Lecture 2: In-class activity

First, get Jupyter notebook up and running.

If you've made it here, you've done this step!

Second, familiarize yourself with python.

As a first pass, you may want to play around with the examples in the cells below. Hit Shift+Enter to evaluate a cell.

```
print("Hello World!")  # Hello world is really easy in python!
```

```
Hello World!
```

```
A = [6,4,3,8,5] # A is a list.
print(A) # you can print it out!
```

```
[6, 4, 3, 8, 5]
```

```
A[0] # lists are zero-indexed.
```

```
6
```

```
# slicing up lists:
print(A[2:4]) # this is the list [A[2],A[3]] (it doesn't include A[4])
print(A[2:])  # this notation starts with A[2] and goes to the end
print(A[:4])  # this starts at the beginning and goes up until A[3]
print(A[:])   # this just returns a copy of the whole list
```

```
[3, 8]
[3, 8, 5]
[6, 4, 3, 8]
[6, 4, 3, 8, 5]
```

```
len(A) # get the length of a list
```

```
5
```

```
A.append(7) # this appends "7" to A
print(A)
# what happens if you evaluate this cell multiple times?
```

```
[6, 4, 3, 8, 5, 7]
```

```
A = A[:5] # let's set A back to how it was.
print(A)
```

```
[6, 4, 3, 8, 5]
```

```
A = A + ["cat"]  # Python is totally cool with this
print(A)
```

```
[6, 4, 3, 8, 5, 'cat']
```

```
A = [6,4,3,8,5]
for x in A:  # we can iterate over items in a list to get a for loop
    print(2*x)
```

```
12
8
6
16
10
```

```
# Notice that there's no {} or ; or anything like that.
#Python uses the whitespace to tell what's in the loop and what's not.

for x in A:
    print(3*x)
print("This is outside the loop")

print("---")

for x in A:
    print(3*x)
    print("This is inside the loop")
```

```
18
12
9
24
15
This is outside the loop
---
18
This is inside the loop
12
This is inside the loop
9
This is inside the loop
24
This is inside the loop
15
This is inside the loop
```

```
T = range(5)  # the range function gives you a way to iterate over a range of i
for x in T:
    print(x)
```

```
0
1
2
3
4
```

```
for i in range(5):  # we can also use the range function to iterate over A
    print(2*A[i])
```

```
12
8
6
16
10
```

```
for i in range(len(A)):  # and if we don't know how long A is to begin with, we
    print(2*A[i])
```

```
12
8
6
16
10
```

```
B = [] # make an empty list
for x in A:
    B.append(2*x)
print(B)
```

```
[12, 8, 6, 16, 10]
```

```
C = [ 2*x for x in A ]
# This makes exactly the same list B that we had before, but in just one line.
print(C)
```

```
[12, 8, 6, 16, 10]
```

```
def f(x,y):  # this is how we define a function.  Notice that x and y don't hav
    return x + y

print(f(2,3))  # python has one version of + for integers
print(f([1,2,3],[4,5,6]))  # and another version for lists
print(f("hello ", "world"))  # and another version for strings
# what happens if you do f(2, "cat")?
```

```
5
[1, 2, 3, 4, 5, 6]
hello world
```

As a more serious pass, here is a nice tutorial:
https://www.programiz.com/python-programming

For now you just need to be able to understand (and maybe slightly modify) other people's python code.

Let's setup the SQL environment

```
#Install pysqlite3 for python and import pandas to use later
!pip install pysqlite3
from pysqlite3 import dbapi2 as sqlite3
print(sqlite3.sqlite_version)
import pandas as pd
from IPython.display import display, HTML
```

```
Collecting pysqlite3
  Downloading pysqlite3-0.5.4.tar.gz (40 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.7/40.7 kB 1.8 MB/s eta 0:0
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: pysqlite3
  Building wheel for pysqlite3 (setup.py) ... done
  Created wheel for pysqlite3: filename=pysqlite3-0.5.4-cp311-cp311-linux_x
  Stored in directory: /root/.cache/pip/wheels/83/05/4e/8fb9d7378ff72e4fd02
Successfully built pysqlite3
Installing collected packages: pysqlite3
Successfully installed pysqlite3-0.5.4
3.37.2
```

Let's define some helper functions for running queries and printing results

```
dbname = "music_streaming4.db"

def printSqlResults(cursor, tblName):
  try:
    df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor.descript
    display(HTML("<b><font color=Green> " + tblName + "</font></b>" + df.to_htm
  except:
    pass

def runSql(caption, query):
  conn = sqlite3.connect(dbname) # Connect to the database
  cursor = conn.cursor() # Create a cursor (think: it's like a "pointer")
  cursor.execute(query) # Execute the query
  printSqlResults(cursor, caption) # Print the results
  conn.close()

def runStepByStepSql(query, fromline):
  lines = query.strip().split('\n')
  for lineidx in range(fromline, len(lines)):
    partial_query = '\n'.join(lines[:lineidx])
    caption = 'Query till line:' +  partial_query
    runSql(caption, partial_query + ';')
```

Let's setup a Schema and insert some data

```
# Connect to database (creates the file if it doesn't exist)
"""
1. Connections: A connection represents a connection to a database through
which we can execute SQL queries. The dbname here specifies the database.
In SQLlite, if the DB doesn't exist, it will be created.
2. Cursors: A cursor is an object associated with a database connection.
It allows you to execute SQL queries, fetch query results.
"""
conn = sqlite3.connect(dbname)
cursor = conn.cursor()

# Create the Users table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Users (
    user_id INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE
);
""")

# Create the Songs table
```

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS Songs (
    song_id INTEGER PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    artist VARCHAR(100) NOT NULL,
    genre VARCHAR(100)
);
""")

# Create the Listens table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Listens (
    listen_id INTEGER PRIMARY KEY,
    user_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    rating FLOAT,
    listen_time TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id)
);
""")

# Create the recommendations table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Recommendations (
    user_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    recommendation_id not NULL,
    recommendation_time TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id)
);
""")

# Commit changes and close the connection
conn.commit()
conn.close()
```

```
# Connect to database again and insert sample data
conn = sqlite3.connect(dbname)
sqlite3.enable_callback_tracebacks(True)

cursor = conn.cursor()
cursor.execute("delete from Songs;")
cursor.execute("delete from Users;")
cursor.execute("delete from Listens;")
cursor.execute("delete from Recommendations;")
```

```
# Insert sample users
cursor.execute("""
INSERT INTO Users (user_id, name, email)
VALUES
    (1, 'Mickey', 'mickey@example.com'),
    (2, 'Minnie', 'minnie@example.com'),
    (3, 'Daffy', 'daffy@example.com'),
    (4, 'Pluto', 'pluto@example.com');
""")

# Insert sample songs from Taylor Swift, Ed Sheeran, Beatles
cursor.execute("""
INSERT INTO Songs (song_id, title, artist, genre)
VALUES
    (1, 'Evermore', 'Taylor Swift', 'Pop'),
    (2, 'Willow', 'Taylor Swift', 'Pop'),
    (3, 'Shape of You', 'Ed Sheeran', 'Rock'),
    (4, 'Photograph', 'Ed Sheeran', 'Rock'),
    (5, 'Shivers', 'Ed Sheeran', 'Rock'),
    (6, 'Yesterday', 'Beatles', 'Classic'),
    (7, 'Yellow Submarine', 'Beatles', 'Classic'),
    (8, 'Hey Jude', 'Beatles', 'Classic'),
    (9, 'Bad Blood', 'Taylor Swift', 'Rock'),
    (10, 'DJ Mix', 'DJ', NULL);
""")

# Insert sample listens
cursor.execute("""
INSERT INTO Listens (listen_id, user_id, song_id, rating)
VALUES
    (1, 1, 1, 4.5),
    (2, 1, 2, 4.2),
    (3, 1, 6, 3.9),
    (4, 2, 2, 4.7),
    (5, 2, 7, 4.6),
    (6, 2, 8, 3.9),
    (7, 3, 1, 2.9),
    (8, 3, 2, 4.9),
    (9, 3, 6, NULL);
""")
# Commit changes and close the connection
conn.commit()
conn.close()

runSql('Users', "select * from Users;")
runSql('Songs', "select * from Songs;")
runSql('Listens', "select * from Listens;")
```

Users

user_id	name	email
1	Mickey	mickey@example.com
2	Minnie	minnie@example.com
3	Daffy	daffy@example.com
4	Pluto	pluto@example.com

Songs

song_id	title	artist	genre
1	Evermore	Taylor Swift	Pop
2	Willow	Taylor Swift	Pop
3	Shape of You	Ed Sheeran	Rock
4	Photograph	Ed Sheeran	Rock
5	Shivers	Ed Sheeran	Rock
6	Yesterday	Beatles	Classic
7	Yellow Submarine	Beatles	Classic
8	Hey Jude	Beatles	Classic
9	Bad Blood	Taylor Swift	Rock
10	DJ Mix	DJ	None

Listens

listen_id	user_id	song_id	rating	listen_time
1	1	1	4.5	None
2	1	2	4.2	None
3	1	6	3.9	None
4	2	2	4.7	None
5	2	7	4.6	None
6	2	8	3.9	None
7	3	1	2.9	None
8	3	2	4.9	None
9	3	6	NaN	None