SWIFT
SHENZHEN UNIVERSITY WEBANK INSTITUTE OF FINTECH
深大微众金融科技学院

Chapter 0: Python Recap

Artificial Intelligence and Machine Learning

Why Python?

Top three programming language[1]

One of the biggest languages for data science[2]

Widely available tutorials[3]

1. https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019 2. https://bigdata-madesimple.com/top-8-programming-languages-every-data-scientist-should-master-in-2019/
3. http://pypl.github.io/PYPL.html

Python offers the most comprehensive set of Data Science capabilities today for a wide range of domains

Data handling

NumPy
Pandas[1]
SciPy
PySpark

Data Mining

Scrapy
Statsmodels

Machine Learning
Deep Learning

SciKit-Learn
Keras
TensorFlow

Natural Language Processing

NLTK
Spacy
Word2vec
Gensim
PyTorch

Visualization

Matplotlib[1]
Seaborn
Bokeh
Plotly
PyViz
Holoviews

1. We will use the packages highlighted in red in today's exercise

Programming languages are like written languages or mathematical expressions

Symbols

- words comma question-mark quote apostrophe full-stop exclamation
- numbers operators brackets
- reserved-words variables operators delimiters literals

Grammar examples

- [conjunction] [noun] [preposition] [adjective]
- [bracket] [number] [operator] [number] [bracket] [operator] [number]
- [reserved-word] [variable] [operator] [literal] [delimiter]

The full grammar for a language consists of complex rules of construction

Python grammar has only 35 reserved words

conditional:	if, else, elif
looping:	while, for, break, continue
logic:	and, not, or
values:	True, False, None
namespaces:	import, from, as, del, global
object creation:	class, def, lambda
functions:	return, yield
exceptions:	try, except, finally, raise
misc:	pass, await, async, assert, with, exec, in, is

https://docs.python.org/3.7/reference/lexical_analysis.html#keywords

By the end of this workshop you'll be familiar with **~35%** of these

One of the stand-out advantages of Python is that it is considered a very “readable” programming language

- Try to spot reserved-words, builtins, literals, variables, symbols in the example below
- Can you tell what this program is doing and how it would run with an input class_list?

```
class_size = len(class_list)
honor_list = []

for student in class_list:
    print(f'Processing student {student.name} with grade {student.grade}')

    if student.grade > 80.0:
        student.honors=True
        honor_list.append(student)

    if student.absent_days > 10:
        print(f'Student {student.name} had high absentee rate: {student.absent_days}')

honor_student_count = len(honor_list)

print(f'There are {honor_student_count} honor students in this class')
```

Before we move ahead…

…any questions from prework?

Concepts—Lists and Loops

Lists

Define an empty list with []

Several useful functions to apply to a list

- .append(el) adds an element to the back
- len(list) returns the length of the list
- Access an item in a list with list[x], with x being the index (starting at 0)

```
digital_list = ['gamma', 'dv', 'platinion']
digital_list.append('omnia')
digital_list
> ['gamma', 'dv', 'platinion', 'omnia']
digital_list[0]
> 'gamma'
digital_list[-1]
> 'omnia'
```

Concepts—Lists and Loops

Loops

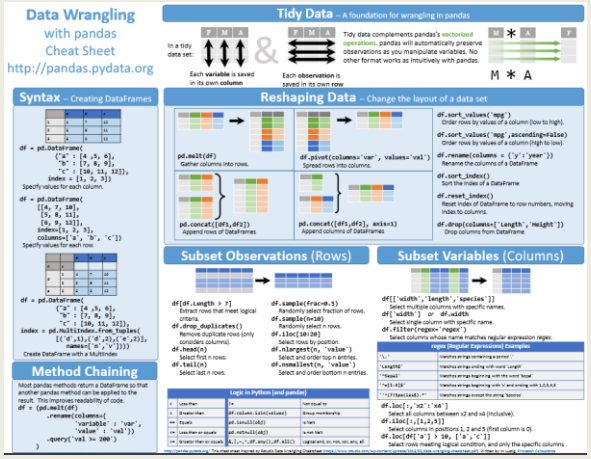
To iterate through every element in myList
You can call the elements anything

for el in myList:
    statements to do something ....

```
for abm in digital_list:
    if abm == 'gamma':
        print('this abm is the best')
    else:
        print(abm)
 > this abm is the best
    dv
    platinion
    omnia
```

Pandas is very are well documented: cheat sheet and cookbook are your friends

Pandas Cheat Sheet



Source: https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

Cookbook

https://pandas.pydata.org/pandas-docs/stable/cookbook.html

A set of ~30 commands is all it takes to assess a 2M-row dataset and to effectively team with GAMMA

Internal Python commands

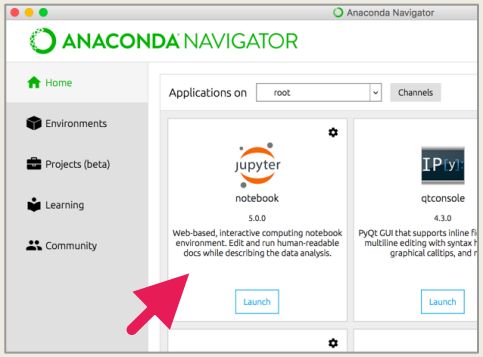
1) list()
2) dict()
3) import
4) type()
5) for-loop
6) if-elif-else
7) sum()
8) round()
9) .append()
10) .pop()
11) def()
12) return

pandas and **matplotlib** package commands

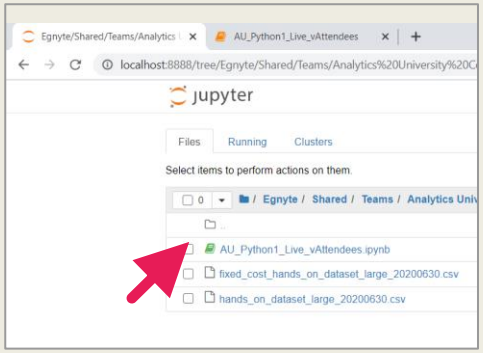
1) .read_csv()
2) .shape
3) .head()
4) .columns
5) .index
6) .iloc[]
7) .loc[]
8) .describe()
9) .unique()
10) .value_counts()
11) .sort_values()
12) .sort_index()
13) .groupby()
14) .to_csv()
15) .hist()
16) .plot.bar()
17) .plot.scatter()
18) .plt.show()
19) .plt.title()

Let's jump over to our Jupyter notebook

- Open Anaconda Navigator
- Launch Jupyter Notebook



- Navigate to where you've download course assets
- Open notebook "AU_Python1_Live_vAttendees.ipynb"



Zen of Python: PEP 20

Python has strong community-driven cultural norms,
akin to how you greet someone, dress for a formal event,
or structure a team decision making meeting

If you are unaware of these norms or break them intentionally you and your work will signal your exceptionalism and unwelcome violation of de facto community standards

If you ever need a reminder of the Zen of Python,
it is near at hand:

```
import this
```

Zen of Python

1. Beautiful is better than ugly
2. Explicit is better than implicit
3. Simple is better than complex
4. Complex is better than complicated
5. Flat is better than nested
6. Sparse is better than dense
7. Readability counts
8. Special cases aren’t special enough to break the rules
9. Although practicality beats purity
10. Errors should never pass silently
11. Unless explicitly silenced
12.
13. In the face of ambiguity, refuse the temptation to guess
13. There should be one—and preferably only one—obvious way to do it
14. Although that way may not be obvious at first unless you're Dutch
15. Now is better than never
16. Although never is often better than *right* now
17. If the implementation is hard to explain, it’s a bad idea
18. If the implementation is easy to explain, it may be a good idea
19. Namespaces are one honking great idea—let's do more of those!

Focused on web development

Strong in statistics & data visualization

Runs faster, Codes slower

Strong in math, proprietary

C# Powerful, but focused on the MS universe

Good for ad-hoc scripts, but not supported by Microsoft anymore

References

Magic methods

- https://rszalski.github.io/magicmethods/

Idiomatic Python

- https://david.goodger.org/projects/pycon/2007/idiomatic/handout.html

For more practice for Python

- https://automatetheboringstuff.com/

Truthiness in Python

- Logic statements are very important in any programming language
  - They control program flow
- In Python, ==, <=, >=, <, >, `is`, `in` all return Boolean values
- `2 == 1 + 1`
- `True`
- `"a" in ["a", "A", "b"]`
- `True`
- To invert a logical statement, use ~

```
~ "a" in "apple"
False
```

Concepts — If-else

If-then-else

A control structure to test values

Uses if, elif or else

Always separated with a ':'

Statements to execute are indented by a tab (or consistent no. of spaces)

```
if name == 'Chrissy':
   print('fun trainer')
elif name == 'John':
   print('other trainer')
else:
   print('student')
```

Concepts — Functions

Functions

Objects that take parameters, and return one or multiple values

Use them by providing the parameters if any, between brackets

e.g. max(a,b)

```
max(1,2)
> 2
sum([3,4,5,6])
> 18
```

Concepts—Lists and Loops

Lists

Define an empty list with []

Several useful functions to apply to a list

- .append(el) adds an element to the back
- len(list) returns the length of the list
- Access an item in a list with list[x], with x being the index (starting at 0)

```
digital_list = ['gamma', 'dv', 'platinion']
digital_list.append('omnia')
digital_list
> ['gamma', 'dv', 'platinion', 'omnia']
digital_list[0]
> 'gamma'
digital_list[-1]
> 'omnia'
```

Concepts—Lists and Loops

Loops

To iterate through every element in myList
You can call the elements anything

for el in myList:
    statements to do something ....

```
for abm in digital_list:
    if abm == 'gamma':
        print('this abm is the best')
    else:
        print(abm)
 > this abm is the best
    dv
    platinion
    omnia
```

Concepts — Dictionaries

Dictionaries

A dictionary captures “key-value” pairs

You can lookup the value if you know the key, instantly

A value can be anything (string, list, dictionary, …)

Initializing: d = {“key” : value}
Setting a new value: d['key2'] = value2

Lookup these useful functions

- Items(), values(), keys(), set_default()

Dictionary vs a List

In a list you need to know where something is to pull it up

In a dictionary, you can give the location a name and not have to know which item it is

Concepts— User Defined Functions and Scope

User Defined Functions

You can define your own functions that take a defined number of variables, and return an output

Functions are incredible powerful to build repetitive functionality

Basic template

```
def function_name (prm_1, ..., prm_n):
  statement 1
  ...
  statement n
```

```
def double(x):
  return x*2

double(4)
> 8
```

Concepts—User defined Functions and Scope

Scope

Variables live where they are “defined”

For example, a function defines a variable called “myvar”. This variable is not accessible outside of the function

Similarly, if there is a “myvar” defined outside of the function, it's value is not changed by the function

```
def double(x):
  return x*2
double(4)
> 8
x
> Error: x is not defined
```

iloc
Integer-location based selection by position

loc
Row selection by label/index or with a boolean

Data selection with pandas DataFrames

iloc indexer syntax:

data.iloc[<row selection> , <column selection>]

List of rows: [0,1,2]	*List of cols: [0,1,2]*
Slice of rows: 2:4	*Slice of cols: 2:4*
Single values: 1	*Single col: 1*

loc indexer syntax:

data.loc[<row selection> , <column selection>]

Index value: 'UK'	*Named col: 'Capital'*
List of labels: ['UK', 'USA']	*List of cols: ['Population', 'Capital']*
Logical index: df['Capital'] == 'Paris'	*Slice of cols: 'Temperature':'Population'*

Data selection with pandas DataFrames

What part of the DataFrame is selected using the following statements?

1 *df.iloc[[1, 2]]*

	Temperature	Population	Capital
UK	8.5	67	London
France	10.7	66	Paris
USA	8.6	326	Washington D.C.
Japan	11.2	127	Tokyo

2 *df.loc[:, ['Temperature']]*

	Temperature	Population	Capital
UK	8.5	67	London
France	10.7	66	Paris
USA	8.6	326	Washington D.C.
Japan	11.2	127	Tokyo

3

	Temperature	Population	Capital
UK	8.5	67	London
France	10.7	66	Paris
USA	8.6	326	Washington D.C.
Japan	11.2	127	Tokyo

4

	Temperature	Population	Capital
UK	8.5	67	London
France	10.7	66	Paris
USA	8.6	326	Washington D.C.
Japan	11.2	127	Tokyo

THANKS

金 融 先 锋　　科 技 向 善