# Python Notes

test: <https://tests4geeks.com/start/myself?ts=python>

openhatch Python <http://bit.ly/intermediate-python-projects>

Algorithms in Python

<http://interactivepython.org/runestone/static/pythonds/Recursion/DynamicProgramming.html>

Stats text <http://www.greenteapress.com/thinkstats/thinkstats.pdf>

<https://content.uploady.com/download/Dougherty-Introduction-to-Econometrics.pdf?f=Dougherty-Introduction-to-Econometrics.pdf&fid=ibORMVijwEZ&p=Ac1Lx6D5xel&m=application%2Fpdf&s=1&u=https%3A%2F%2Fwww.uploady.com%2Fdownload%2FibORMVijwEZ%2F0KjmWVqBYjIyMCnq&tp=remote&t=1454144760&ex=172800&ip=1241397950&h=db3a2c513be468f455c3957faf6c25ff21323a5b>

Textbook <http://www.greenteapress.com/thinkpython/thinkpython.pdf>

<https://www.dataquest.io/mission/1/python-basics>

Textbook Python for Data Science.<https://www.kevinsheppard.com/images/0/09/Python_introduction.pdf>

exercises: <http://www.pythonschool.net/basics/lists-exercises/>

<https://github.com/zhiwehu/Python-programming-exercises/blob/master/100%2B%20Python%20challenging%20programming%20exercises.txt>

<https://github.com/pythonschool/Basics/tree/master/4%20-%20Lists>

<https://github.com/pythonschool/Basics/tree/master/4%20-%20Lists/Additional%20exercises%204>

<http://learnpython.org/en/Multiple_Function_Arguments>

<http://www.practicepython.org/exercise/2015/11/26/15-27-tic-tac-toe-draw.html>

SQL in Pandas (python)

The [**pandasql package**](https://pypi.python.org/pypi/pandasql) allows us to perform queries on dataframes using the [**SQLite syntax**](http://www.sqlite.org/lang.html).

https://pypi.python.org/pypi/pandasql

http://www.sqlite.org/lang.html

strings:

1. in order to print path

print('C:\Users\Desktop\newPics')

Due to \n the output prints ewPics on new line to avoid this append string with r

print(r'C:\Users\Desktop\newPics') # note double quotes don't work with 'r'

2. In order to do quotient, do 5//4. Modulo 5%4. Exponentiation: 5\*\*4

3. a = "tang"

a\*5 # prints 'tangtangtangtangtang'

4.

5. len('tang'); len(a) # prints out 4

6. Convert a variable from type string to list and then append

7. Append elements at end of list and deleting elements from list

8.

## What is the difference between list and tuple in Python?

<http://www.hacksparrow.com/python-difference-between-list-and-tuple.html>

To put it lightly, tuples are lists which can't be edited. Once you create a tuple, you cannot edit it, it is immutable. Lists on the other hand are mutable, you can edit them, they work like the array object in JavaScript or PHP. You can add items, delete items from a list; but you can't do that to a tuple, tuples have a fixed size.

Here is an example demonstrating the mutable nature of lists in Python.

>>> animals = ['cat', 'dog']  
>>> animals  
['cat', 'dog']  
>>> animals.append('mat')  
>>> animals  
['cat', 'dog', 'mat']  
>>> animals[2] = 'bat'  
>>> animals  
['cat', 'dog', 'bat']

An example showing the immutable nature of tuples in Python.

>>> point = (3,7)  
>>> point  
(3, 7)  
>>> point[1]  
7  
>>> point[1] = 4  
Traceback (most recent call last):  
 File "", line 1, in   
TypeError: 'tuple' object does not support item assignment

9. Pandas Library

PANDAS SERIES is a mix of list and dictionary.

Key difference between pandas series and numpy arrays is INDEXING.

In series you can assign data and index names in just one statement (some what like a dictionary).

In numpy arrays you would need two arrays to achieve this.

10. Pandas series: vector operations happen by matching common indexes present in both the series. If an index is present in only one series and not in other then the outcome of any vector operation is NaN. to exclude missing (NaN ) values from result set do( result\_variable.dropna())

11. Pandas SQL

 q = """

        select

            gender, district, sum(aadhaar\_generated)

        from

            aadhaar\_data

        where age > 50

        group by gender, district;

    """

## Example of entire code

**import** pandas  
**import** pandasql  
  
**def aggregate\_query**(filename):  
 # Read in our aadhaar\_data csv to a pandas dataframe. Afterwards, we rename the columns  
 # by replacing spaces with underscores and setting all characters to lowercase, so the  
 # column names more closely resemble columns names one might find in a table.  
  
 aadhaar\_data = pandas.read\_csv(filename)  
 aadhaar\_data.rename(columns = **lambda** x: x.replace(' ', '\_').lower(), inplace=True)  
  
 # Write a query that will select from the aadhaar\_data table how many men and how  
 # many women over the age of 50 have had aadhaar generated for them in each district.  
 # aadhaar\_generated is a column in the Aadhaar Data that denotes the number who have had  
 # aadhaar generated in each row of the table.  
 #  
 # Note that in this quiz, the SQL query keywords are case sensitive.  
 # For example, if you want to do a sum make sure you type 'sum' rather than 'SUM'.  
 #  
  
 # The possible columns to select from aadhaar data are:  
 # 1) registrar  
 # 2) enrolment\_agency  
 # 3) state  
 # 4) district  
 # 5) sub\_district  
 # 6) pin\_code  
 # 7) gender  
 # 8) age  
 # 9) aadhaar\_generated  
 # 10) enrolment\_rejected  
 # 11) residents\_providing\_email,  
 # 12) residents\_providing\_mobile\_number  
 #  
 # You can download a copy of the aadhaar data that we are passing  
 # into this exercise below:  
 # https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/aadhaar\_data.csv  
  
 #men n women over the age of 50 with aadhaar generated in each district  
 q = """  
 select  
 gender, district, sum(aadhaar\_generated)  
 from  
 aadhaar\_data  
 where age > 50  
 group by gender, district;  
 """  
  
 # Execute your SQL command against the pandas frame  
 aadhaar\_solution = pandasql.sqldf(q.lower(), locals())  
 **return** aadhaar\_solution  
  
  
filename = '/Users/varsha/Documents/UdacityDataScience/PythonForDataScience/aadhaar\_data.csv'  
**print** aggregate\_query(filename)