INFSCI 2915: Machine Learning Introduction to Python

Mai Abdelhakim

School of Computing and Information
610 IS Building

Spring 2018

Python

- General-purpose programming, interpreted language, popular for many data science applications

- Can Interact with code through terminal

- Installation recommended: **Anaconda** https://www.continuum.io/downloads **Choose Python 3.6**
  - Comes with Numpy, Scipy, Matplotlib, Pandas, Jupyter Notebook, Scikit Learn
  - Includes Spyder: Python development environment
  - Jupyter notebook

Jupyter notebook

- **Jupyter** notebook (http://jupyter.org): interactive web application for coding
  - Incorporate code, text and images in one file
- Supports many programing languages, e.g. Python, R, Julia
  - By separate Kernels
- Start it from anaconda or from the terminal (type jupyter notebook)

Python Libraries

- **Numpy** (http://numpy.org): **multidimensional data** storage and computation
- **SCiPy** (http://scipy.org): **numerical** tools, e.g. interpolation
- **Pandas** (http://pandas.pydata.org): **Dataframe objects**, and tools to manipulate and filter data
- **Matplotlib** (http://matplotlib.org): **visualization**
- **Seaborn**: https://seaborn.pydata.org/
  - Python visualization library based on matplotlib.
- **Scikit-Learn** (http://scikit-learn.org): common **machine learning algorithms**

Python Syntax

- Command line or scripts (.py files), notebooks (.ipynb)
- Comments are indicated by #
  - E.g. # this line is a comment
- The end of line is end of statement no ‘;’ needed
  - E.g. x=5
- Semicolon (;) can be used to separate statements on the same line
  - E.g x=5; y=8
- Printing: built-in print function
  - print(x)
  - print("The value of x is equal to", x)

Python Syntax

- Variables have no attached type information
- Everything is treated as an object
- Example:
  X=10
  type(x) ➔ output: int
- Using python, define two variables: y='Hello' and variable z is set to 3.14
  - get type of y and z?

Everything is an Object

- Objects has attributes and methods accessed by **dot operator**

- Example:
  - List.append(4)
  - x.real (real part of a number), y.imag (imaginary part)

Operations

Arithmetic operations

Operator	Name	Description
a + b	Addition	Sum of a and b
a - b	Subtraction	Difference of a and b
a * b	Multiplication	Product of a and b
a / b	True division	Quotient of a and b
a // b	Floor division	Quotient of a and b, removing fractional parts
a % b	Modulus	Remainder after division of a by b
a ** b	Exponentiation	a raised to the power of b
-a	Negation	The negative of a

A whirlwind Tour of Python, by Jake VanderPlas (available online)

Comparison operations, return true/false

Operation	Description
a == b	a equal to b
a != b	a not equal to b
a < b	a less than b
a > b	a greater than b
a <= b	a less than or equal to b
a >= b	a greater than or equal to b

Lists

- https://docs.python.org/3/tutorial/datastructures.html
- L=[1,2,3], type(L) is list
- **Length** of list using *len:* ***len(L)***
- ***Sum*** *elements in list:* ***sum(L)***
- **Append a single element** to the end of the list using append method:
  - *L.append(4)* #this appends 4 to the end of L
- Addition **concatenates** to the list (can also use extend)
  - L + [5,6,7]
- Can be of any type and of **mixed types**:
  - L2= [1, `two', 3.14, [4,5,6]]
- Indexing:
  - **Starts from Zero**: e.g. L[0], L[1],…
  - You can access end of list can be through negative sign starting -1
    - L=[1,2,3]; Print(L[-1]) => Output: 3

Lists

- Access multiple elements:
  - **List [Start : End : Step]**
    - Start counts from 0
    - End is index of the last element that will **not** be included
  - From the **third** to the **fourth** element: L[**2**:4]
    - Element with index 4 (5th element in the list) will not be included
  - If “End” is not specified, default is end of the list
    - L[2:] is the same as L[2:len(L)]
  - If “Start” is left out, then zero is assumed
    - L[:3] is the same as L[0:3]
  - Step size:
    - Entire list with step size of 2: L[::2]
    - Negative step is possible;
      - L=[1,2,3]; L=[::-1] => output: (3,2,1)

Dictionaries

- https://docs.python.org/3/tutorial/datastructures.html#dictionaries
- Flexible mapping of keys to values
- Can be created by comma-separated list of **{key:value}**
  - Grade={‘Alex’: 10, ’Peter’:15, ‘Games’: 20}
- **Index** is through a valid **key** (not zero-based)
  - print(Grade[‘Peter’])
  - **Order is not important**
- Add new element using a new key
  - Grade[‘Jeff’]=16
- Find keys in a dictionary: **Grade.keys()**
- Find values with **Grades.values()**

Indentation

- Whitespace at the beginning is meaningful
- Code block: statements that should be treated as a unit
- Code blocks are **preceded by a colon ":"**
- Amount of indenting must be consistent in the code (typically 4 spaces)

```
if statement:
        #code block, code without indentation will not be part of it

while condition:
        #code block
```

Functions

- Define you own function using **def**
- **Example:**

```
def myFunctionName(inputArgument1, inputArgument2):
        # code here
         print ‘first input is:’, inputArgument1, ‘second one is:’ , inputArgument1
```

- Call function with name
  myFunctionName(2,4)

Conditional Statements

- Allow a code block to execute only if a condition is satisfied

```
if x==0:
    print (’x is equal to 0’)
elif x>0:
    print(‘x is greater than zero’)
else:
    print(‘x is not greater than or equal to zero’)
```

Boolean Operations

- Combine Boolean values using : **and**, **or** and **not**
  - **Example:**

```
if (x < =9) and (y > 2):
    #code block
If (x>10) or (x%2==0):
```

Loops

- For loops: for repeating a code block a number of times

```
for N in [0,1,2,3]:
    print('N is equal to', N)
```

This can be any list

- **range(n)** is an **object** that **generates sequence** from **0 to n-1**, and is often used in for loops

```
for N in range(4):
    print('N is equal to', N)
```

Loops

- While loop: condition is checked in each iteration
  while condition:
  print('this code block will be executed when condition is satisfied')

- Break the loop entirely using ***break***

- *Example:*
  x=2
  while x<5**:**
  print(x)
  x=x+1

- https://wiki.python.org/moin/WhileLoop

Numpy: Numerical Python

- Multidimentional arrays storage and efficient manipulation
- https://docs.scipy.org/doc/numpy-dev/user/quickstart.html
- Example:

```
import numpy
x=numpy.array( [ [1,2,3] , [4,5,6] ] ) # 2x3 matrix
x.T # transpose of x
x.shape # get dimension of x
type(x) #get type of x numpy.array
x.reshape(3,2)
```

- Efficient element-wise operation on data
  - E.g. x*2 will multiply each element in the array by 2

Pandas

- Enabled labeled interface for multidimensional in the form of DataFrame object
- Labeled column-oriented data
- Example:

```
import pandas as pd
                    # “as pd” gives pandas a short name

dataFrame1=pd.DataFrame({‘labelCol1’:[‘a’,’b’,’c’,’d’], ‘labelCol2’: [1,2,3,4]})
```

- Two ways to access column:
  - **dataFrame1 [‘labelCol2’]**
  - **dataFrame1. labelCol2**

Pandas

- Efficient way to do operations on each column independently

- Example: sum the second column in dataFrame1 (which has label ‘labelCol2’):

  dataFrame1 [‘labelCol2’].sum() #this sums element of elements with label ‘labelCol2’

- Example: Print all people and their age, if they are older than 20 years old

  myDataFrame=pd.DataFrame({‘Name’: [‘John’,’Peter’,’Anna’], ‘Age’: [20, 28, 50]})

  print(**myDataFrame[myDataFrame.Age>20]**)

Pandas

- Can put numpy array into Pandas DataFrame
- Example:

```
import pandas as pd
import numpy as np
x=np.array([[1,2,3],[4,5,6]])
colLabels=['Col1','Col2','Col3']
myDataFrame=pd.DataFrame(x, columns= colLabels )
```

- Column and row labels

```
RowLabel=['r1','r2']
Mydataframe = pd.DataFrame(x, index= RowLabel, columns= colLabels )
Mydataframe.loc['r1','Col2']
```

- Many functions: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html

Matplotlib

- Visualization tool - create plots
- View examples:
  http://matplotlib.org/examples/index.html
- Example:

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(-10,10,100) #100 is the number of samples in the plot
y=np.sin(x)
plt.plot(x, y, marker='o’)
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.title('sin function')
```

- When using Jupyter notebook, at the beginning type:
  %matplotlib inline

Scikit-Learn

- Open source project used in academia and industry
- Number of machine learning algorithms
- http://scikit-learn.org/stable/documentation

Exercise 1

1. Create a dictionary of score of each team. That is keys:values are

   Team1: 4,
   Team2: 3,
   Team3: 5,
   Team4: 2,

2. Add to the dictionary ‘Team5’ who has a score ‘5’
3. PRINT all the keys of the dictionary using the keys() method
4. Find the length VALUES in the dictionary (use the len function) and print it
5. Get the average score of the teams and print it (use the sum and len functions)

Exercise 2

1. Generate a numpy array with 2 columns, where first column contains numbers from 0 to 5 and second column is [0, 1, 4, 9, 16, 25]
2. Check the shape of your array. It should be 6x2.
3. Put the array into a data frame, with column labels ‘x’ and ‘y’
4. Plot ‘x’ versus ‘y’
5. Get average of elements in second column of the data frame