

Style and Style Guides

What is the difference between the two code snippets to the right? Functionally they're the same, but they have very different **style**.

Code style refers to how code is designed to make it **readable** and **maintainable**. It's about the parts of code that matter to humans, not the Python interpreter. Style includes aspects like comments, how variables are named, spacing, and more.

Python has an official style guide – [PEP 8](#) – but organizations can have their own style guides which may work differently. The particular style that is chosen is less important than being **consistent** within a large code project.

```
def f3(lst):  
    ret=[]  
    iterator    =lst.split(" ")  
    for i in (iterator):  
        ret.append( i.lower() )  
    return(ret)
```

```
def getWords(sentence):  
    words = [ ]  
    for w in sentence.split(" "):  
        # make all words lowercase  
        words.append(w.lower())  
    return words
```

Libraries, APIs, and Documentation

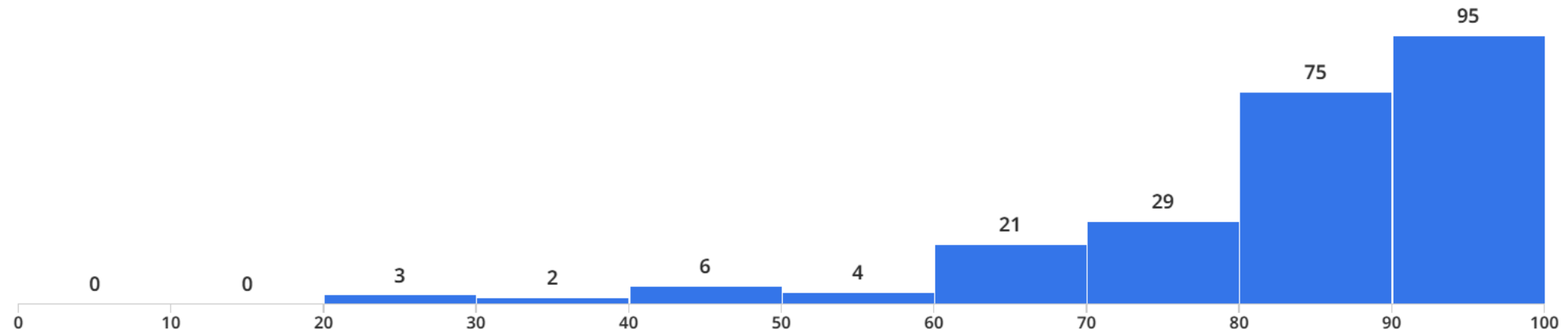
15-110 – Monday 10/05

Announcements

- Hw5 was due today
- No assignment released this week due to fall break
- Quizlet4 on Wednesday
- Deadline to register to vote in PA is **Monday October 19th**
 - <https://www.pavoterservices.pa.gov/pages/VoterRegistrationApplication.aspx>

Announcements – Exam1 Grades

- Median grade was an **87.25** – well done!
 - #4 – Code Tracing with Strings and Lists was extra challenging.



Minimum

21.0

Median

87.25

Maximum

100.0

Mean

83.46

Std Dev ?

--

Learning Goals

- Identify Python **libraries** and **web APIs** that can help you generate programs that do interesting things.
- Learn how to work with Python **libraries** and **web APIs** by using **documentation** and **tutorials**.
- Given a piece of Python code that uses constructs you have not learned before, use **resources** like online tutorials and AI tools to determine what the new constructs do.

External Libraries

Recap: Python Modules

The Python programming language comes with a large set of built-in functions that cover a range of different purposes. However, it would take too long to load all these functions every time we want to run a program.

Python organizes its different functions into individual files called **modules**. When you run Python, it loads only a small set of functions from the built-in module. To use any other functions, you must **import** them.

Python Standard Library

We've already used a few core modules for homework assignments - mainly `math` and `random`.

For a full list of modules in the Python Standard Library, look here:
<https://docs.python.org/3/library/index.html>

External Libraries

There are many other libraries that have been built by developers outside of the core Python team to add additional functionality to the language. These libraries don't come as part of the Python language, but can be added in. We call these **external libraries**.

Popular external libraries include matplotlib, scikit-learn, and pygame.



Finding Useful Libraries

One of the main strengths of Python as a language is that there are thousands of external libraries available, which means that you can start many projects based on work others have done instead of starting from scratch.

You can find a list of popular libraries here:

wiki.python.org/moin/UsefulModules

And a more complete list of easily-installable libraries here: pypi.org

Install External Libraries with `pip`

In order to use an external library, you must first **install** it on your machine. To install, you'll need to download the files from the internet to your computer, then integrate them with the main Python library so that the language knows where the new library is located.

It is usually possible to install libraries manually, but this process can be a major pain. Luckily, Python also gives us a streamlined approach for installing libraries – the **`pip` tool**! This feature can locate libraries that are indexed in the Python Package Index (the list of commonly-used libraries linked on the last slide), download them, and attempt to install them.

Running pip

Traditionally, programmers run `pip` from the **terminal**. This is a command interface that lets you make changes directly to your computer. You can access this in the application **Terminal** (Mac/Linux) or **Command Prompt** (Windows). To run `pip` in the terminal, use the command:

```
pip install library-name
```

This will identify the library and start the download and installation process. It may run into a **dependency error** if the library needs another library to already be installed – in general, installing that library and then running `pip` again will fix the problem.

Thonny Has its Own Installer

If you're using Thonny, you can install libraries directly in the application.

Go to **Tools > Manage Packages**. Search for the library you want to install and click on it in the results. Then click 'Install' and Thonny will handle the pip installation for you.

You do: try installing the language analysis library **nltk**. Go to Tools > Manage Packages, search for nltk, and click **Install**.

Using an Installed Library

Once you've successfully installed a library, you should be able to put

```
import module-name
```

at the top of a Python file, and it will load the module the same way it would load a regular module from the standard library. For example:

```
import nltk
```

Note: this may fail if you have multiple versions of Python installed on your machine and you install in the terminal. Make sure to use the `pip` associated with the version of Python you're using in your editor. You can check your editor's version in Thonny with Help > About Thonny, then call `pip` using

```
pythonversion-number -m pip install library-name
```

Example: Coding with nltk

Let's generate some code that uses nltk to analyze the sentiment of each of the chapters of a book.

Copying all the text of a book into a Python file would be unwieldy. Let's use **files** to hold onto large datasets instead.

Sidebar: Coding with Files

File System Organization

The files on your computer are organized in **directories**, or **folders**. Folders can hold other folders or files.

When you're working with files, always make sure you know which sequence of folders your file is located in. A sequence of folders from the top-level of the computer to a specific file is called a **filepath**.

For example, **Users > krivers > Documents > sample.txt** refers to the file **sample.txt** in the **Documents** folder, which is in the **krivers** folder, which is in the **Users** folder, which is at the top level of the computer.

Opening Files in Python

To interact with a file in Python we'll need to access its contents. We can do this by using the built-in function `open(filepath)`. This will create a **File object** which we can read from or write to.

```
f = open("/Users/kdrivers/Documents/sample.txt")
```

`open` can either take a full filepath or a **relative path** (relative from the location of the python file). Relative paths are preferred as they work across different computers. It's often easiest to put the file you want to read/write in the same directory as the python file so you can simply refer to the filename directly.

```
f = open("sample.txt")
```

```
# if .py file is in Documents, will search for this file there
```

Reading and Writing from Files

When we open a file we can specify whether we plan to **read from** or **write to** the file. This will change the **mode** we use to open the file.

```
filename = "sample.txt"
```

```
f1 = open(filename, "r") # read mode
```

```
text = f1.read() # reads the whole file as a single string
```

```
f2 = open("sample2.txt", "w") # write mode
```

```
f2.write(text) # writes a string to the file
```

Be Careful When Programming With Files

WARNING: when you write to files in Python, backups are not preserved. If you overwrite a file, the previous contents are gone forever.

Be careful when writing to files! Make sure you're using the correct filename before you run the program. Avoid overwriting original data whenever possible; you can always wait and delete it after you're done.

Example: Coding with nltk

We can find a book on [Project Gutenberg](#) and download it as a .txt to read it using Python.

Next, generate code that uses nltk to do sentiment analysis on the book.

Ensure the Python code and .txt file are saved in the same directory, then run to see the results.

Web APIs

APIs Are Interfaces

Programs commonly need to access data to find results. This data can be provided directly via a file, or it can be accessed from an online service using a **web API**.

API stands for Application Programming Interface. APIs describe how a program can interact with a system without getting into the details of how the system actually works.

APIs use **abstraction** to separate the part we care about (the interface) from the implementation.

Web APIs Provide Data

A web API often provides a set of commands that can be used to access a **data set**. Often the data being provided is unavailable locally, or too large to be stored locally.

We can use the **HTTP protocol** to make structured requests of a web API, then receive a structured response back.

A **protocol** is a standard for formatting data/actions so that different devices can understand each other. We'll see more protocols in future lectures.

Example API Communication

Pokéapi provides information about the popular game Pokémon.

We can make an HTTP GET request to look up information about Growlithe by setting the name parameter to Growlithe:

<https://pokeapi.co/api/v2/pokemon/Growlithe/>

We receive a set of data in the JSON format about that Pokemon and can access individual parameter names to get specific results.

```
data["types"] # [ "fire" ]
```

JSON is a protocol for storing heavily structured data

API Keys

Some Web APIs are free to use; others require an **API key** which you have to apply/pay for. This is a secret code that basically acts as your account with the service.

Most APIs limit the number of requests you can make per day.

To explore free APIs, try the following directory:

<https://github.com/public-apis/public-apis>

Example: Coding with Open-Meteo

Open-Meteo is a free API that provides weather data.

Let's generate some code to get the current temperature in Pittsburgh.

The website provides multiple **endpoints** – individual places where different kinds of data can be accessed. Let's explore what's available:

https://open-meteo.com/en/features#available_apis

Documentation

Documentation Describes Functionality

Suppose you find a new library or API you want to use. How can you learn how it works?

Often the best answer is to **read the documentation**. Most external libraries and web APIs have official documentation that describes how to use the system.

nltk: <https://www.nltk.org/>

Open-Meteo: <https://open-meteo.com/en/docs>

References Describe Functions

Library documentation usually includes a **reference** that describes the modules available in a library and the functions implemented in those modules. Similarly, web API documentation includes all the parameters available in the input/output of a request.

This is an API as well, but one that describes how to use the system.

Example:

https://www.nltk.org/api/nltk.sentiment.sentiment_analyzer.html#nltk.sentiment.sentiment_analyzer.SentimentAnalyzer

Tutorials Provide Examples

Many systems provide **tutorials** that give detailed examples of how to do popular things with a system. Looking over tutorials can give us a sense of what is possible with a library.

If tutorials aren't available on the main page for a library, try searching online to see if other websites have created tutorials of their own.

Example: <https://www.nltk.org/howto/sentiment.html>

Other Resources - Github

Github is a popular site for storing open-source code online so others can use and contribute to it.

Browsing Github can help you find projects other programmers have created before that might prove useful.

Example: <https://github.com/samiramunir/Simple-Sentiment-Analysis-using-NLTK/>

Always check whether a project has a **software license** that describes how the code may be used. Some (like the MIT license) allow code to be used openly; others (like GNU GPLv3) prevent code from being used for commercial purposes.

Always cite code that you get from someone else!

Other Resources - StackOverflow

StackOverflow is a popular site for asking questions about programming.

If you've run into an odd error about a library/API, there are good odds someone on StackOverflow has encountered it before and already gotten help.

It's also a good place to discuss broader questions about how to solve computational problems.

Example: <https://stackoverflow.com/questions/44522536/german-stemming-for-sentiment-analysis-in-python-nltk>

Other Resources – AI Agents

Finally, **AI agents** can be a great way to get personalized support when learning about a new library or API.

Often the agents draw information directly from the resources mentioned before, and they can get right to the part you care about.

As always, verify AI results before trusting them.

Example: Zombie Simulation with Pygame

[Pygame](#) is a very popular Python library for building games and simulations.

We can use an AI agent to generate a game quickly, but the code structure is different from what we've seen before.

Let's use Pygame's documentation and tutorials to understand how the code works.

Learning Goals

- Identify Python **libraries** and **web APIs** that can help you generate programs that do interesting things.
- Learn how to work with Python **libraries** and **web APIs** by using **documentation** and **tutorials**.
- Given a piece of Python code that uses constructs you have not learned before, use **resources** like online tutorials and AI tools to determine what the new constructs do.