Practical Notes - Intro

TRI-DUC NGHIEM

DUC.NGHIEM@PIPEPREDICT.COM

Overview

Introduction to Python

Numpy, pandas - why do array, tensor matter?

Scikit-learn what a toolkit!

Matplotlib – a useful visualization library

Introduction to Python Syntax – **Basic**

- Famous IDE: spyder, pycharm, Sublime Text, and so on. BUT out of all, Visualstudio Code is the most popular: Visual Studio Code - Code Editing. Redefined
- Your python script file should start with (in unix OS only): **#!/usr/bin/python3**
- Block of code is based on indent level, which each tab equals to 4 whitespaces
- Procedure, function, method are defined as def
  - def my_procedure(my_parameter):
- Main function
  - **if** __name__ == "__main__":

Introduction to Python Syntax – **Basic Operations**

- +, -, *, /, ** (power, exponent), // (floor division), % (modulus)
- &, |, ^ , ~, <<, >>: and, or, xor, not, shift left, shift right
- and, or, not
- ==, != (<>), >, >=, <, <=
- in, not in membership checking
- is, is not checking memory location of the two instances/ variables
- ternary operator:
  - In C-like: a = Boolean_expression? Value_1: value_2
  - In python: a = value_1 if Boolean_expression else: value_2

Introduction to Python
Syntax – **Conditional Expression**

- Condition expression
  - **if** condition_1 **and/or** condition_2:
  - **elif** condition_3:
  - **else**:

Introduction to Python
Data Structure - **List**

- A list is an array of objects or values:
  - a = [1, 2, 3]
- You can add more instance to it:
  - a.append(4)
- You can join two list:
  - a += [4, 5, 6]
- You can remove an instance at its position by:
  - a.pop(3) #remove the 4^{th} element (index starts from 0)

Introduction to Python
Data Structure – **List (Advanced)**

- You can extract a slice from the list:
  - b = a[:5] #first 5 elements
  - b = a[5:] #from the 6th elements to the end of the list
  - b = a[5:10] #from the 6th to (include) the 10th element
- b = a[-1] # the last element
- Copy array, filter and more:
  - b = a # be careful! a and b are now pointed to the same address, changing one will result the same to the other
  - b = [e **for** e **in** a]
  - b = [e **for** e **in** a **if** e % 2] #filter to get only odd numbers from a

Introduction to Python
Data Structure - **tuple**

- A group of instances (values, objects) can be groupped together as a tuple
  - a_tuple = (“I”, “am”, “on”, “the”, “highway”, “to”, “hell”)
  - Looks like a list hah?, it does!
- You can access to the i-th instance by its index a_tuple[i]
- You can convert it to list: list(a_tuple)
- You can assign variables with tuple’s instances:
  - var_1, var_2 = (1,2)
- So what is difference(s) between tuple and list:
  - Tuple is immutable, unchangeable!

Introduction to Python
Data Structure - **Dictionary**

- A dictionary is a data structure holding a set of pairs: key, value
  - a = { “key_1”: value, “key_2”: value_2}
- It can be considered as an object with attributes (keys) and their values
- a.keys() returns a list of keys of a dictionary
- a.items() returns a list of tuples (key,value)

Introduction to Python Syntax - **Loop**

- Loop sucks! It's super slow! But in case you have to use:
- For loop
  - **for** instance **in** instance_list:
  - **for** index **in** range(len(instance_list)):
  - **for** key, value **in** my_dictionary.items():
- While loop
  - **while** condition:
- **break** interrupts the loop, and get out of the cycle
- **continue** next cycle of the loop without caring about the rest of the block

Introduction to Python Syntax – Try Catch Exception

- try:
  - Your block of code
- except Exception:
  - Do your catching/loging error
- except:
  - Do a default catching

Exercises

- Problem 1: Write a function which takes a list of numbers as input, returns a tuple of (min, max) values.
- Problem 2: Write a function which takes a list of numbers as input, returns a tuple of (mean, standard deviation) values.
  - Mean = $\frac{1}{n}\sum_1^n x_i$
  - Std = $\sqrt{\frac{1}{n}\sum_1^N (x_i - mean)^2}$

Introduction to Python Syntax - OOP

- Object Oriented Programming:
  - class MyClassName(ParentClass):
    - def __init__(self, parameters):

      """this is constructor of the class"""

      - self.my_class_attribute = whatever #declare class attribute here, default public
      - self._my_protected_attribute = None #declare protected attribute
      - self.__my_private_attribute = None #declare a private attribute
    - def my_method(self, parameters): # declare a class' method

Introduction to Python
Comments and Documentation

- comment: starts with #
- Block comment locates between: “””here is block comments””” or ‘’’here is block comments’’’ (Be consistent, use only one of them in your style, the first one is more preferred)
- Documentation
  - Try to document your function as:

```
def my_function(param_1, param_2...):
    """How the function works

    Parameters
    ----------
    param_1: type
        What does it do?
    ...
    Returns
    -------
    Type
        Description of output(s)
    """
```

Exercises

- Write a class named array with following attributes:
  - M: number of rows
  - N: number of columns
  - A: the array itself, storing number
- Write a following methods for it, which calculate the corresponded function on the given list of number
  - Constructor with only (m,n)
  - Create_array(input_array)
  - Multiplication(array_2)
  - Transpose
- Call and play with those methods you've just written.

Numpy, pandas - why array, tensor matters?

- Python's loop is slow. Imagine looping over a million rows of a training instance list will take like forever.
- Numpy and pandas provide structures, those can apply computational function to every "row" in parallel
- Widely supported by big guys (tensorflow, pytorch) and almost all toolkits out there

Basic with Numpy

- Installation and Documentation:
  - pip3 install numpy
  - NumPy
- NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides
  - a multidimensional array object,
  - various derived objects (such as masked arrays and matrices),
  - and an assortment of routines for fast operations on arrays, including:
    - mathematical, logical,
    - shape manipulation,
    - sorting, selecting,
    - I/O,
    - discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation
    - and much more.

Numpy in Practice

- NumPy: the absolute basics for beginners — NumPy v1.20 Manual
- NumPy basics — NumPy v1.20 Manual

- First: **import** numpy **as** np
- Then you can do your task with numpy

Basic with Pandas

- Getting started — pandas 1.2.3 documentation (pydata.org)
- Getting started tutorials — pandas 1.2.3 documentation (pydata.org)
- pandas is well suited for many different kinds of data:
  - Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
  - Ordered and unordered (not necessarily fixed-frequency) time series data.
  - Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
  - Any other form of observational / statistical data sets. The data need not be labelled at all to be placed into a pandas data structure

Why pandas?

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets
- Intuitive merging and joining data sets
- Flexible reshaping and pivoting of data sets
- Hierarchical labeling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format
- Time series-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting, and lagging.

Scikit-learn

- scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation (scikit-learn.org)
- It provides a rich library for many machine learning algorithms:
  - Supervised Machine Learning:
    - Classification
    - Regression
  - Unsupervised Machine Learning:
    - Clustering
- as well as tools for preprocessing data:
  - Features extraction, features scaling
  - Dimensionality reduction

Matplotlib – a useful visualization library

- Matplotlib: Python plotting — Matplotlib 3.3.4 documentation
- Tutorial: Tutorials — Matplotlib 3.3.4 documentation