greatlearning
Learning for Life

Python -1

Learning Objectives

- Introduction to Python
- Installation Steps
- Data Types in Python
- Numpy
- Pandas
- Case Study

Python (What and Why ?)

- Python is the most popular programming language & choice for Data Scientist / Data Engineer across the world
- Very rich libraries & functions
- Community support
- Easy to deploy in production
- Support for all the new state of the art technologies (like deep learning)

Installation Steps

Install using the instruction given in the below links -

1. Install Jupyter - http://jupyter.org/install
Preferred installation method is through **Anaconda distribution.**

Install **Python 3.6 version.**

2. Anaconda 5.2 For Linux Installer - https://www.anaconda.com/download/#linux
3. Anaconda 5.2 For macOS Installer - https://www.anaconda.com/download/#macos
4. Anaconda 5.2 For Windows Installer - https://www.anaconda.com/download/#windows

(You need to download the version compatible with your OS)

Common python libraries

- NumPy – handling multi-dimensional arrays
- Scipy – Statistical package
- Matplotlib, seaborn – Visualisation
- Pandas – handling arrays & dataframes

Types of common variables

- Integer
- Float
- String
- Logical

Data Types in Python

greatlearning
Learning for Life

Apart from data types like int, string, float Python has the below data types which are very useful for data science -

1. List
2. Tuples
3. Dictionaries

Python data structures - List vs Tuple vs Dictionary

greatlearning
Learning for Life

- Both list & tuples are ordered sequence of objects
- Both can contain mixed data types
- List is mutable while a tuple is like a list but immutable
- Tuples are faster and consume less memory
- Dictionary list of items in terms of a key and a value

Numpy

greatlearning
Learning for Life

NumPy is the fundamental package for scientific computing with Python.

It contains -

- a powerful N-dimensional array object - ndarray
- sophisticated (broadcasting) functions
- useful linear algebra, Fourier transform, and random number capabilities etc.

Refer - http://www.numpy.org/

Pandas

A library written for the Python programming language for data manipulation and analysis.

In particular, it offers data structures and operations for manipulating numerical tables and dataframes.

Practical example of the usage of dataframe, series & array on a dataset

greatlearning
Learning for Life

Demographic data

Country Name	Birth rate	Internet users	Income Group
Aruba	10.244	78.9	High income
Afghanistan	35.253	5.9	Low income
Angola	45.985	19.1	Upper middle income
Albania	12.877	57.2	Upper middle income
United Arab Emirates	11.044	88	High income

Convert to dataframe

	Country Name	Country Code	Birth rate	Internet users
0	Aruba	ABW	10.244	78.9
1	Afghanistan	AFG	35.253	5.9
2	Angola	AGO	45.985	19.1
3	Albania	ALB	12.877	57.2
4	United Arab Emirates	ARE	11.044	88.0

Extract Birth rate as Pandas Series

```
0     10.244
1     35.253
2     45.985
3     12.877
4     11.044
```

Extract birth rate as numpy array

```
array([10.244, 35.253, 45.985, 12.877, 11.044])
```

Convert dataframe to numpy array

```
array([['Aruba', 'ABW', 10.244000000000002, 78.9],
       ['Afghanistan', 'AFG', 35.253, 5.9],
       ['Angola', 'AGO', 45.985, 19.1],
       ['Albania', 'ALB', 12.877, 57.2],
       ['United Arab Emirates', 'ARE', 11.044, 88.0]], dtype=object)
```

Typical use cases for unique & valuecou

greatlearning
Learning for Life

Mainly used in understanding proportions of levels of a categorical variable

```
employee = pd.read_csv('employee.csv')
employee.columns
```

```
Index(['employee_id', 'department', 'education', 'gender',
       'recruitment_channel', 'no_of_trainings', 'age', 'previous_year_rating',
       'length_of_service', 'avg_training_score', 'attrition'],
      dtype='object')
```

```
employee['recruitment_channel'].unique()
```

```
array(['sourcing', 'other', 'referred'], dtype=object)
```

•**unique** - What are the sources of recruitment in the dataset

•value_count – proportion of attrition in a dataset

```
employee['attrition'].value_counts()
```

```
N    866
Y    259
Name: attrition, dtype: int64
```

```
employee['recruitment_channel'].value_counts()
```

```
other       607
sourcing    494
referred     24
```

•value_count – proportion of employees recruited from each recruitment source

Typical use cases of a python di

- Use whenever a mapping from a key to a value is required
- mapping a field Unique ID to their human-friendly labels

dictionary of genres in a movie database	Sentiment analysis
• {'id': 28, 'name': 'Action'}, {'id': 35, 'name': 'Comedy'}, {'id': 10402, 'name': 'Music'}, {'id': 10751, 'name': 'Family'}, {'id': 12, 'name': 'Adventure'}	• based on dictionary of positive & negative words

Application of group by

- Similar to pivot tables in excel
- What is the average age of employees in each department in the employee dataset?
- What is the highest rating a movie has received in the genre “comedy”?

```
d = emp1.groupby(by='department', axis=0)
```

```
dept_age = round(d.mean(),1)
dept_age = dept_age.sort_values(by='age',ascending=False)
dept_age
```

department	age
Procurement	36.2
Operations	36.1
Technology	35.5
HR	35.4
SalesMarketing	35.3
RandD	34.0
Legal	33.8
Analytics	32.4
Finance	31.1

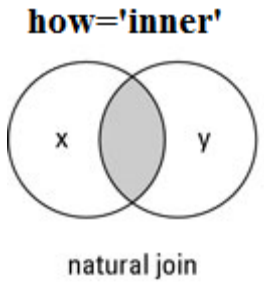
Merge vs Join

- Merge – merges 2 df using a unique column identifier (primary key)
- Join – to join 2 dataframes by the index

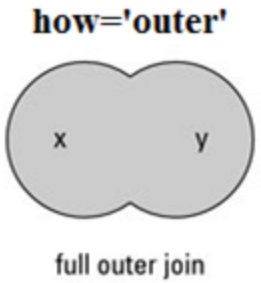
Types of merge

greatlearning
Learning for Life

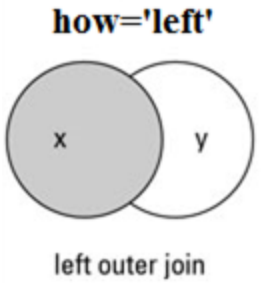
Natural join - Intersection	Full outer join - Union	Left outer join	Right outer join
• To keep only rows that match from the data frames, specify the argument how=**'inner'.**	• To keep all rows from both data frames, specify how=**'outer'.**	• To include all the rows of your data frame x and only those from y that match, specify how=**'left'.**	• To include all the rows of your data frame y and only those from x that match, specify how=**'right'.**



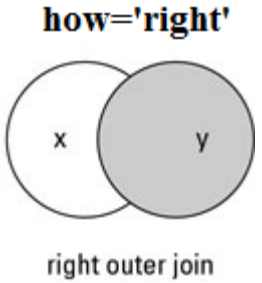
natural join



full outer join



left outer join



right outer join

Uses of Merge & Join

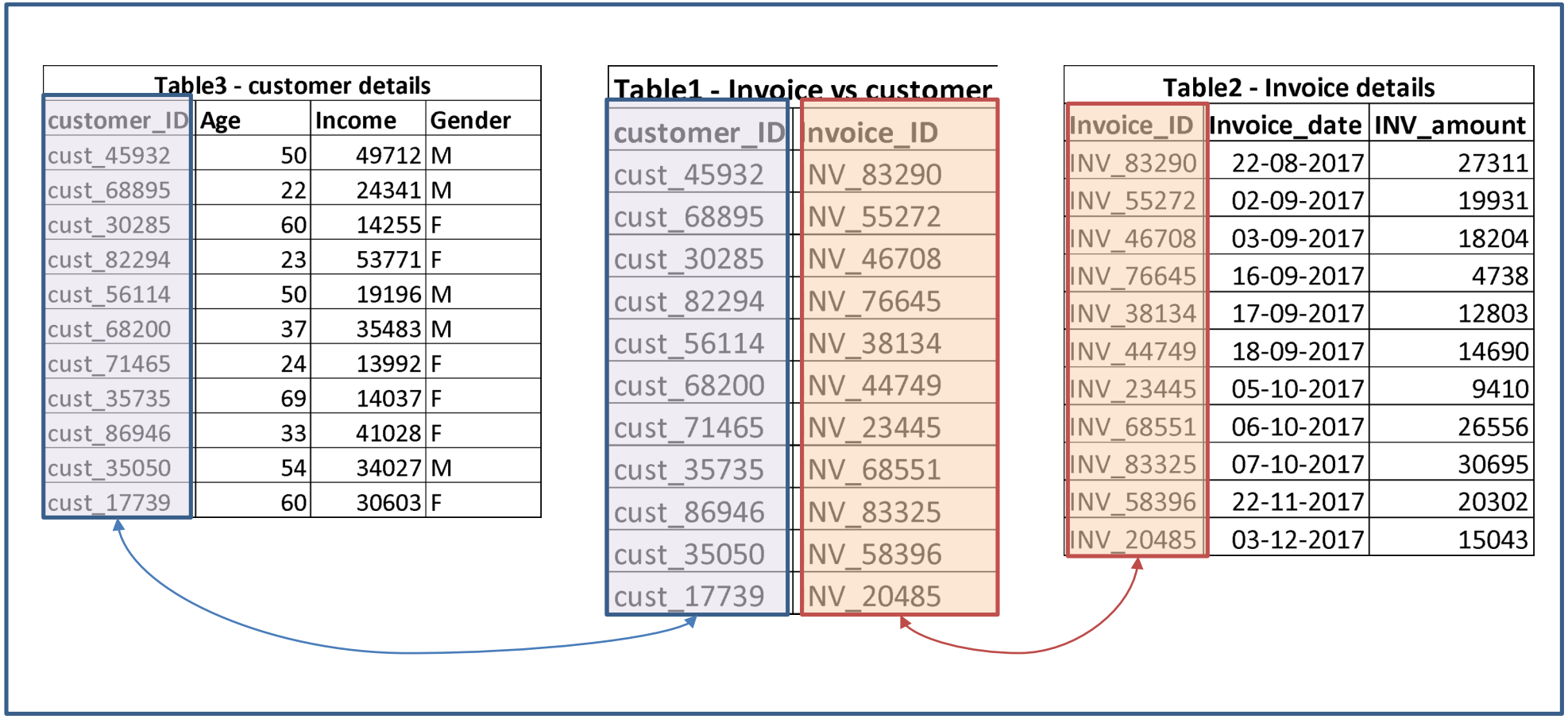


Table3 - customer details

customer_ID	Age	Income	Gender
cust_45932	50	49712	M
cust_68895	22	24341	M
cust_30285	60	14255	F
cust_82294	23	53771	F
cust_56114	50	19196	M
cust_68200	37	35483	M
cust_71465	24	13992	F
cust_35735	69	14037	F
cust_86946	33	41028	F
cust_35050	54	34027	M
cust_17739	60	30603	F

Table1 - Invoice vs customer

customer_ID	nvoice_ID
cust_45932	NV_83290
cust_68895	NV_55272
cust_30285	NV_46708
cust_82294	NV_76645
cust_56114	NV_38134
cust_68200	NV_44749
cust_71465	NV_23445
cust_35735	NV_68551
cust_86946	NV_83325
cust_35050	NV_58396
cust_17739	NV_20485

Table2 - Invoice details

Invoice_ID	Invoice_date	INV_amount
INV_83290	22-08-2017	27311
INV_55272	02-09-2017	19931
INV_46708	03-09-2017	18204
INV_76645	16-09-2017	4738
INV_38134	17-09-2017	12803
INV_44749	18-09-2017	14690
INV_23445	05-10-2017	9410
INV_68551	06-10-2017	26556
INV_83325	07-10-2017	30695
INV_58396	22-11-2017	20302
INV_20485	03-12-2017	15043

customer_ID
Invoice_ID
Invoice_date
INV_amount
Age
Income
Gender

Create a transactional dataframe per customer to analyse purchasing patterns according to customer demography

Case Study

Uber Drive Data Analysis -

The data of a driver's uber trips are available for year 2016.

Your manager wants you to explore this data to give him some useful insights about the trip behaviour of a Uber driver.

Dataset -

The dataset contains Start Date, End Date, Start Location, End Location, Miles Driven and Purpose of drive (Business, Personal, Meals, Errands, Meetings, Customer Support etc.)

Steps

1. Import the libraries
2. Get the data and observe it
3. Check missing values, either remove it or fill it.
4. Get summary of data using python function.
5. Explore the data parameter wise

Here we have information of destination(start and stop), time(start and stop), category and purpose of trip, miles covered.

greatlearning
Learning for Life

HappyLearning

Questions?