Discrete Structures: CMPSC 102

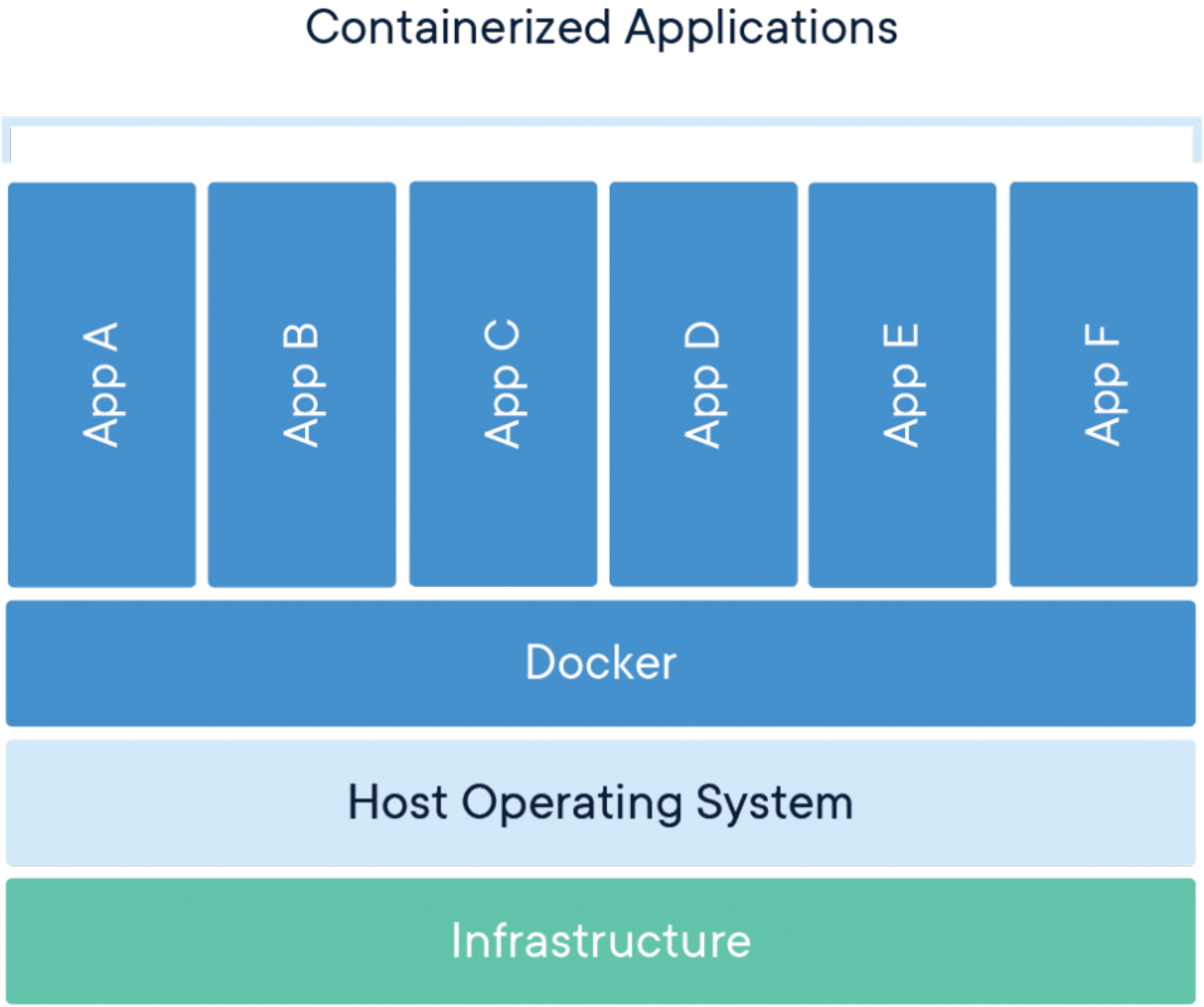
Oliver BONHAM-CARTER

Fall 2019
Week 2

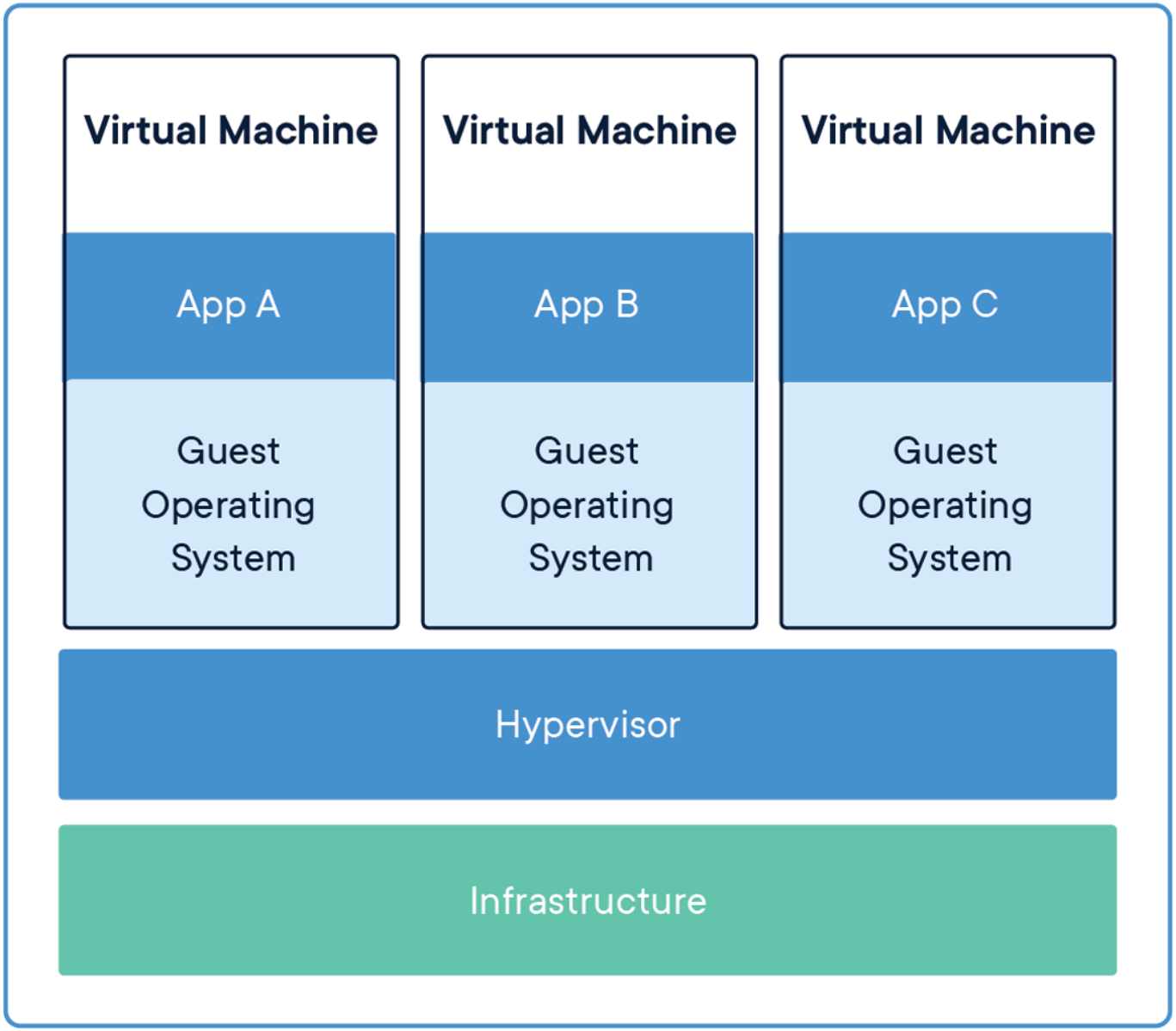
What is Docker?

- Using Docker to run Python in a virtual machine.

More about containers?
See: `https://www.docker.com/resources/what-container`



Virtual Machines
Docker



Commands

How did you install Docker?

- Find the ToolBox Docker Quickstart Terminal to run
- Or, find the Docker Application to run
- You will be using the terminal window that is opened for your own work

Toolbox Quickstart command, MacOS

- `bash --login '/Applications/Docker/Docker Quickstart Terminal.app/Contents/Resources/Scripts/start.sh'`

Check that Docker is working

- `docker run -t hello-world`

List Docker's containers

- `docker images ls`

Python3

Running Python3 with Docker

- `docker run -t python`

Sample output

```
obonhamcarter$ docker run -t python
Unable to find image 'python:latest' locally
latest: Pulling from library/python
4ae16bd47783: Already exists
bbab4ec87ac4: Already exists
2ea1f7804402: Already exists
96465440c208: Already exists
6ac892e64b94: Already exists
5b3ec9e84adf: Already exists
317202007d7c: Already exists
ba1ee226143f: Already exists
e33fb8e92c2f: Already exists
Digest: sha256:5f980d2673056ca33580979bcfb70c82f70ceccf857b77b740da40701e5d473e
Status: Downloaded newer image for python:latest
Python 3.7.4 (default, Aug 14 2019, 12:09:51)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

About Python

- `www.python.org`
- Download python3 if you are using your own hardware

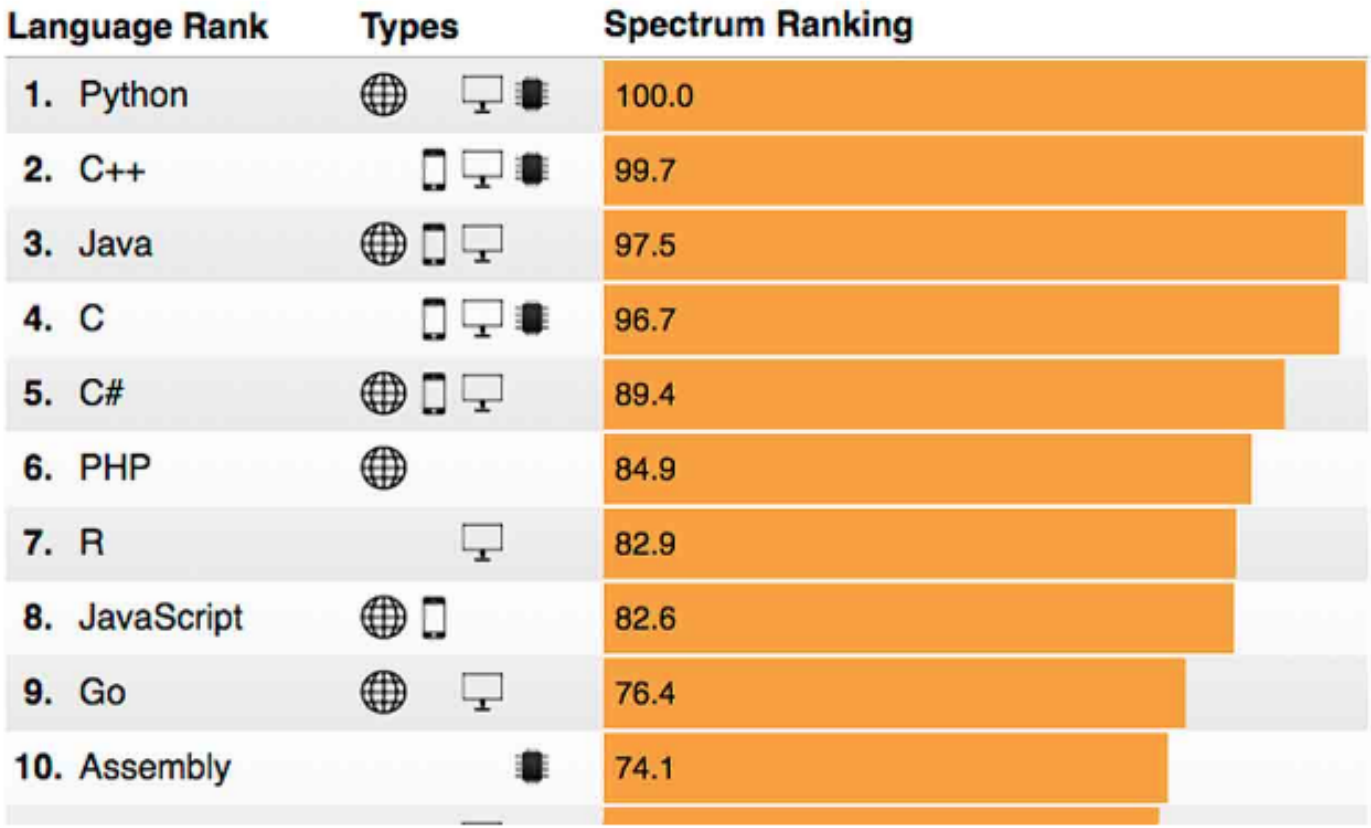
About Python...

- Is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Excellent for Rapid Application Development thanks to Its high-level built in data structures, combined with dynamic typing and dynamic binding
- A scripting language for tool-making or automation
- Used for *quick and dirty* solutions, quick automation, or to connect existing components together from other languages.

About Python...

- Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
- Python supports modules and packages, which encourages program modularity and code reuse.
- The Python interpreter and the extensive standard library are open source and freely available in all major platforms

The 2018 Top Programming Languages



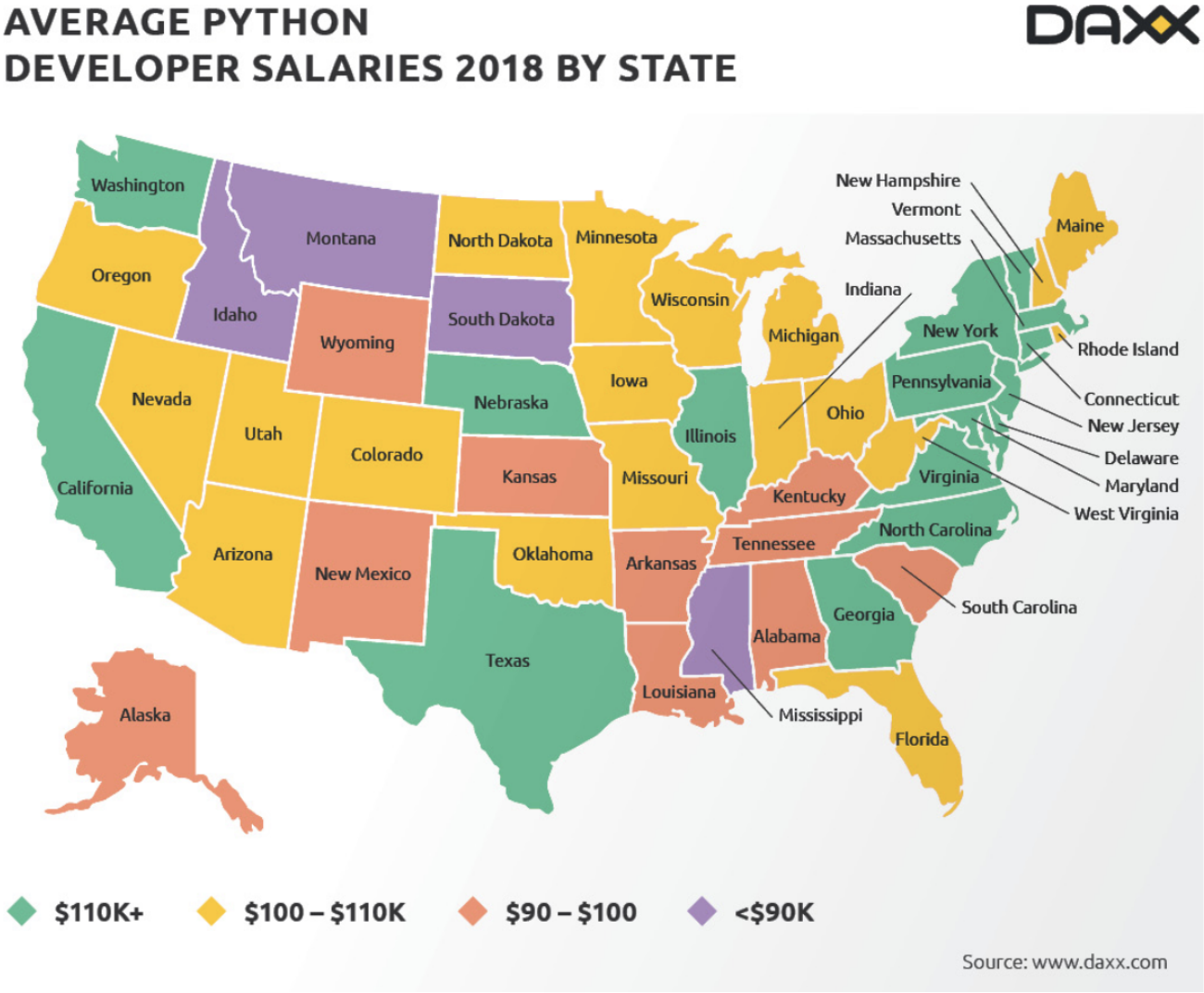
https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages

The 2018 Top Programming Languages

Most wanted programming languages 2018

https://www.daxx.com/article/python-developer-salary-usa

US High-Paying Python Development Jobs



https://www.daxx.com/article/python-developer-salary-usa

Average Salaries in Programming

Skill	Average salaries	Monthly jobs advertised
Python	US$116,379	6,550
Ruby	US$115,005	1,080
Java	US$112,592	10,443
Perl	US$111,928	1,398
C++	US$108,123	3,567
JavaScript	US$103,503	8,764
C#	US$101,715	4,101
PHP	US$94,690	1,664
ASP.NET	US$95,551	1,289
C	US$95,166	5,639

https://www.daxx.com/article/python-developer-salary-usa

Who Uses Python

Users of Python Programming

- Industrial Light and Magic (George Lucas to create the FX for Star Wars).
- Google
  - Googles very first web-crawling spider was first written in Java 1.0 and was so difficult that they rewrote it into Python.
- Facebook
  - Responsible for multiple services in infrastructure management
- Netflix
  - Used to power data analyses tasks from the server side
- Dropbox
  - Built its API in Python
- And others; Instagram, Spotify, Quora, Reddit

Where can I learn more about the language?

Free online resources

- Think Python First Edition, by Allen B. Downey
  - `http://greenteapress.com/wp/think-python/`
- A Collection of Tutorials
  - `https://wiki.python.org/moin/BeginnersGuide/Programmers`
- Interactive Python Tutorial
  - `https://www.learnpython.org/`
- Host, run, and code Python in the cloud!
  - `https://www.pythonanywhere.com/`

Running the Python3 Shell

- Type statements or expressions at prompt:
- print("Hello, world")
- x = 12**2
- print(x)
- print(x/2)
- # bla bla bla...
  - (this is a comment: everything after the # is ignored)

Data types

Note: Use identifiers to help you remember the types!

- Integers, counting numbers
  - num_int = 1
- Floats, decimals
  - num_float = 3.1415
- Strings
  - s_str = " Hello World"

Combining variables in print statements

```
x_int = 1
print(" The integer variable is :", x_int)

num_float = 3.14
print(" The float variable is :", num_float)

s_str =("Hello World'')
print(" The integer is equal to", s_str)
```

Mathematical Operators

Mathematical Operators

- $3 + 4$ # Addition
- $3 - 4$ # Subtraction
- $3 * 4$ # Multiplication
- $3/4$ # Division ($\frac{3}{4}$)
- $3\%4$ # Modulus; Returns the remainder from the division
- $3 * *4$ # Powers; raise three to the power of four

$$= 3 * 3 * 3 * 3$$
$$= 3^4$$
$$= \text{pow(3,4)}$$

Calling positions in strings


```
# Remember each char of a string has own position

s_str = "ABC"
s_str[0] = 'A'
s_str[1] = 'B'
s_str[2] = 'C'
s_str[200] = ??


# Another way to iterate
# through a string using its length

for i_int in range(len(s_str)):
  print(s_str[i_int])
```

Strings

Examples of working with strings

```
"hello" + " world"  # concatenation
"hello" * 3  #repetition
"hello"[0] # indexing
"hello"[-1] # indexing from end
"hello"[1:4] # slicing out a subsequence
len("hello") # determine how many characters, size
"hello" < "jello" # comparison of ABC order
"e" in "hello" # True, "e" is found in the string

# General rule:
    single quotes and double quotes are the same
    'abc' == "abc"
```

Working with strings

Characters at the front

```
line = "python programming is fun"
line.startswith("python") # True
line.startswith(" python") # False. Why is this?
```

Conditional Statements

Watch for the white space in the code!


```
if condition :          |       if a > b:
   statements           |          print("I'll take a")
else:                   |       else:
   statements           |          print("I'll take b")
```

Conditional Statements

Watch for the white space in the code!


```
if testScore > medianScore:
  print("Above average.")
else:
  if testScore == medianScore:
    print("Average.")
  else:
    print("Below average.")
```

Conditionals: If statements

An `if` statement is a programming conditional statement that, if proved true, performs a specific function or task. If the condition is false, then another procedure is performed instead.

```
num_int = 5 #  Assignment of 5 to variable "num_int"
if num_int == 3: # condition to check
  print("  True")  # condition is true
else: # condition is not true
  print("  False") # num_int, is NOT equal to 3


#make a compressed conditional statement,
# no "else" statement necessary
num_int = 4
if num_int == 4: print("True")
```

If statements


```
name_str = "Bill"
if name_str == "Bill":
  print("Hello Bill!")
else:
  print("You are not Bill.")
  # Place the name in a string to print
  print("Your name is :",name_str)
```

For statements

A `for` loop is a statement in programming that performs predefined tasks while or until a predetermined condition is met.

```
# counter program
for i_int in range(10):
  print("  Count is:" ,i_int)
  # Note: you could add some conditional
  # if-statement here to check the value of i_int.

# Iterate through the string's chars
s_str = "hello world"
for i_int in s_str:
  print(i_int)
  # Note: you could add an if-statement here
```

Find a Single Variable in a File

Watch out! Python uses white spaces (spaces and tabs) to define its blocks of code.

Make a source code: `vim nameReader.py`

```
file = open("names.txt")
for line in file:
    print("  Reading this line: ",line)
    if line.startswith("James"):
        print("** Found the name: ",line)
```

Make a textfile: `vim names.txt`

```
Jane smyth
Betty Davis
John smith
Buffalo Bill
James Bond
```

Find Two Variables in File

Make a source code: vim findEmail.py

```
file = open("emails.txt")
for line in file:
    name, email = line.split(",")
    if name == "James Bond":
        print("  ** Found email: ",email)
```

Make a textfile: vim emails.txt

```
Jane smyth,smythj$ac.edu
Betty Davis,davisb@ac.edu
John smith,smithj@mum.com
Buffalo Bill,buffalob@prairie.com
James Bond,bondj@magestySecrets.co.uk
```

Find the Summation of Numbers in a File

Make a source code: vim numberChecker.py

```
file = open("numbers.txt")
sum_int = 0
# defined outside of loop to be used inside
  # and outside of loop
for num in file:
    n_int = int(num) # convert string to integer
    print(" Reading this number: ",n_int)
    sum_int = sum_int + n_int
print("  ** The summation of the number is :", sum_int)
```

Make a textfile: vim numbers.txt

```
1
2
3
4
5
6
```