CoGrammar

Python Packages for Data Science

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Data Science Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Data Science Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Lecture Objectives

- **Learn about the multiple Python packages that allow us to perform data analysis easily.**

Python Packages for Data Science

- To recap, data engineering involves data wrangling, cleansing and preparation, and can be defined as, “the process by which you identify, collect, merge, and preprocess one or more datasets in preparation for data cleansing” (Jones 2018).

- Once we’ve wrangled the data into a format we can work with, we clean the data by mending any errors in the data set.

Numpy Python Package

- NumPy stands for Numeric Python. It's and open-source extension module for Python and hence uses Python syntax.
- NumPy brings the power of data structures to Python. Due to its array-oriented programming, it's easy to use NumPy to work with large multidimensional arrays and matrices and carry out scientific computations.

Arrays

- ★ Arrays in Python differ from Lists in the following ways :
  - They only consist of elements of the same data type.
  - They can handle arithmetic operations.
  - They are preferred for larger chunks of data.
- ★ In addition, arrays require proper modules in order to perform operations on them and use to their full potential.

Using NumPy

```
import numpy as np

values = [22.5, 24.9, 27.8, 27.1] # Define a list with floats as values
print(type(values)) # Output : <class 'list'>

# Now we convert into a numpy array
numpy_values = np.array(values)
print(type(numpy_values)) # Output : <class 'numpy.ndarray'>
print(numpy_values)
```

Why use NumPy over Lists?

You might wonder, what is the difference between NumPy arrays and core Python lists? The main benefits include smaller memory consumption, better runtime and ease of manipulation of data compared to Python.

With NumPy arrays we don't have to loop through them!

Examples

```python
import numpy as np

values = [22.5, 24.9, 27.8, 27.1]
numpy_values = np.array(values)
# Let's imagine the values are temperatures in degrees celcius
# and we'd like to convert them to farenheit

# using numpy's scalar multiplication
conversion = numpy_values * 1.8 + 32
print(conversion) # Output : [72.5  76.82 82.04 80.78]
```

Examples

```
# Using core Python

values = [22.5, 24.9, 27.8, 27.1]

conversion = []
for x in values:

    conversion.append(round(x * 1.8 + 32,2))

print(conversion) # Output : [72.5, 76.82, 82.04, 80.78]

# There's a shorthand version, but still uses a loop
conversion = [x * 1.8 + 32 for x in values]
```

Let’s Breathe

Let’s take a small break before moving on to the next topic.

Pandas

- ★ **Pandas is a Python module that contains high-level data structures and tools designed for fast and easy data analysis.**
- ★ **Pandas is built on NumPy and makes it easy to use in NumPy oriented applications, such as data structures with labeled axes.**
- ★ **Pandas also provides for explicit data alignment, which prevents common errors that spawn from misaligned data coming from various sources.**

Pandas

- **Before we delve deeper, it's important to understand the three fundamental pandas data structures :**
  - **Series (one-dimensional)**
  - **DataFrame (two-dimensional)**
  - **Index**

Series

- **A Series is a one-dimensional labelled array that can hold any data type**

Example

```
import numpy as np
import pandas as pd

data = pd.Series([1, 3, 5, np.nan, 6, 8])
print(data)

'''
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
'''
```

DataFrame

- The pandas library documentation defines a DataFrame as a “two-dimensional, size-mutable, with labelled rows and columns”.
- To simplify, think of a DataFrame as a table of data with the following characteristics (Lynn 2018)

Pandas

- ★ There can be multiple rows and columns in the data.
- ★ Each row represents a sample of data.
- ★ Each column contains a different variable that describes the samples.
- ★ The data in every column is usually the same type of data.
- ★ Usually, unlike an excel data set, DataFrames avoid having missing values, and there are no gaps and empty values between rows or columns.

Examples

```
import numpy as np
import pandas as pd

dates = pd.date_range('20220226', periods=6) # periods creates 6 days here

df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))

print(df)
```

Examples

```python
import numpy as np
import pandas as pd

# Or create a DataFrame by passing a dict of objects that can be converted into a series
df = pd.DataFrame({
    'A' : 1.,
    'B' : pd.Timestamp('20220226'),
    'C' : pd.Series(1, index=list(range(4)), dtype='float32'),
    'D' : np.array([3] * 4, dtype='int32'),
    'E' : pd.Categorical(["test", "train", "test", "train"]),
    'F' : 'foo'
})

print(df)
```

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

CoGrammar

Thank you for joining!