READING PYTHON

a program of commands for talking to your computer

Dr Heather Froehlich
University of Arizona Libraries
froehlich@arizona.edu

CC BY

Expectations

By the end of this time you should be able to…

- Be able to identify some discrete pieces of code
- Identify (at a high level) the processes we are going through

Things to keep in mind

- We are not going to be pros at this
- This is the culmination of approx 10 years of practice + knowledge

Python can run on your computer… but

- But that involves a lot of steps…
- And then you have to write the whole thing from scratch ☹

- Jupyter and Colab are computational notebooks
  - workbook where you can run things in the browser
  - Someone else writes the code, you can work with it

Some python basics

Python has a syntax...

....and it builds on what happened immediately before

Something = some kind of activity

- x = 0 + 1
- y = x + 1

-> Read from left to right

<- And right to left

Python also likes using color to differentiate between different things!

Variables in python

- A variable is like a tiny container where you store values and data
  - Filenames
  - Words
  - Numbers
  - Collections of words and numbers
  - Etc

useful vocabulary for python

=	Assign variable
_	Temporary function in a variable
.	Attribute of a variable
[ ]	A list of items that **can** be changed, usually within a variable assignment
“ ”, ‘ ’	Defines a string (i.e. specific letters in an order)
( )	A list of things that **aren’t** changeable.
( )	Closed parentheses () “calls” or activates a particular activity
#	Starts a comment (often lives on top of a set of code)

Let’s look at an example

```
[ ] # Reset index and add column names to make wrangling easier
    paper_df = paper_df.reset_index()
    paper_df.columns = ["Filename", "Text"]
    paper_df.head()
```

Let’s look at an example

```
[ ] # Reset index and add column names to make wrangling easier
    paper_df = paper_df.reset_index()
    paper_df.columns = ["Filename", "Text"]
    paper_df.head()
```

Let’s look at an example

```
[ ] # Reset index and add column names to make wrangling easier
    paper_df = paper_df.reset_index()
    paper_df.columns = ["Filename", "Text"]
    paper_df.head()
```

Let’s look at an example

```
[ ] # Reset index and add column names to make wrangling easier
    paper_df = paper_df.reset_index()
    paper_df.columns = ["Filename", "Text"]
    paper_df.head()
```

Let’s look at an example

```
[ ] # Reset index and add column names to make wrangling easier
    paper_df = paper_df.reset_index()
    paper_df.columns = ["Filename", "Text"]
    paper_df.head()
```

Some other good tips and tricks

`df`	Short for “dataframe”. Lets python play nicely with data
`head()`	Displays the initial rows of a data set
`import`	Introduce a new library or package
`from`	Tells python where to get information from
`print`	Show exactly this outcome
`for`	Every time this thing happens, do a specific action

Libraries in python

- A library or package is a collection of pre-written code kits
  - They allow you do a bunch of different things
- Sometimes you have to add a bunch of them in a row
  - There is probably an order to care about
  - What do you need to know?

Let’s look at an example

Import Packages

```
[ ] # Import spacy
    import spacy

    # Load spaCy visualizer
    from spacy import displacy

    # Import pandas DataFrame packages
    import pandas as pd

    # Import graphing package
    import plotly.graph_objects as go
    import plotly.express as px
```

Let’s look at an example

Import Packages

```
[ ] # Import spacy
    import spacy

    # Load spaCy visualizer
    from spacy import displacy

    # Import pandas DataFrame packages
    import pandas as pd

    # Import graphing package
    import plotly.graph_objects as go
    import plotly.express as px
```

?

pandas is library specifically for working with data, cleaning it up, etc

Let’s look at some more examples

We’re going to use samples from Melanie Walsh’s great *Introduction to Cultural Analytics & Python* (2021)

Check it out at: *https://melaniewalsh.github.io/Intro-Cultural-Analytics/welcome.html*

Let’s look at some more examples

```
for person in bellevue_people.values():
    print(person)
```

```
{'name': 'Mary Gallagher', 'age': 28, 'profession': 'married'}
{'name': 'John Sanin(?)', 'age': 19, 'profession': 'laborer'}
```

```
for person in bellevue_people.values():
    if person['age'] > 20:
        name = person['name']
        age = person['age']
        print(f'{name} is more than 20 years old. She is {age}.')
```

```
Mary Gallagher is more than 20 years old. She is 28.
```

https://melaniewalsh.github.io/Intro-Cultural-Analytics/02-Python/11-Dictionaries.html/

Let’s look at some more examples

```
for person in bellevue_people.values():
    print(person)
```

```
{'name': 'Mary Gallagher', 'age': 28, 'profession': 'married'}
{'name': 'John Sanin(?)', 'age': 19, 'profession': 'laborer'}
```

```
for person in bellevue_people.values():
    if person['age'] > 20:
        name = person['name']
        age = person['age']
        print(f'{name} is more than 20 years old. She is {age}.')
```

```
Mary Gallagher is more than 20 years old. She is 28.
```

Let’s look at some more examples

```
for person in bellevue_people.values():
    print(person)
```

```
{'name': 'Mary Gallagher', 'age': 28, 'profession': 'married'}
{'name': 'John Sanin(?)', 'age': 19, 'profession': 'laborer'}
```

```
for person in bellevue_people.values():
    if person['age'] > 20:
        name = person['name']
        age = person['age']
        print(f'{name} is more than 20 years old. She is {age}.')
```

```
Mary Gallagher is more than 20 years old. She is 28.
```

Let’s look at some more examples

```
# List of sentences
sentences = ["I like the Marvel movies",
             "I don't like the Marvel movies",
             "I despise the Marvel movies with every fiber of my being",
             "I don't *not* live the Marvel movies"]

# Loop through list of sentences
for sentence in sentences:
    # Run VADER on each sentence
    sentiment_scores = sentimentAnalyser.polarity_scores(sentence)

    # Print scores for each sentence
    print(f"""'{sentence}' \n
😕 Negative Sentiment: {sentiment_scores['neg']} \n
😐 Neutral Sentiment: {sentiment_scores['neu']} \n
😀 Positive Sentiment: {sentiment_scores['pos']} \n
✨ Compound Sentiment: {sentiment_scores['compound']} \n
--- \n""")
```

https://melaniewalsh.github.io/Intro-Cultural-Analytics/05-Text-Analysis/04-Sentiment-Analysis.html/

Let’s look at some r

```
# List of sentences
sentences = ["I like the Marvel movies",
             "I don't like the Marvel movies"
             "I despise the Marvel movies wit
             "I don't *not* live the Marvel m

# Loop through list of sentences
for sentence in sentences:
    # Run VADER on each sentence
    sentiment_scores = sentimentAnalyser.pola

    # Print scores for each sentence
    print(f"""'{sentence}' \n
😕 Negative Sentiment: {sentiment_scores['neg
😐 Neutral Sentiment: {sentiment_scores['neu'
😀 Positive Sentiment: {sentiment_scores['pos
✨ Compound Sentiment: {sentiment_scores['com
--- \n""")
```

```
'I like the Marvel movies'

😕 Negative Sentiment: 0.0

😐 Neutral Sentiment: 0.361

😀 Positive Sentiment: 0.639

✨ Compound Sentiment: 0.6486

---

'I don't like the Marvel movies'

😕 Negative Sentiment: 0.526

😐 Neutral Sentiment: 0.474

😀 Positive Sentiment: 0.0

✨ Compound Sentiment: -0.5334

---

'I despise the Marvel movies with every fiber of my being'

😕 Negative Sentiment: 0.169

😐 Neutral Sentiment: 0.634

😀 Positive Sentiment: 0.197

✨ Compound Sentiment: 0.1027

---

'I don't *not* live the Marvel movies'

😕 Negative Sentiment: 0.28

😐 Neutral Sentiment: 0.72

😀 Positive Sentiment: 0.0

✨ Compound Sentiment: -0.3252
```

Let’s look at some more examples

```
people = []

for document in chunked_documents:
    for named_entity in document.ents:
        if named_entity.label_ == "PERSON":
            people.append(named_entity.text)

people_tally = Counter(people)

df = pd.DataFrame(people_tally.most_common(), columns=['character', 'count'])
df
```

https://melaniewalsh.github.io/Intro-Cultural-Analytics/05-Text-Analysis/12-Named-Entity-Recognition.html

Let’s look at some more examples

```
people = []

for document in chunked_documents:
    for named_entity in document.ents:
        if named_entity.label_ == "PERSON":
            people.append(named_entity.text)

people_tally = Counter(people)

df = pd.DataFrame(people_tally.most_common(), columns=['char
df
```

	character	count
0	Jo	1256
1	Amy	645
2	Laurie	570
3	Beth	465
4	Meg	311
5	John	144
6	Hannah	122
7	Brooke	96
8	Laurence	85
9	Bhaer	77

https://melaniewalsh.github.io/Intro-Cultural-Analytics/05-Text-Analysis/12-Named-Entity-Recognition.html

Last one…

```python
def get_neighbor_words(keyword, bigrams, pos_label = None):

    neighbor_words = []
    keyword = keyword.lower()

    for bigram in bigrams:

        #Extract just the lowercased words (not the labels) for each bigram
        words = [word.lower() for word, label in bigram]

        #Check to see if keyword is in the bigram
        if keyword in words:

            for word, label in bigram:

                #Now focus on the neighbor word, not the keyword
                if word.lower() != keyword:
                    #If the neighbor word matches the right pos_label, append it to the
                    if label == pos_label or pos_label == None:
                        neighbor_words.append(word.lower())

    return Counter(neighbor_words).most_common()
```

#If the neighbor word matches the right pos_label, append it to the master list

https://melaniewalsh.github.io/Intro-Cultural-Analytics/05-Text-Analysis/13-POS-Keywords.html/

Additional resources

- Melanie's whole book! https://melaniewalsh.github.io/Intro-Cultural-Analytics

- Constellate's Text Analysis tutorials and courses https://github.com/ithaka/constellate-notebooks
- Codeacademy (and other web-based tutorials) https://try.codecademy.com/learn-python-3

- Library Carpentry python workshop https://librarycarpentry.org/lc-python-intro/aio.html

save a copy of these slides!