Python Review – examples

Dr. Huiping Cao

A little bit advanced

- Main function
- Use arguments
- Pandas package

```
import sys


def main():
        print("This is the name of the program:", sys.argv[0])
        …


if __name__ == '__main__’:
        main()
```

Main function

- Before executing code, Python interpreter reads source file and define few **special variables/global variables**.
- If the python interpreter is running that module (the source file) as the main program, it sets the special **__name__** variable to have a value **“__main__”**.
- If this file is being imported from another module, **__name__** will be set to the **module’s name**. Module’s name is available as value to __name__ global variable.

- Source: https://www.geeksforgeeks.org/what-does-the-if-__name__-__main__-do/

Use arguments – source code

```
#argumenteg.py

import sys

def main():
    print("The number of arguments:", len(sys.argv))
    print("This is the name of the program:", sys.argv[0])
    print("Argument List:", str(sys.argv))

    if len(sys.argv)>=2:
        number = int(sys.argv[2])
        print("File name = ", sys.argv[1])
        print("number=", number, "number*number=", number*number)
        file = open(sys.argv[1], 'r')

if __name__ == '__main__':
    main()
```

Use arguments – run the program from command line

```
$ python argumenteg.py 8ints.txt 3
    The number of arguments: 3
    This is the name of the program: argumenteg.py
    Argument List: ['argumenteg.py', '8ints.txt', '3’]
    File name =  8ints.txt
    number= 3 number*number= 9
```

Reference: https://www.geeksforgeeks.org/how-to-use-sys-argv-in-python/

Pandas – read in a file to a data frame

- Read in a csv file
- Example

```
import pandas as pd

columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
iris_df = pd.read_csv(iris_dataset, names=columns, header=None)

print(iris_df.head())
```

```
sepal_length  sepal_width  petal_length  petal_width       class
0         5.1          3.5           1.4          0.2  Iris-setosa
1         4.9          3.0           1.4          0.2  Iris-setosa
2         4.7          3.2           1.3          0.2  Iris-setosa
3         4.6          3.1           1.5          0.2  Iris-setosa
4         5.0          3.6           1.4          0.2  Iris-setosa
```

Pandas – attributes

- **DataFrame.shape**: Return a tuple representing the dimensionality of the DataFrame.
- Example

```
rows, columns = iris_df.shape

print("rows: {}, columns: {}".format(rows, columns))
```

```
rows: 150, columns: 5
```

Pandas – attributes

- **DataFrame.columns:** The column labels of the DataFrame.
- **DataFrame.dtypes**: Return the dtypes in the DataFrame.
  - This returns a Series with the data type of each column. The result's index is the original DataFrame's columns. Columns with mixed types are stored with the object dtype.
- Example

```
print(iris_df.columns)
print(iris_df.dtypes)
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
'class'], dtype='object')

sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
class           object
dtype: object
```

Get a data series (column) from a data frame

- DataFrame[‘<column name>’]

```
classes = iris_df['class’]
print(classes)
```

```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
           ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
```

Row selection from a data frame

```
setosa_row_index = iris_df['class'] == 'Iris-setosa'
setosa_rows = iris_df[setosa_row_index]
print("setosa_row_index: ", setosa_row_index)
print("setosa_rows: ", setosa_rows)
print("# of setosa_rows: ", len(setosa_rows))
```

setosa_row_index:

```
0      True
1      True
2      True
3      True
4      True
      ...
145    False
146    False
147    False
148    False
149    False
```

setosa_rows:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
…					
…					
49	5.0	3.3	1.4	0.2	Iris-setosa

# of setosa_rows: 50

Aggregation on Panda Data Series

- Aggregation functions: **mean, max, median, min, count, sum, std**, etc.

```
val1 = iris_df['sepal_length'].mean()
print("mean value of sepal_length:", val1)

setosa_df = iris_df[iris_df['class'] == 'Iris-setosa']
print("mean value of Setosa sepal_length:",
setosa_df['sepal_length'].mean())
```

mean value of sepal_length: 5.843333333333335
mean value of Setosa sepal_length: 5.005999999999999

Remove missing values

- DataFrame.**dropna**()

```
df = pd.DataFrame({"name": ['Alfred', 'Batman', 'Catwoman’],
                   "toy": [np.nan, 'Batmobile', 'Bullwhip'],
                    "born": [pd.NaT, pd.Timestamp("1940-04-25"), pd.NaT]})

print(df)
```

```
   name toy born
0 Alfred NaN NaT
1 Batman Batmobile 1940-04-25
2 Catwoman Bullwhip NaT
```

```
df.dropna()
print(df)
```

```
   name toy born
1 Batman Batmobile 1940-04-25
```

Unique/distinct values

- **pandas.unique**(values): Return unique values based on a hash table.
- **set** data type

```
unique_class_values1 = pd.unique(iris_df['class'])
unique_class_values2 = set(iris_df['class'])
print("version1: Unique class values: ", unique_class_values1)
print("version2: Unique class values: ", unique_class_values2)
```

```
version1: Unique class values:  ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
version2: Unique class values:  {'Iris-versicolor', 'Iris-virginica', 'Iris-setosa'}
```

More resources

- Textbook source code: https://github.com/rasbt/python-machine-learning-book-3rd-edition
- Conda: managing Python: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-python.html
- Pandas: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html