingly, most Python programmers are not a fan of them. Consider using a function rather than a lambda in more complex situations.

Exception Handling

Exception handling to keep from angering the user gods. Keep your errors under control and handle the flow.

Arrays

Arrays for life. Python arrays don’t contain any surprises, which makes them easy to use.

Operators

Operators for operating stuff.

Most Python operators are standard but watch out for type comparisons. Type comparisons with operators will compare the type names rather than the types themselves. Use the **isInstance** built in function to compare types instead.

There are many other constructs to learn in Python. It’s a joy to work with Python thanks to its natural and straightforward behavior.

Learn Python Beyond the Basics

There are thousands of Python books and loads of free tutorials online. Here are a couple of noteworthy titles that will help you learn effectively. I’ll leave you with

The Zen of Python.

- **Books**
  - Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Python (O'Reilly)
  - Learning Python, 5th Edition (O'Reilly)
  - Python Cookbook, Third edition (O'Reilly)
  - Python Crash Course (Eric Matthes)
  - Head First Python: A Brain-Friendly Guide (Head First Books)
  - Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code (Zed Shaw's Hard Way Series)
- **Videos**
  - Learning Python (LinkedIn)
  - Python Essential Training (LinkedIn)
  - Python Best Practices for Code Quality (Pluralsight)
  - Complete Python Developer in 2020: Zero to Mastery (Udemy)