Winter Institute in Data Science and Big Data

Introducing Python

Le Bao

Massive Data Institute, Georgetown University

January 6, 2022

Plan

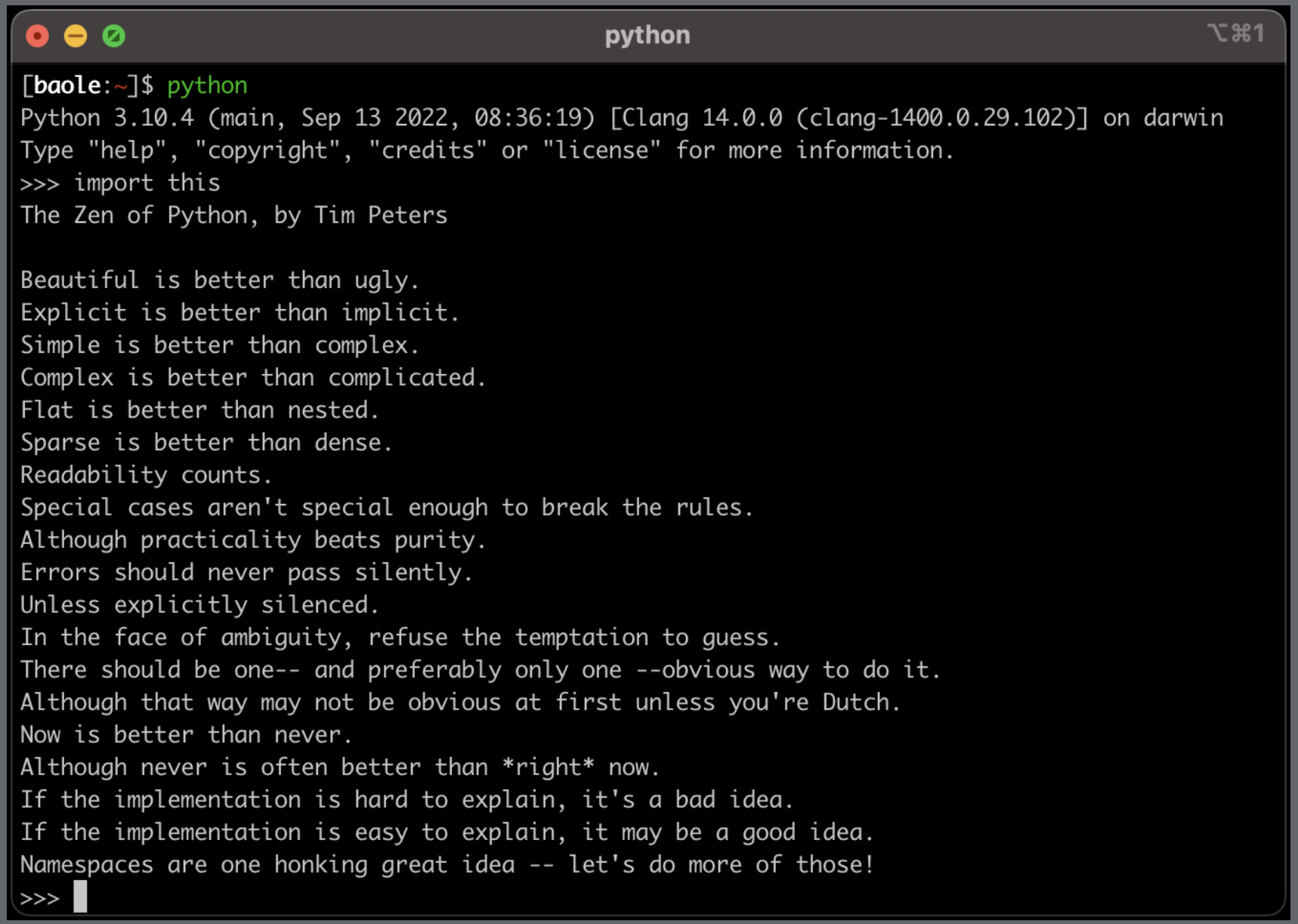
- Python
  - Basics
  - Data structures
  - Application: working with web data
- Comparative computing
  - Operating systems and system dependencies
  - Docker
  - Cloud computing with Code Ocean
- Containers and cloud computing
  - Operating systems and system dependencies
  - Docker
  - Cloud computing with Code Ocean

What is Python?

“Python is a high-level programming language, and its core design philosophy is all about code readability and a syntax which allows programmers to express concepts in a few lines of code.”

- Open-source
- General-purpose
- Interpreted
- Object-oriented
- Currently, one of the most popular programming languages

The Zen of Python



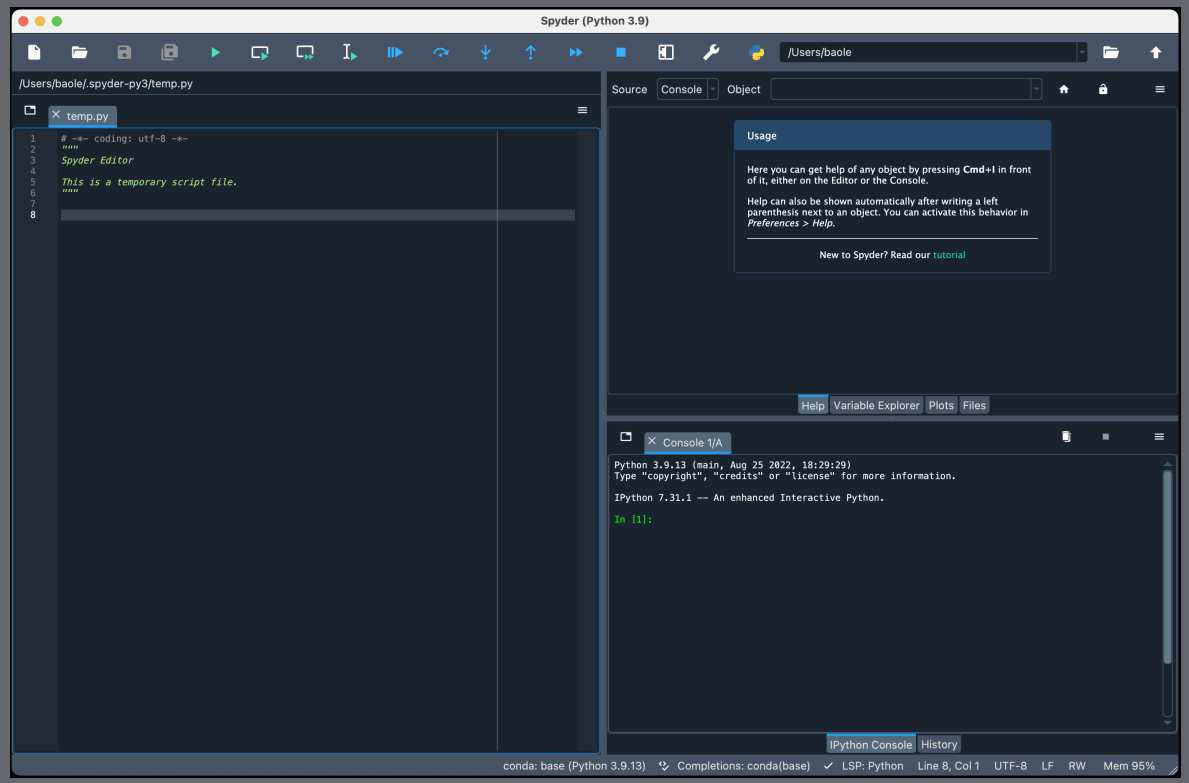
Python Ecosystem

- Web data & development: `BeautifulSoup`, `Selenium`, `Scrapy`, `Django`, `Flask`
- Data analysis and modeling: `Pandas`, `NumPy`, `Matplotlib`, `SciPy`
- Machine learning: `Scikit-Learn`, `TensorFlow`, `Keras`, `PyTorch`, `LightGBM`
- Computer vision: `OpenCV`, `Pillow`
- GUI app and game dev: `TkinterA`, `PygameA`, `PyOpenGL`
- Code presenting and reproducibity: `JupyterLab`
- Internet of Things: Raspberry Pi + python

Setup

- Anaconda distribution: https://www.anaconda.com/products/distribution
  - Everything you need is *out of the box*

- Spyder (IDE/editor) - like RStudio



Data Types

- Integer `<int>`
- Float `<float>`
- String `<str>`
- Boolean `<bool>`

```
type(5)
```

```
int
```

```
type('hello')
```

```
str
```

```
type(3.1415926)
```

```
float
```

Variables

```
x = 3.5
y = 2.7
z = x + y
z
```

```
6.2
```

```
x = 'hello'
y = 'world'
z = x + ' ' + y
z
```

```
'hello world'
```

Operators

- `+` plus
- `-` minus
- `*` multiply
- `/` divide
- `==` equal (`<`, `>`, `<=`, `>=`)
- `and`
- `or`
- `not`

Operators

```
x = 3.5
y = 'python'
x == 3.6
```

```
False
```

```
type(y) == str
```

```
True
```

```
(x == 3.6) and (type(y) == str)
```

```
False
```

```
(x == 3.6) or (type(y) == str)
```

```
True
```

Conditions

- **if/else**:
  - if/else statement executes a block of code if a specified condition is true. If condition is not met, another block of code can be executed.

```
num = 3.14

if (num >= 10):
    print('The number is larger than 10')
elif (num < 0):
    print('The number is negative')
elif (num == 0):
    print('The number equals zero')
else:
    print('The number is smaller than 10')
```

```
The number is smaller than 10
```

Loops

- For loops

```
for i in range(5):
  print(i)
```

```
0
1
2
3
4
```

```
range(5)
```

```
range(0, 5)
```

```
for i in range(1,5):
  print(i)
```

```
1
2
3
4
```

Loops

- While loops

```
num = 10

while(num <= 20):
    print('The number', num, 'is smaller than 20')
    num+=2
```

```
The number 10 is smaller than 20
The number 12 is smaller than 20
The number 14 is smaller than 20
The number 16 is smaller than 20
The number 18 is smaller than 20
The number 20 is smaller than 20
```

Functions

```
def sum_two_number(a,b):
  res = a + b
  return res

sum_two_number(5,7)
```

```
12
```

```
def even_or_odd(x):
  if x%2==0:
    return 'Even'
  else:
    return 'Odd'
```

```
even_or_odd(4)
```

```
'Even'
```

```
even_or_odd(201)
```

```
'Odd'
```

Exercise: Function

- Write a function to compare two values and return the larger one

```
1 def find_larger_number(a,b):
2   if a >= b: res = a
3   if a < b: res = b
4   return res
```

```
find_larger_number(5,7)
```

```
7
```

Data Structures

- Tuples, Lists, Sets, Dictionaries
- “Containers” that organize and group data according to type.
- Each is unique in its own way

Tuple

- A tuple is an (**immutable**) ordered list of values

```
t1 = () # Empty tuple
t2 = (1,2,3)
t3 = (1, "AU", 3.1, False, 1)
t3
```

```
(1, 'AU', 3.1, False, 1)
```

- Methods for tuples

```
t3.count(1)
```

```
2
```

```
t3.index("AU")
```

```
1
```

```
list(t3)
```

```
[1, 'AU', 3.1, False, 1]
```

List

- A list is the Python equivalent of an array, but is mutable and can contain elements of different types

```
l1 = []
l1 = list()
l2 = [1,"AU", True]
l2
```

```
[1, 'AU', True]
```

- Indexing/Slicing

```
nums = list(range(10))
print(nums)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
nums[5:6]
```

```
[5]
```

List

- List methods

```
progs = ["Python", "R", "Julia"]
print(progs)
```

```
['Python', 'R', 'Julia']
```

```
progs.append("C++")
print(progs)
```

```
['Python', 'R', 'Julia', 'C++']
```

```
progs.remove("C++")
progs.insert(1, "C++")
print(progs)
```

```
['Python', 'C++', 'R', 'Julia']
```

```
nums = [1,3,2,5,4]
nums.sort()
print(nums)
```

```
[1, 2, 3, 4, 5]
```

List

- Use list with loops

```
for p in progs:
  print(p)
```

```
Python
C++
R
Julia
```

```
for i in range(2,len(progs)):
  print(progs[i])
```

```
R
Julia
```

```
len(progs)
```

```
4
```

Sets

- A set is an **unordered** collection with **no duplicate elements**.

```
progs = {"python", "R", "C++", "R", "Julia"}
print(progs)
```

```
{'python', 'Julia', 'R', 'C++'}
```

```
nums = [1, 2, 3, 4, 4]
unique_nums = set(nums)
print(unique_nums)
```

```
{1, 2, 3, 4}
```

```
print(list(unique_nums))
```

```
[1, 2, 3, 4]
```

Dictionary

- A dictionary stores (key, value) pairs to map data

```
prog_dict = {1:"C++", 3:"R", 2:"python", 4:"Julia"}
print(prog_dict)
```

```
{1: 'C++', 3: 'R', 2: 'python', 4: 'Julia'}
```

```
prog_dict[2]
```

```
'python'
```

```
tokyo = {'city':"Tokyo", 'country':"Japan", 'pop': 37468000}
tokyo['country']
```

```
'Japan'
```

```
print(tokyo.keys())
```

```
dict_keys(['city', 'country', 'pop'])
```

Dictionary

- Dictionary, list, and loop

```
delhi = {'city':"Delhi", 'country':"India", 'pop': 28514000}
shanghai = {'city':"Shanghai", 'country':"China", 'pop': 25582000}
megacities= [tokyo,delhi,shanghai]
print(megacities)
```

```
[{'city': 'Tokyo', 'country': 'Japan', 'pop': 37468000}, {'city': 'Delhi',
'country': 'India', 'pop': 28514000}, {'city': 'Shanghai', 'country': 'China',
'pop': 25582000}]
```

```
for m in megacities:
  print(m['city'] + ", " + m['country'], "has a population of", m['pop'])
```

```
Tokyo, Japan has a population of 37468000
Delhi, India has a population of 28514000
Shanghai, China has a population of 25582000
```

Exercise: List, Function, and Loop

- Create a list of numbers and find all the prime numbers in it
- Hint:
  - You may need to use list, function, and/or loop.
  - You can use for loop to test all the divisors.

Exercise: List, Function, and Loop

- Code

```
find_prime(1,10)
```

```
[1, 2, 3, 5, 7]
```

```
len(find_prime(1,100))
```

```
26
```

Web Data

```
#! pip install requests pandas matplotlib
import requests
import io
import pandas as pd
import matplotlib.pyplot as plt

#Predictit data
url = "https://www.predictit.org/Resource/DownloadMarketChartData?marketid=7326
#Download data
web_response = requests.get(url, timeout = 30, stream = True)
#Read data
f = io.BytesIO(web_response.content)
speaker_bet = pd.read_csv(f)
```

Betting the House Speaker

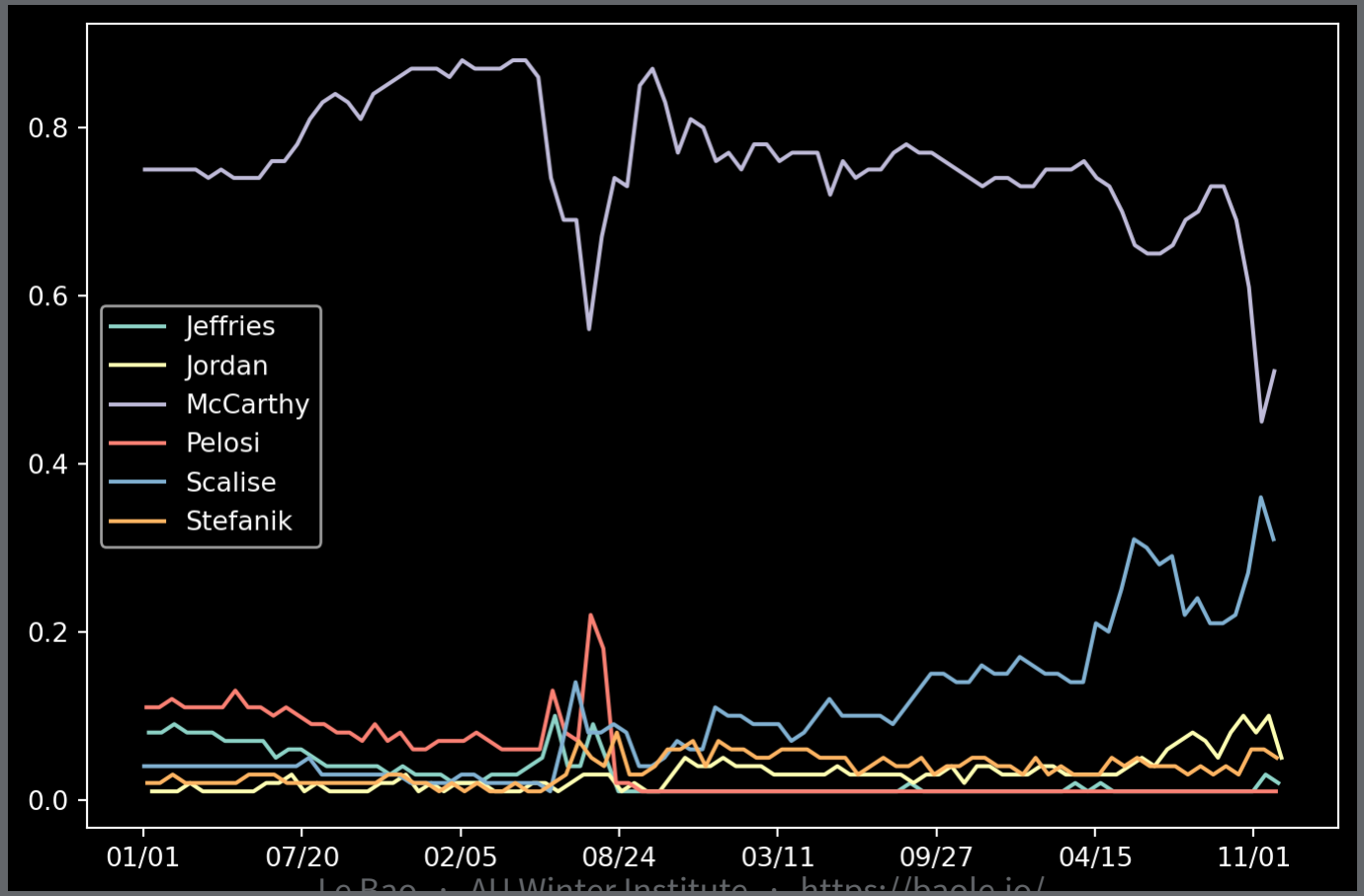
```
with pd.option_context('display.max_rows', 6):
    print(speaker_bet)
```

```
      ContractName                  Date OpenSharePrice HighSharePrice  \
0            Trump  10/8/2022 12:00:00 AM          $0.03          $0.04
1          Scalise  10/8/2022 12:00:00 AM          $0.04          $0.04
2         McCarthy  10/8/2022 12:00:00 AM          $0.75          $0.75
...            ...                   ...            ...            ...
1437  Ocasio-Cortez   1/5/2023 12:00:00 AM          $0.01          $0.01
1438          Clark   1/5/2023 12:00:00 AM          $0.01          $0.01
1439          Banks   1/5/2023 12:00:00 AM          $0.01          $0.02

     LowSharePrice CloseSharePrice  TradeVolume
0            $0.03           $0.04          340
1            $0.04           $0.04          565
2            $0.74           $0.75         1203
...            ...             ...          ...
1437         $0.01           $0.01           15
1438         $0.01           $0.01            0
1439         $0.01           $0.01        80290

[1440 rows x 7 columns]
```

Betting the House Speaker

```
speaker_bet['OpenSharePrice'] = speaker_bet['OpenSharePrice'].str.replace('$',
candidates = ['McCarthy','Scalise','Stefanik','Jordan','Jeffries','Pelosi']
```

- Code



Web Scraping

- Better still, use APIs
- Web Scraping
  - Static webpages
  - Dynamic webpages
  - Hidden webpages
- Parse html code to data
  - Interact with web elements: `BeautifulSoup`
  - Extract attributes from html elements

```
table_elements = results.find_all("div", class_="table")
for element in table_elements:
    print(element.text.strip())
```

Example

```
#! pip install bs4
import requests
from bs4 import BeautifulSoup as bSoup
```

- NYTimes 2020 Election page: https://www.nytimes.com/interactive/2020/11/03/us/elections/results-president.html?action=click&pgtype=Article&state=default&module=styln-elections-2020&region=TOP_BANNER&context=storyline_menu_recirc