**Python cheatsheet**

|  |  |
| --- | --- |
| **Easter egg** | Import this |
| **Import pandas** | import pandas as pd |
| **Import NumPy** | import numpy as np |
| **Series (a column)** | pd.Series([21, 22, 23, 24], name = ‘age’)  *age = column name* |
| **Select an element (row; from 0 to …)** | table\_name[3]  *Select the 4th row* |
| **Make a custom axis to a Series** |  |
|  |  |
| **Index** is also called an **axis**; each element is called axis label.  Data in columns is called as **values**.  A series ideally should have the same datatype throughout its values (same format for the whole column). |  |
| **Dataframe** | A series is a column and a dataframe has multiple columns |
| **Create a dataframe** |  |
| **Columns and rows in a dataframe (table)** | df.shape  (3, 7)  3 rows and 7 columns |
| **Axis 0**  **Axis 1** | Vertical axis (rows)  Horizontal axis (column names) |
| **Select a column** | df[‘ADDRESS’]  Returns a series |
| **Select several columns** | df[[‘ADDRESS’, ‘CITY’]] |
| **Retrieve axis 1 / column names / information** | df.columns  Index([‘first\_name’, ‘last\_name’, ‘email’], dtype=‘object’) |
| **Retrieve axis 0 / vertical axis information** | df.index  RangeIndex(start=0, stop=3, step=1) or min 0, max 3 (3 rows) |
| **Retrieve a row** |  |
| **Make a column as an index** | E.g. you want to filter by last name. Instead of the default 0, 1, 2 etc. you can make the index as Nield, Scala Morrison.  *df.set\_index(‘last\_name’, inplace = True*) inplace is true so it edits the existing df instead of creating a new one  *df.loc[‘Scala’]* you can now use the last name to search for this row  **df.reset\_index(inplace = True)** reset the axis to default |

**Importing Data**





Read csv as text:



Read csv into as a pandas dataframe:



Import an SQL database:



Read a JSON file:



Read a JSON file as a pandas dataframe:



**Selecting Rows and Columns**

**loc and iloc**

* **loc** works on **labels** assigned to the axis, **iloc** works on **numbers**.
* E.g. if axis is surname then **loc** will work only on the surname like ‘**vanli’** while **iloc** will only work on **numbers**.

**Select** (first two) **rows** df.iloc[0:2] or df.iloc[:2]

**Exclude first row** df.iloc[1:]

**Select all** df.iloc[:]

**All rows and columns 2-3** df.iloc[:, 1:3] you are selecting columns 2 and 3 with indexes 1 and 2

**Select last two columns** df.iloc[:, -2:] count from 0 to -1, -2 etc. from right to left. Select the column you want to have and with : you will select everything to the right

Select the last row df.iloc[-1]

Select all rows and last column df.iloc[:, -1]

If you want to get the first 2 rows you need to select the third index, in this case 2:



**Select specific columns and rows**. In this case you will select column index 0 and column index 2 (first\_name and email). In the second query you will select second row and third column



**Select rows and columns using loc** df.loc[["samiam","thomasnield"], "email"]

**Reset index** df.reset\_index(inplace = True)

**Select values that start with a specific letter** condition = df["username"].str.startswith("s") username = column, s = letter start

df[condition]

**Multiple conditions. AND & OR are & and |** condition = df["username"].str.startswith("s") & df["email"].str.contains("gmail")

df[condition]

**Drop columns and rows**





**Remove columns:**



**Remove columns by selecting specific columns like column 1 and 4:**



**Adding rows and columns (appending)**

**Add a column at the end of the dataframe:**



**Add a column at a specific place:**



**Add a row:**



**Using concat (merge two datasets):**



**Updating data**

**Making a column in caps lock (upper):**



**Updating on a condition:**



**Update row on condition:**



**Unpivoting data (melting):**



Id\_vars = untouched columns

Value\_vars = columns that will be unpivoted

Var\_name = column names will be moved to this column

Value\_name = column values will be moved to this column

