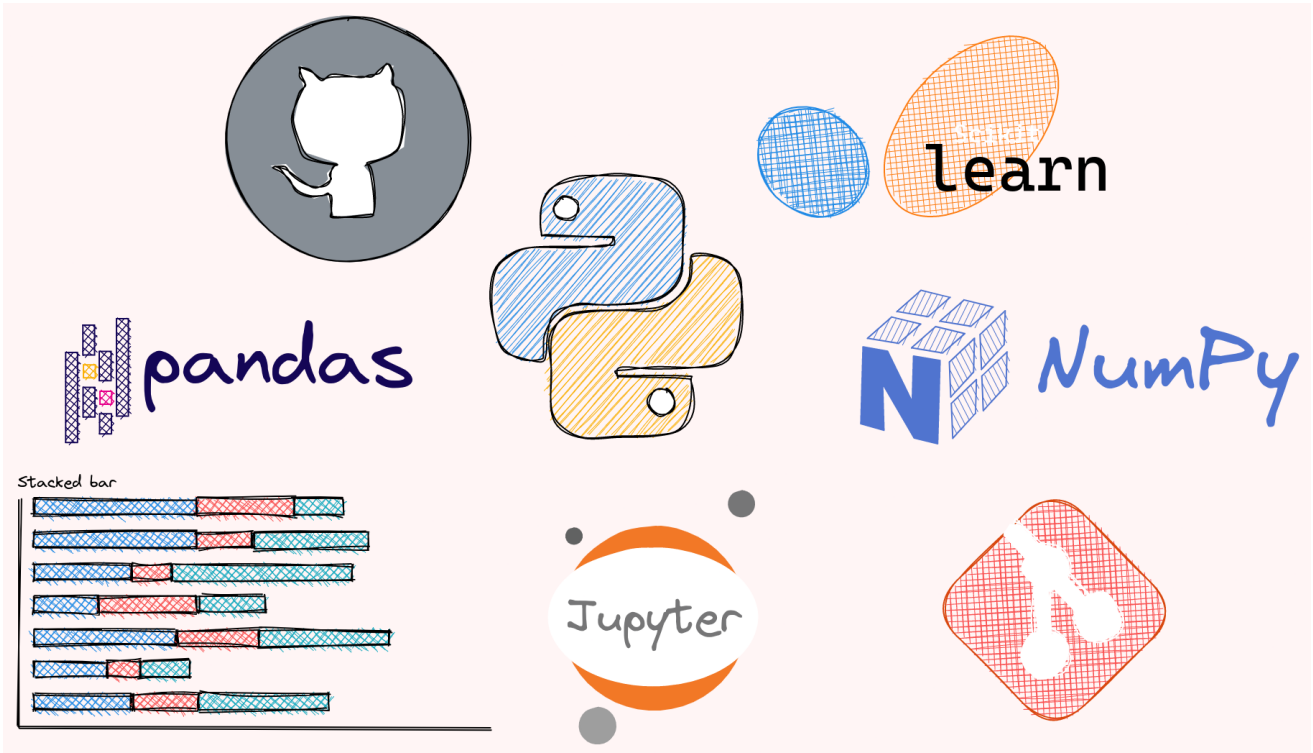
Efficient Python Tricks and Tools for Data Scientists

By Khuyen Tran



Create a New DataFrame Using Existing DataFrame

This section covers some pandas methods to use an existing DataFrame to create a new DataFrame with different functionalities.

pandas.DataFrame.agg: Aggregate over Columns or Rows Using Multiple Operations

If you want to aggregate over columns or rows using one or more operations, try `pd.DataFrame.agg`.

```
from collections import Counter
import pandas as pd


def count_two(nums: list):
    return Counter(nums)[2]


df = pd.DataFrame({"coll": [1, 3, 5], "col2": [2, 4, 6]})
df.agg(["sum", count_two])
```

	coll	col2
sum	9	12
count_two	0	1

pandas.DataFrame.agg: Apply Different Aggregations to Different Columns

If you want to apply different aggregations to different columns, insert a dictionary of column and aggregation methods to the `pd.DataFrame.agg` method.

```
import pandas as pd

df = pd.DataFrame({"a": [1, 2, 3, 4], "b": [2, 3, 4, 5]})

df.agg({"a": ["sum", "mean"], "b": ["min", "max"]})
```

	a	b
sum	10.0	NaN
mean	2.5	NaN
min	NaN	2.0
max	NaN	5.0

Assign Name to a Pandas Aggregation

By default, aggregating a column returns the name of that column.

```
import pandas as pd

df = pd.DataFrame({"size": ["S", "S", "M",
"L"], "price": [2, 3, 4, 5]})

print(df.groupby('size').agg({'price':
'mean'}))
```

```
      price
size
L       5.0
M       4.0
S       2.5
```

If you want to assign a new name to an aggregation, add `name = (column, agg_method)` to `agg`.

```
df.groupby('size').agg(mean_price=('price',
'mean'))
```

	mean_price

size	mean_price

size	
L	5.0
M	4.0
S	2.5

pandas.pivot_table: Turn Your DataFrame Into a Pivot Table

A pivot table is useful to summarize and analyze the patterns in your data. If you want to turn your DataFrame into a pivot table, use `pandas.pivot_table`.

```
import pandas as pd

df = pd.DataFrame(
    {
        "item": ["apple", "apple", "apple", "apple", "apple"],
        "size": ["small", "small", "large", "large", "large"],
        "location": ["Walmart", "Aldi", "Walmart", "Aldi", "Aldi"],
        "price": [3, 2, 4, 3, 2.5],
    }
)

df
```

	item	size	location	price
0	apple	small	Walmart	3.0

	item	size	location	price
1	apple	small	Aldi	2.0
2	apple	large	Walmart	4.0
3	apple	large	Aldi	3.0
4	apple	large	Aldi	2.5

```
pivot = pd.pivot_table(
    df, values="price", index=["item",
"size"], columns=["location"], aggfunc="mean"
)
pivot
```

	location	Aldi	Walmart
item	**size**		
apple	**large**	2.75	4.0
	small	2.00	3.0

DataFrame.groupby.sample: Get a Random Sample of Items from Each Category in a Column

If you want to get a random sample of items from each category in a column, use `pandas.DataFrame.groupby.sample`.This method is useful when you want to get a subset of a DataFrame while keeping all categories in a column.

```
import pandas as pd

df = pd.DataFrame({"col1": ["a", "a", "b", "c", "c", "d"], "col2": [4, 5, 6, 7, 8, 9]})
df.groupby("col1").sample(n=1)
```

	col1	col2
0	a	4
2	b	6
4	c	8
5	d	9

To get 2 items from each category, use `n=2`.

```
df = pd.DataFrame(
    {
        "col1": ["a", "a", "b", "b", "b", "c",
"c", "d", "d"],
        "col2": [4, 5, 6, 7, 8, 9, 10, 11,
12],
    }
)
df.groupby("col1").sample(n=2)
```

	col1	col2
0	a	4
1	a	5
4	b	8
2	b	6
5	c	9
6	c	10
8	d	12
7	d	11

pandas.melt: Unpivot a DataFrame

If you want to unpivot a DataFrame from wide to long format, use pandas.melt.

For example, you can use pandas.melt to turn multiple columns (`Aldi`, `Walmart`, `Costco`) into values of one column (`store`).

```
import pandas as pd

df = pd.DataFrame(
    {"fruit": ["apple", "orange"], "Aldi": [1, 2], "Walmart": [3, 4], "Costco": [5, 6]}
)
df
```

	fruit	Aldi	Walmart	Costco
0	apple	1	3	5
1	orange	2	4	6

```
df.melt(id_vars=["fruit"], value_vars=["Aldi", "Walmart", "Costco"], var_name='store')
```

---

	fruit	store	value
0	apple	Aldi	1
1	orange	Aldi	2
2	apple	Walmart	3
3	orange	Walmart	4
4	apple	Costco	5
5	orange	Costco	6

pandas.crosstab: Create a Cross Tabulation

Cross tabulation allows you to analyze the relationship between multiple variables. To turn a pandas DataFrame into a cross tabulation, use `pandas.crosstab`.

```
import pandas as pd

network = [
    ("Ben", "Khuyen"),
    ("Ben", "Josh"),
    ("Lauren", "Thinh"),
    ("Lauren", "Khuyen"),
    ("Khuyen", "Josh"),
]

# Create a dataframe of the network
friends1 = pd.DataFrame(network, columns=
["person1", "person2"])

# Reverse the order of the columns
friends2 = pd.DataFrame(network, columns=
["person2", "person1"])

# Create a symmetric dataframe
```

```
friends = pd.concat([friends1, friends2])

# Create a cross tabulation
print(pd.crosstab(friends.person1,
friends.person2))
```

```
person2  Ben  Josh  Khuyen  Lauren  Thinh
person1
Ben        0     1       1       0      0
Josh       1     0       1       0      0
Khuyen     1     1       0       1      0
Lauren     0     0       1       0      1
Thinh      0     0       0       1      0
```