Lab 03: Fun with `pandas`!

Below are some exercises to get you working with `pandas` to manipulate data. As always, get as far as you can, and ask for help when you need it! Your teacher (me), you instructor, and your classmates are all here to help each other get better at coding. Getting the code to work is important, but do also take the time to make sure you understand what the commands are doing. This time, (with the exception of the Stroop challenge), all I've given you is the code to download the data. Then you are on your own. For the Stroop challenge, I gave the you code for the first step—after that, it's up to you :-)

Music sales challenge

Write a script that:

1. Combines the tables of best-selling physical singles and best-selling digital singles on the Wikipedia page "List_of_best-selling_singles"
2. Adds a column which marks whether each row is from the list of physical singles or digital singles
3. Outputs the artist and single name for the year you were born. If there is no entry for that year, take the closest year after you were born.
4. Outputs the artist and single name for the year you were 15 years old.

In [4]:
```
# Starter code...

import pandas as pd

rawdata = pd.read_html("https://en.wikipedia.org/wiki/List_of_best-selling_singles")
```

In [5]:
```
# Physical
df_P = rawdata[0]
df_P["Category"] = "Physical"
# Digital
df_D = rawdata[3]
df_D["Category"] = "Digital"

df = pd.concat([df_P, df_D])
# Closest thing to 2003
df[df["Released"] == 2008]

# Birthdate + given age
df[df["Released"] == 2003 + 15]
```

Out[5]:

	Artist	Single	Released	Sales (in millions)	Source	Category
12	Lil Nas X featuring Billy Ray Cyrus	"Old Town Road"	2018	18.4[a]	[56]	Digital
20	Drake	"God's Plan"	2018	15.3[a]	[49]	Digital

Space challenge

1. Make a single dataframe that combines the space missions from the 1950's to the 2020's
2. Write a script that returns the year with the most launches
3. Write a script that returns the most common month for launches
4. Write a script that ranks the months from most launches to fewest launches

In [6]:
```
#1. Make a single dataframe that combines the space missions from the 1950's to the 2020's


rawdata = pd.read_html("https://en.wikipedia.org/wiki/Timeline_of_Solar_System_exploration")

def CombineDF(rawdata, min, max, column_name="Decade", start_decade=1950, increment=10):
    df_list = []

    for x in range(min, max):
        rawdata[x][column_name] = start_decade + (increment * x)
        df_list.append(rawdata[x])

    df = pd.concat(df_list, ignore_index=True)

    return df
df = CombineDF(rawdata, 0, 8)

df
```

Out[6]:

	Mission name	Launch date	Description	Ref(s)	Decade
0	Sputnik 1	4 October 1957	First Earth orbiter	[1][2]	1950
1	Sputnik 2	3 November 1957	Earth orbiter, first animal in orbit, a dog na...	[2][3][4]	1950
2	Explorer 1	1 February 1958	Earth orbiter; discovered Van Allen radiation ...	[5]	1950
3	Vanguard 1	17 March 1958	Earth orbiter; oldest spacecraft still in Eart...	[6]	1950
4	Luna 1	2 January 1959	First lunar flyby (attempted lunar impact?); f...	[7][8][9][10]	1950
...	...	...	...	...	...
230	DRO A/B	13 March 2024	Lunar orbiters	[506]	2020
231	Queqiao-2 (including Tiandu-1 and 2)	20 March 2024	Lunar orbiters	[507]	2020
232	Chang'e 6 (including Pakistan's ICUBE-Q cubesat)	3 May 2024	Lunar sample return, rover and orbiters; first...	[508][509]	2020
233	Hera (3 orbiters)	7 October 2024	Asteroid 65803 Didymos rendezvous	[510]	2020
234	Europa Clipper	14 October 2024	Jupiter orbiter, Europa multiple flyby	[511][512][513]	2020

235 rows × 5 columns

In [7]:
```
# 2. Write a script that returns the year with the most launches

row_counts = df.groupby("Decade").size()

print(row_counts)
```

```
Decade
1950     8
1960    73
1970    42
1980    14
1990    21
2000    24
2010    28
2020    25
dtype: int64
```

In [8]:
```
#3. Write a script that returns the most common month for launches

df["Month"] = df["Launch date"].str.split().str[1]

month_counts = {}

for month in df["Month"]:
    if month in month_counts:
        month_counts[month] += 1
    else:
        month_counts[month] = 1

month_count_list = []


for month, count in month_counts.items():

    month_count_list.append((month, count))

month_count_list
```

Out[8]:
```
[('October', 24),
 ('November', 30),
 ('February', 14),
 ('March', 15),
 ('January', 19),
 ('September', 22),
 ('April', 13),
 ('August', 27),
 ('July', 21),
 ('May', 17),
 ('June', 14),
 ('December', 19)]
```

In [9]:
```
#4. Write a script that ranks the months from most launches to fewest launches

month_count_list.sort(key=lambda x: x[1], reverse=True)

month_count_list
```

Out[9]:
```
[('November', 30),
 ('August', 27),
 ('October', 24),
 ('September', 22),
 ('July', 21),
 ('January', 19),
 ('December', 19),
 ('May', 17),
 ('March', 15),
 ('February', 14),
 ('June', 14),
 ('April', 13)]
```

Supervillain challenge

1. Write a script that combines the tables showing supervillain debuts from the 30's through the 2010's
2. Write a script that ranks each decade in terms of how many supervillains debuted in that decade
3. Write a script that ranks the different comics companies in terms of how many supervillains they have, and display the results in a nice table (pandas dataframe)

In [10]:
```
#1. Write a script that combines the tables showing supervillain debuts from the 30's through the 2010's

rawdata = pd.read_html("https://en.wikipedia.org/wiki/List_of_comic_book_supervillain_debuts")


df = CombineDF(rawdata, 1,9, start_decade= 1930)
df
```

Out[10]:

	0	1	Decade	Character / Team	Year Debuted	Company	Creator/s	First Appearance
0	NaN	This article includes a list of general refere...	1940	NaN	NaN	NaN	NaN	NaN
1	NaN	This article needs additional citations for ve...	1950	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	1960	Ultra-Humanite	1939 (June)	DC	Jerry Siegel, Joe Shuster	Action Comics (vol. 1) #13
3	NaN	NaN	1960	Dr. Death	1939 (July)	DC	Bob Kane, Bill Finger	Detective Comics (vol. 1) #29
4	NaN	NaN	1960	The Monk	1939 (September)	DC	Bob Kane, Bill Finger	Detective Comics (vol. 1) #31
...	...	...	...	...	...	...	...	...
491	NaN	NaN	2010	Crossbones	1989	Marvel	Mark Gruenwald, Kieron Dwyer	Captain America #360
492	NaN	NaN	2010	Mud Pack	1989	DC	NaN	Detective Comics #604
493	NaN	NaN	2010	Artemiz	1989	DC	John Ostrander	Suicide Squad #35
494	NaN	NaN	2010	Anarky	1989	DC	Alan Grant, Norm Breyfogle	Detective Comics #608
495	NaN	NaN	2010	Assembly of Evil	1989	Marvel	NaN	Acts of Vengeance

496 rows × 8 columns

In [11]:
```
#2. Write a script that ranks each decade in terms of how many supervillains debuted in that decade
df["Decade"].value_counts()
```

Out[11]:
```
Decade
1990    228
2000     97
2010     92
1970     47
1980     26
1960      4
1940      1
1950      1
Name: count, dtype: int64
```

In [12]:
```
#3. Write a script that ranks the different comics companies in terms of how many supervillains they have, and display the results in a nice table (pandas dataframe)
company_counts = df["Company"].value_counts().reset_index()
company_counts.columns = ["Company", "Count"]
company_counts
```

Out[12]:

	Company	Count
0	DC	241
1	Marvel	239
2	Fawcett Comics/DC	6
3	Marvel/Timely	4
4	Lev Gleason Publications	1
5	Comico	1
6	Mirage	1

Stroop challenge

Every year between 2015 and 2021, the students in my Language, Cognition, and the Brain course participated in a version of the Stroop task. Using a stopwatch (ok, using their phones), they recorded how fast they could say a list of things (either reading or naming colors or color words). The column names mean "Reading with No Interference", "Naming with Interference", "Naming with No Interference", and "Reading with Interference". The times are in seconds.

Stroop challenge 1:

Transform these data from wide format to long format, so that the result is a dataframe with

- 1 column named "Participant_id" with a unique number for each participant (you can use the row indices)
- 1 column named "Year" with the year data
- 1 column named "Task" that shows which task they were doing
- 1 column named "RT" that shows their response time

In [13]:
```
df = pd.read_csv("https://raw.githubusercontent.com/ethanweed/Stroop/master/Stroop-raw-over-the-years.csv")
df.head()
```

Out[13]:

	Reading_NoInt	Naming_Int	Naming_NoInt	Reading_Int	Year
0	4.16	6.76	4.45	4.65	2015
1	4.35	7.73	4.78	4.46	2015
2	3.60	7.00	4.00	3.50	2015
3	3.90	9.03	4.60	6.30	2015
4	4.22	9.98	6.83	6.24	2015

In [14]:
```
#1) - 1 column named "Participant_id" with a unique number for each participant (you can use the row indices)

# Hello, decided to use my own code!

df["P_ID"] = range(1, len(df) + 1)

df
```

Out[14]:

	Reading_NoInt	Naming_Int	Naming_NoInt	Reading_Int	Year	P_ID
0	4.16	6.76	4.45	4.65	2015	1
1	4.35	7.73	4.78	4.46	2015	2
2	3.60	7.00	4.00	3.50	2015	3
3	3.90	9.03	4.60	6.30	2015	4
4	4.22	9.98	6.83	6.24	2015	5
...	...	...	...	...	...	...
177	4.30	7.08	6.25	4.28	2021	178
178	4.75	9.66	6.12	5.49	2021	179
179	4.98	7.52	6.73	5.16	2021	180
180	5.16	8.81	8.19	5.51	2021	181
181	4.27	10.40	5.32	4.59	2021	182

182 rows × 6 columns

In [15]:
```
#2) - 1 column named "Year" with the year data
#3) - 1 column named "Task" that shows which task they were doing
#4) - 1 column named "RT" that shows their response time
df_long = pd.melt(df,
                  id_vars=["P_ID", "Year"],
                  value_vars=["Reading_NoInt", "Naming_Int", "Naming_NoInt", "Reading_Int"],
                  var_name="Task",
                  value_name="RT")

df_long
```

Out[15]:

	P_ID	Year	Task	RT
0	1	2015	Reading_NoInt	4.16
1	2	2015	Reading_NoInt	4.35
2	3	2015	Reading_NoInt	3.60
3	4	2015	Reading_NoInt	3.90
4	5	2015	Reading_NoInt	4.22
...	...	...	...	...
723	178	2021	Reading_Int	4.28
724	179	2021	Reading_Int	5.49
725	180	2021	Reading_Int	5.16
726	181	2021	Reading_Int	5.51
727	182	2021	Reading_Int	4.59

728 rows × 4 columns

In [16]:
```
# Sorting
df_long_sorted = df_long.sort_values(by="P_ID")
df_long_sorted
```

Out[16]:

	P_ID	Year	Task	RT
0	1	2015	Reading_NoInt	4.16
546	1	2015	Reading_Int	4.65
364	1	2015	Naming_NoInt	4.45
182	1	2015	Naming_Int	6.76
365	2	2015	Naming_NoInt	4.78
...	...	...	...	...
544	181	2021	Naming_NoInt	8.19
545	182	2021	Naming_NoInt	5.32
727	182	2021	Reading_Int	4.59
363	182	2021	Naming_Int	10.40
181	182	2021	Reading_NoInt	4.27

728 rows × 4 columns

In [17]:
```
# Safe checking
# 4 categories, 182 participants
print("728 is 4 times as big as 182 -", 4 * 182 == 728)
```

```
728 is 4 times as big as 182 - True
```

Stroop challenge 2 (Advanced!!!):

Make a new dataframe which shows the mean response time (in seconds) for each task for each year.

In [18]:
```
# Already sorted out, somehow?
df_year_eval = df_long.groupby(["P_ID", "Task", "Year"])["RT"].mean().reset_index()
df_year_eval
```

Out[18]:

	P_ID	Task	Year	RT
0	1	Naming_Int	2015	6.76
1	1	Naming_NoInt	2015	4.45
2	1	Reading_Int	2015	4.65
3	1	Reading_NoInt	2015	4.16
4	2	Naming_Int	2015	7.73
...	...	...	...	...
723	181	Reading_NoInt	2021	5.16
724	182	Naming_Int	2021	10.40
725	182	Naming_NoInt	2021	5.32
726	182	Reading_Int	2021	4.59
727	182	Reading_NoInt	2021	4.27

728 rows × 4 columns