Published in CodeX

Anmol Tomar Follow
Dec 5, 2021 · 5 min read · Listen

10 Python operations for 70% of your Data Analysis

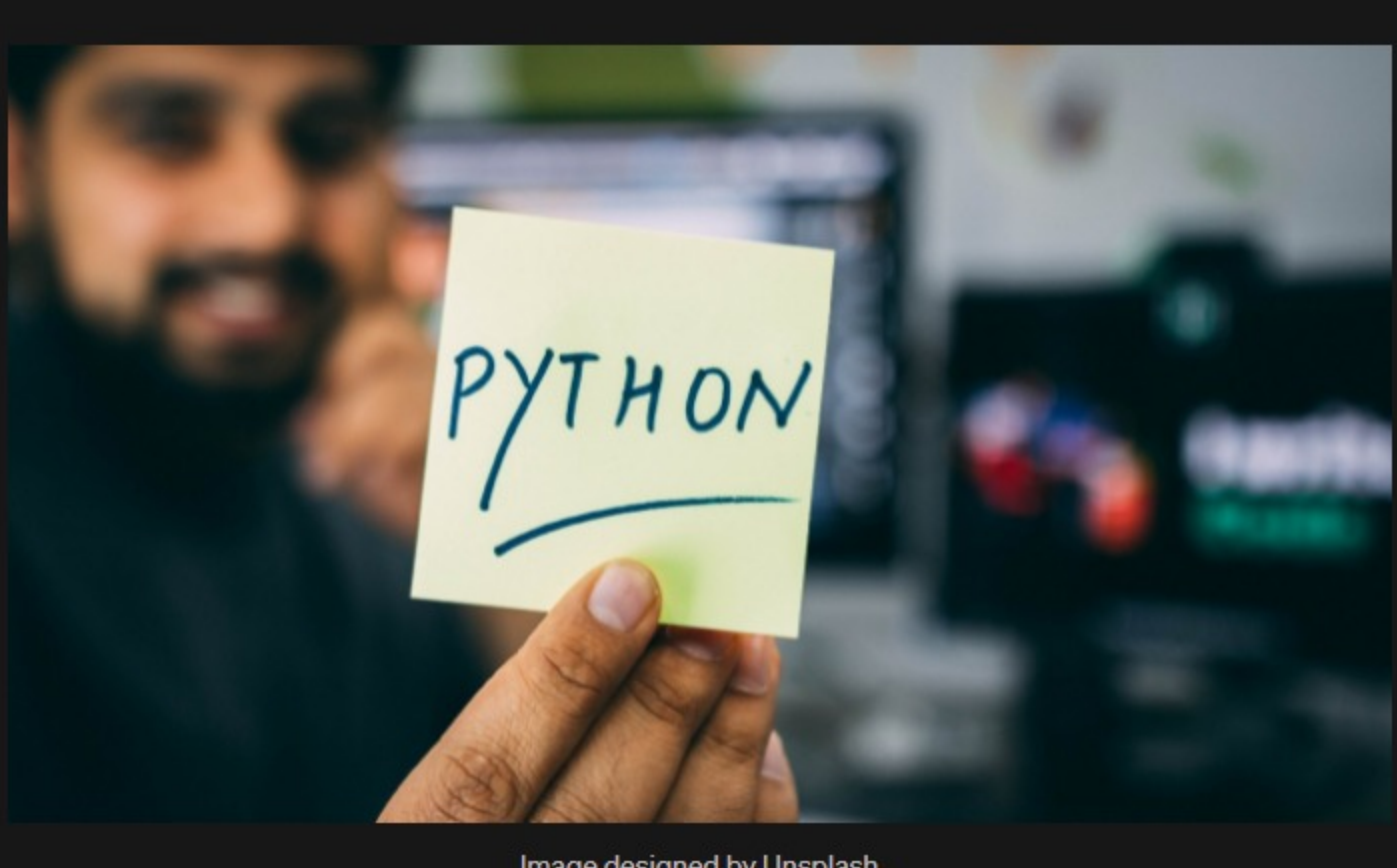

Image designed by Unsplash

Day to day work of a Data Scientist involves a variety of tasks such as data pre-processing, data analysis, machine learning model creation, model deployment.

If you are starting on your journey of becoming a Data Scientist, then the first step is to acquire the skill of data manipulation because it is required in almost every Data Science project. Data Manipulation starts with reading your data and converting it into a form where you can answer your questions from data. Python programming language has Pandas library written for data manipulation and data analysis work.

In this blog, I will give you an overview to top 10 python (pandas) operations that every aspiring Data Scientist should know:

1. **Reading dataset:** Data is the ingredient of every analysis. Knowing how to read data from different file formats such as : csv, excel, text etc is one of the 1st steps that you should be adept at as a Data Scientist. Below is an example of how you can read a csv file, containing Covid-19 data, using pandas.

```
import pandas as pd
# reading the countries_data file along with the location within read_csv function.
countries_df = pd.read_csv('C:/Users/anmol/Desktop/Courses/Python for Data Science/Code/countries_data.
# showing the first 5 rows of the dataframe
countries_df.head()
```

read_csv.py hosted with ❤ by GitHub view raw

Following is the output of countries_df.head() using which we can see the first 5 rows of a dataframe :

	Country	CountryCode	Slug	NewConfirmed	TotalConfirmed	NewDeaths	TotalDeaths	NewRecovered	TotalRecovered	Date	Premium
0	Afghanistan	AF	afghanistan	75	38716	0	1420	404	31638	2020-09-14T18:31:55Z	{}
1	Albania	AL	albania	168	11353	4	334	75	6569	2020-09-14T18:31:55Z	{}
2	Algeria	DZ	algeria	247	48254	7	1612	162	34037	2020-09-14T18:31:55Z	{}
3	Andorra	AD	andorra	0	1344	0	53	0	943	2020-09-14T18:31:55Z	{}
4	Angola	AO	angola	53	3388	2	134	12	1301	2020-09-14T18:31:55Z	{}

2. Summary Statistics : Once you have read the dataset, the next step is to understand the data by looking at the data summaries such as count, mean, standard deviation(std), 25th percentile etc. of numeric columns such as NewConfirmed, TotalConfirmed etc and frequency, top occuring value etc of categorical columns such as Country, CountryCode etc.

Using the describe function we can get the summary of continuous variables of the dataset as shown below :

```
#get summary of continuous variables
countries_df.describe()
```

describe.py hosted with ❤ by GitHub view raw

	NewConfirmed	TotalConfirmed	NewDeaths	TotalDeaths	NewRecovered	TotalRecovered
count	1.880000e+02	1.880000e+02	188.000000	188.000000	188.000000	1.880000e+02
mean	3.527606e+04	1.882256e+05	19.372340	4913.930851	988.175532	1.043737e+05
std	4.664484e+05	8.155154e+05	95.039647	19412.818049	6026.992378	4.349860e+05
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00
25%	1.000000e+00	2.245750e+03	0.000000	37.750000	0.000000	1.205000e+03
50%	5.300000e+01	1.021750e+04	0.000000	208.000000	15.500000	6.556000e+03
75%	4.780000e+02	6.937975e+04	5.250000	1120.000000	263.750000	4.391675e+04
max	6.396100e+06	6.519573e+06	1136.000000	194071.000000	77512.000000	3.780107e+06

Within the describe function, we can set the argument "include = 'all' " to get summaries of both continuous and categorical variables.

```
#get summary of continuous and categorical variables
countries_df.describe(include = 'all')
```

describe_2.py hosted with ❤ by GitHub view raw

	Country	CountryCode	Slug	NewConfirmed	TotalConfirmed	NewDeaths	TotalDeaths	NewRecovered	TotalRecovered	Date	Premium
count	188	187	188	1.880000e+02	1.880000e+02	188.000000	188.000000	188.000000	1.880000e+02	188	188
unique	188	187	188	NaN	NaN	NaN	NaN	NaN	NaN	1	1
top	Montenegro	ES	sierra-leone	NaN	NaN	NaN	NaN	NaN	NaN	2020-09-14T18:31:55Z	{}
freq	1	1	1	NaN	NaN	NaN	NaN	NaN	NaN	188	188
mean	NaN	NaN	NaN	3.527606e+04	1.882256e+05	19.372340	4913.930851	988.175532	1.043737e+05	NaN	NaN
std	NaN	NaN	NaN	4.664484e+05	8.155154e+05	95.039647	19412.818049	6026.992378	4.349860e+05	NaN	NaN
min	NaN	NaN	NaN	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	NaN	NaN
25%	NaN	NaN	NaN	1.000000e+00	2.245750e+03	0.000000	37.750000	0.000000	1.205000e+03	NaN	NaN
50%	NaN	NaN	NaN	5.300000e+01	1.021750e+04	0.000000	208.000000	15.500000	6.556000e+03	NaN	NaN
75%	NaN	NaN	NaN	4.780000e+02	6.937975e+04	5.250000	1120.000000	263.750000	4.391675e+04	NaN	NaN
max	NaN	NaN	NaN	6.396100e+06	6.519573e+06	1136.000000	194071.000000	77512.000000	3.780107e+06	NaN	NaN

> ***Want to get an in-depth understanding of python for Data Analysis ?*** *You can follow the official python documentation or you can enroll into my course on Udemy and become certified in python for Data Analysis. Use the following link to avail 70% discount : https://bit.ly/3148Qq6*

3. Data selection and filtering : Not all the rows and columns of a dataset are required for the analysis. You would need to select the columns of interest and filter some rows based on the question that you are trying to answer.

For example, we can select Country and NewConfirmed columns using the following code :

```
# selecting Country and NewConfirmed columns
countries_df[['Country','NewConfirmed']]
```

selection.py hosted with ❤ by GitHub view raw

	Country	NewConfirmed
0	Afghanistan	75
1	Albania	168
2	Algeria	247
3	Andorra	0
4	Angola	53
...	...	...

We can also filter the data for United States of America as country. Using loc, we can filter a column based on some value as shown below :

```
# filtering USA using country column
countries_df.loc[countries_df['Country'] == 'United States of America']
```

filtering.py hosted with ❤ by GitHub view raw

	Country	CountryCode	Slug	NewConfirmed	TotalConfirmed	NewDeaths	TotalDeaths	NewRecovered	TotalRecovered	Date	Premium
179	United States of America	US	united-states	34450	6519573	378	194071	16748	2451406	2020-09-14T18:31:55Z	{}

4. Aggregation : Finding numeric summaries such as count, sum, mean etc at different variable groupings is data aggregation. It is one of the most frequently performed task of a Data Scientist.

We can find the total of NewConfimed cases across the countries using aggregation. Aggregation is performed using groupby and agg functions. Within groupby function, we provide the level at which we want to perform the aggregation (Country column) and within the aggregation function we provide the column name (NewConfirmed) and mathematical operation (sum) we want to perform on the column.

```
# total NewConfirmed cases across countries
countries_df.groupby(['Country']).agg({'NewConfirmed':'sum'})
```

aggregation.py hosted with ❤ by GitHub view raw

	NewConfirmed
Country	
Afghanistan	75
Albania	168
Algeria	247
Andorra	0
Angola	53
...	...
Viet Nam	3
Western Sahara	0
Yemen	2
Zambia	73
Zimbabwe	18

5. Join : Combining 2 datasets to create one single dataset is done using Join operation. Many times, different information is present in different datasets, for example, one dataset could contain the count of Covid-19 cases across different countries and another dataset could contain the latitude and longitude information of the different countries. Now, if we need to combine these 2 informations then we can perform a join operation as shown below :

```
#reading countires lat and lon data
countries_lat_lon = pd.read_excel('C:/Users/anmol/Desktop/Courses/Python for Data Science/Code/coun

# joining the 2 dataframe : countries_df and countries_lat_lon
# syntax : pd.merge(left_df, right_df, on = 'on_column', how = 'type_of_join')
joined_df = pd.merge(countries_df, countries_lat_lon, on = 'CountryCode', how = 'inner')
joined_df
```

join.py hosted with ❤ by GitHub view raw

	CountryCode	Latitude	Longitude	Country	Slug	NewConfirmed	TotalConfirmed	NewDeaths	TotalDeaths	NewR
0	AF	33.939110	67.709953	Afghanistan	afghanistan	75	38716	0	1420	
1	AL	41.153332	20.168331	Albania	albania	168	11353	4	334	
2	AO	-11.202692	17.873887	Angola	angola	53	3388	2	134	

6. Built-in functions : Knowing the mathematical built-in functions such as min(), max(), mean(), sum() etc is very helpful for performing different analysis. We can apply these functions directly on a dataframe simply by calling them. These functions can be used standalone on a column or within the aggregation function as shown below :

```
# finding sum of NewConfirmed cases of all the countries
countries_df['NewConfirmed'].sum()
# Output : 6,631,899

# finding the sum of NewConfirmed cases across different countries
countries_df.groupby(['Country']).agg({'NewConfirmed':'sum'})

# Output
#           NewConfirmed
#Country
#Afghanistan    75
#Albania       168
#Algeria       247
#Andorra         0
#Angola         53
```

build-in-functions.py hosted with ❤ by GitHub view raw

7. User defined functions : Functions that we write on our own are user-defined functions. We can execute the codes within these functions, whenever needed, by calling that function. For example, we can create a function to add 2 numbers as shown below :

```
# User defined function is created using 'def' keyword, followed by function definition - 'addition
# and 2 arguments num1 and num2
def addition(num1, num2):
    return num1+num2

# calling the function using function name and providing the arguments
print(addition(1,2))
#output : 3
```

user-defined function.py hosted with ❤ by GitHub view raw

8. Pivot : Pivoting is converting the unique values within the rows of a column into multiple new columns. This is advance data manipulation technique. Using pivot_table() function on the Covid-19 dataset, we can convert the country names into individual new columns :

```
# using pivot_table to convert values within the Country column into individual columns and
# filling the values corresponding to these columns with numeric variable - NewConfimed
pivot_df = pd.pivot_table(countries_df,  columns = 'Country', values = 'NewConfirmed')
pivot_df
```

pivot.py hosted with ❤ by GitHub view raw

Country	Afghanistan	Albania	Algeria	Andorra	Angola	Antigua and Barbuda	Argentina	Armenia	Australia	Austria	...	United Kingdom
NewConfirmed	75	168	247	0	53	0	9056	187	41	463	...	3338

9. Iterating over dataframe : Many times it is required to iterate through the index and rows of a data frame. We can iterate through the dataframe using iterrows function :

```
# iterating over the index and row of a dataframe using iterrows() function
for index, row in countries_df.iterrows():
    print('Index is ' + str(index))
    print('Country is '+ str(row['Country']))

# Output :
# Index is 0
# Country is Afghanistan
# Index is 1
# Country is Albania
# .......
```

iterating_over_df.py hosted with ❤ by GitHub view raw

10. String operations : Many times we deal with string columns in our dataset, in such cases it is important to know some basic string operations such as how to convert a string into upper case, lower case and how to find the length of a string in a column.

```
# country column to upper case
countries_df['Country_upper'] = countries_df['Country'].str.upper()

# country column to lower case
countries_df['CountryCode_lower']=countries_df['CountryCode'].str.lower()

# finding length of characters in the country column
countries_df['len'] = countries_df['Country'].str.len()

countries_df.head()
```

strings.py hosted with ❤ by GitHub view raw

NewDeaths	TotalDeaths	NewRecovered	TotalRecovered	Date	Premium	Country_upper	CountryCode_lower	len
0	1420	404	31638	2020-09-14T18:31:55Z	{}	AFGHANISTAN	af	11
4	334	75	6569	2020-09-14T18:31:55Z	{}	ALBANIA	al	7
7	1612	162	34037	2020-09-14T18:31:55Z	{}	ALGERIA	dz	7
0	53	0	943	2020-09-14T18:31:55Z	{}	ANDORRA	ad	7
2	134	12	1301	2020-09-14T18:31:55Z	{}	ANGOLA	ao	6

Knowing how to perform these 10 operations will cater to almost 70% of your data manipulation needs.