Computer Science

Summer Springboard

SUMMER
SPRINGBOARD
Look Inward. Go Upward.

Day 6: Deep Dives

SUMMER SPRINGBOARD
Look Inward. Go Upward.

We interrupt your regularly scheduled lecture...

- To present project 2!
- Each person should take a turn presenting their project.
  - Show a demo of your code and your presentation.
  - After presenting, open the floor to questions / comments.
- Good luck and have fun!

SUMMER SPRINGBOARD
Look Inward. Go Upward.

What’s the goal for today?

- Today, we are going to build upon what we ended on last week
- We will be looking at 2 specific libraries, **pandas** and **numpy** and their features
- Working with these libraries will help you out with your final project

NumPy

Case Study 1: NumPy

- **NumPy** stands for Numerical Python
  - Backbone of scientific computing in Python
  - Gives us the power to work with high-performance arrays and matrices
- Very useful for handling vast amounts of data efficiently
- **NumPy** is also quite versatile
  - Good for complex mathematical problems, creating visualizations, working with AI, etc

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Discussion Time

- Take a few minutes to discuss the following questions with a partner or group:
    - Can you think of any real-world problems where **NumPy** could be useful?
    - Why do we care about efficiency in computing?

SUMMER SPRINGBOARD
Look Inward. Go Upward.

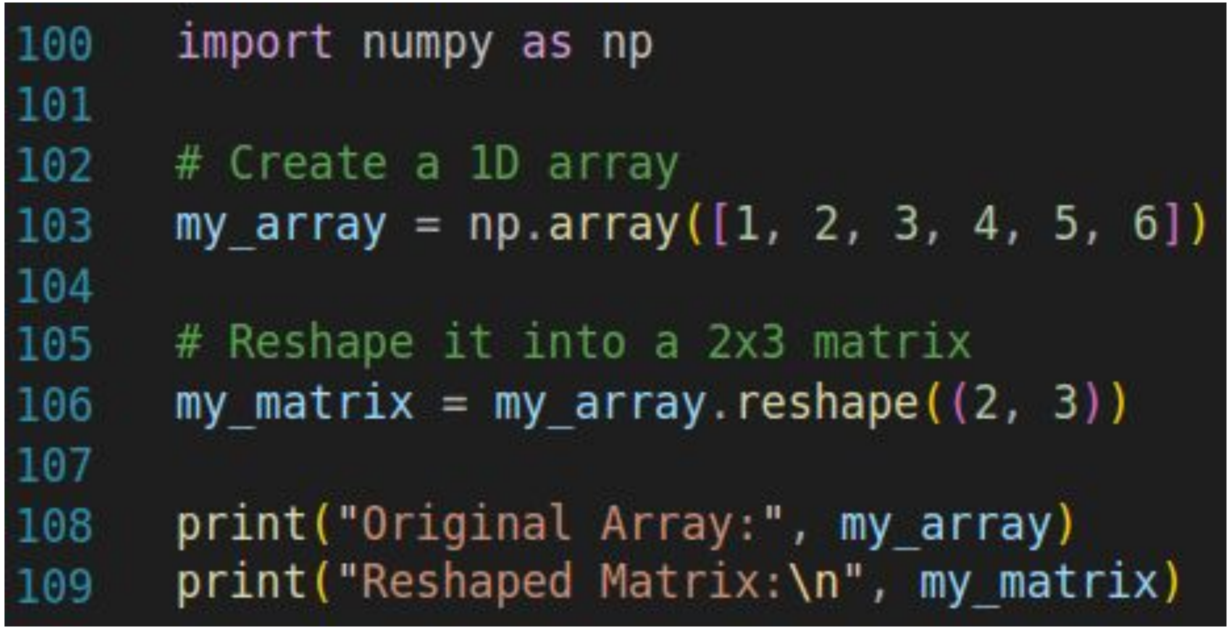
Key Functions in NumPy

- Array creation - the bread and butter of **NumPy**
    - Ex: **np.array([1, 2, 3])** creates an array with 3 elements
    - Ex: **np.arrange(1, 10)** creates an array of numbers from 1 to 9
- Aggregations - find totals, averages, or the biggest number in your data faster than you can say “libraries are super cool!”
    - Ex: **np.sum()** sums all elements in your array, **np.max()** finds the maximum element in an array
- Reshaping - change the shape of your arrays without altering their data
    - **np.reshape()**

SUMMER SPRINGBOARD
Look Inward. Go Upward.

NumPy Examples

- Here is an example of reshaping data; we first create a one-dimensional array of data and then shape it into a 2x3 matrix

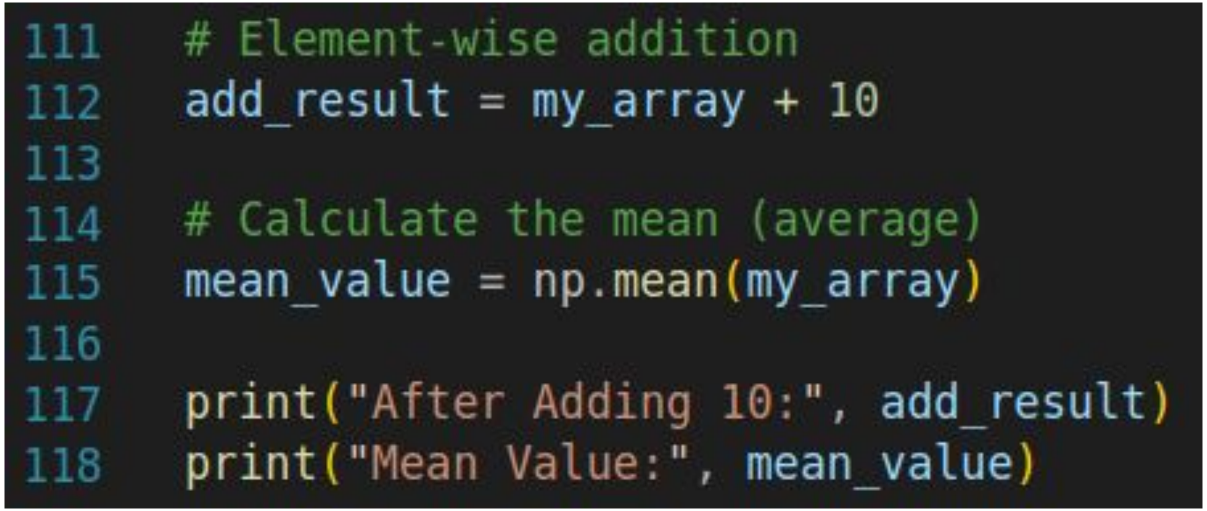


```python
import numpy as np

# Create a 1D array
my_array = np.array([1, 2, 3, 4, 5, 6])

# Reshape it into a 2x3 matrix
my_matrix = my_array.reshape((2, 3))

print("Original Array:", my_array)
print("Reshaped Matrix:\n", my_matrix)
```

SUMMER SPRINGBOARD
Look Inward. Go Upward.

NumPy Examples (Cont'd)

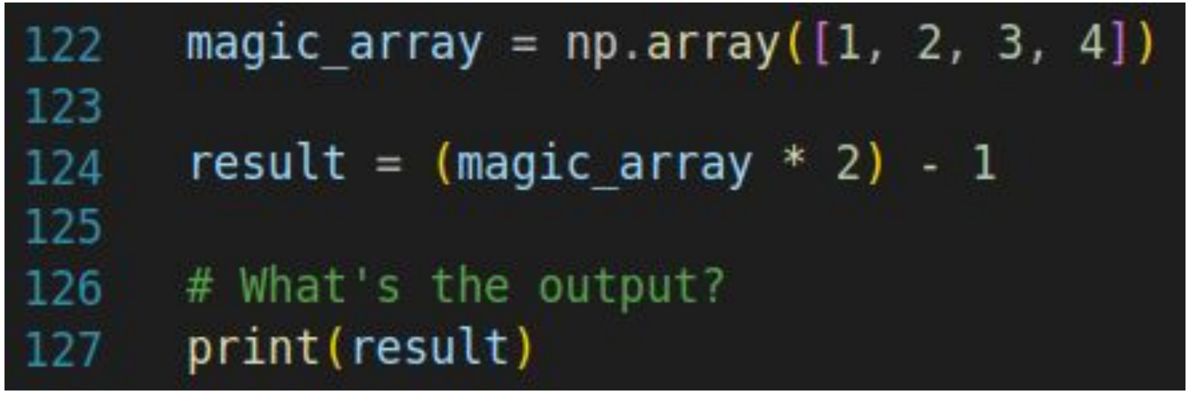
- Below, we see that we first do an operation on our array and then find its mean value. The addition results in the array **[11, 12, 13, 14, 15, 16]** and the mean value of the original array is **3.5**.



```python
# Element-wise addition
add_result = my_array + 10

# Calculate the mean (average)
mean_value = np.mean(my_array)

print("After Adding 10:", add_result)
print("Mean Value:", mean_value)
```

Predict the Output...

- Predict the output of the following code snippets.

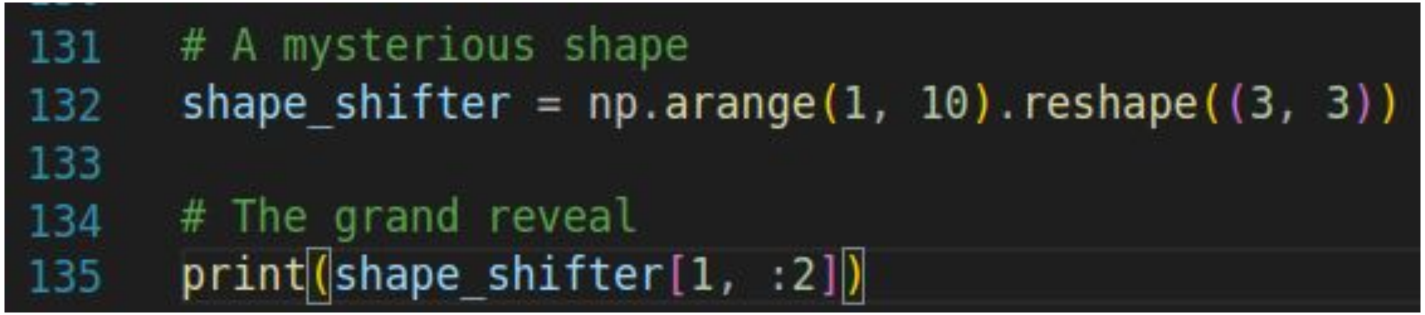


```
magic_array = np.array([1, 2, 3, 4])

result = (magic_array * 2) - 1

# What's the output?
print(result)
```

- Snippet 1: What will be printed when we run this snippet?
  - A) **[2, 4, 6, 8]**
  - B) **[1, 3, 5, 7]**
  - C) **[0, 1, 2, 3]**
  - D) **Error**

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Predict the Output… (Cont'd)



```python
# A mysterious shape
shape_shifter = np.arange(1, 10).reshape((3, 3))

# The grand reveal
print(shape_shifter[1, :2])
```

- Snippet 2: What will be printed when we run this snippet?
  - A) **[4 5 6]**
  - B) **[2 3]**
  - C) **[4 5]**
  - D) **Error**

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Takeaway

- With just a few lines of code, **NumPy** is able to efficiently manipulate and analyze data
- The possibilities are endless!

Practice Problem Time!

- Open up today's Google Colab notebook and work until the first "***PAUSE***"

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Pandas

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Case Study 2: Pandas

- **Pandas** (Panel Data) is a powerful library that makes it easy to explore, manipulate, and analyze data
    - Transforms raw data into something insightful
- Handles data in a way that is both powerful and intuitive
- Key features include:
    - Data manipulation: clean, transform, and merge data
    - Data analysis: perform complex analyses to gain insight
    - Flexibility: works with many forms of data, from tabular data to time series data

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Key Concepts in Pandas

- **Dataframe** - table with rows and columns, where each column can be of a different type
    - Think of this as a spreadsheet
- **Series** - one-dimensional array that can hold any data type
    - Single column from a spreadsheet
- **Index** - both **dataframes** and **series** have an index, which helps locate data

Working with Data in Pandas

- Load data: load your data from your chosen source, like CSV files, into **Pandas** DataFrames
- Exploring data: use commands to peek into your data, understand its structure, and start asking questions
- Cleaning data: deal with missing values, duplicate data, and unwanted entries

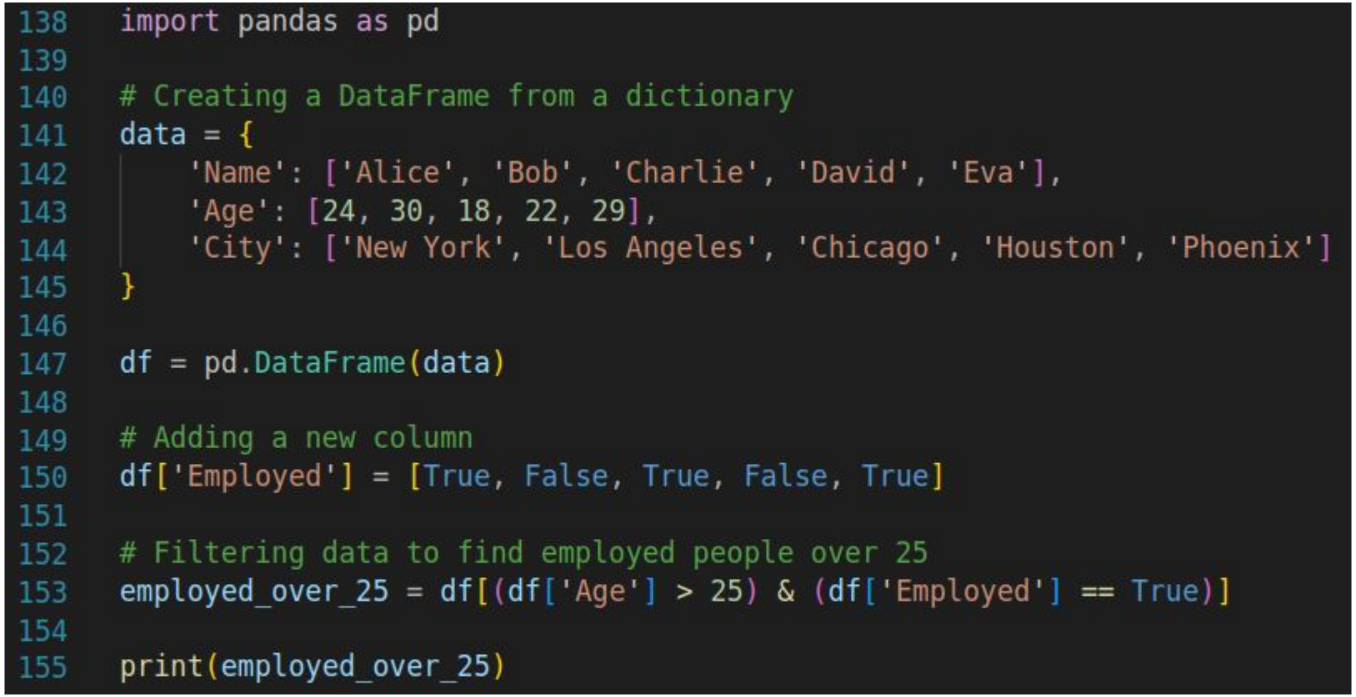
Discussion Time Pt 2

- Take a few minutes to discuss the following questions with a partner or group:
    - Can you think of any real-world problems where **Pandas** could be useful?
    - What do you think are the benefits of storing data in **dataframes**?

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Pandas Examples

- Here is an example snippet of creating and manipulating a **dataframe** in **Pandas**:

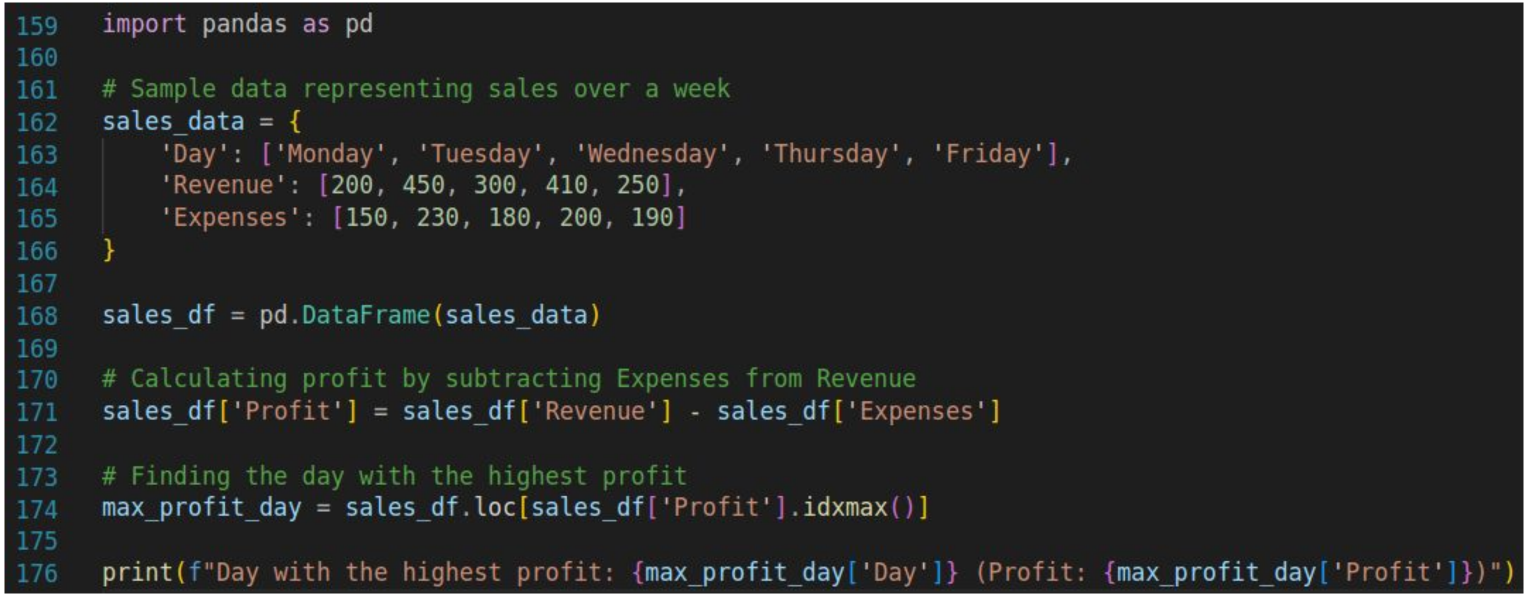


```python
import pandas as pd

# Creating a DataFrame from a dictionary
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [24, 30, 18, 22, 29],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix']
}

df = pd.DataFrame(data)

# Adding a new column
df['Employed'] = [True, False, True, False, True]

# Filtering data to find employed people over 25
employed_over_25 = df[(df['Age'] > 25) & (df['Employed'] == True)]

print(employed_over_25)
```

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Pandas Examples (Cont'd)

- Here is an example snippet of creating analyzing data in **Pandas**:

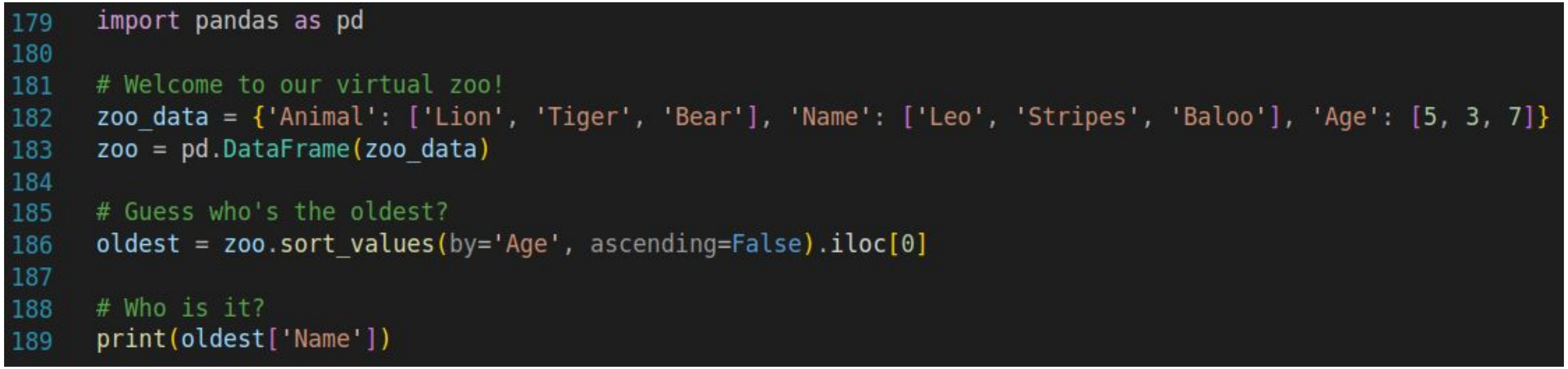


```python
import pandas as pd

# Sample data representing sales over a week
sales_data = {
    'Day': ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'],
    'Revenue': [200, 450, 300, 410, 250],
    'Expenses': [150, 230, 180, 200, 190]
}

sales_df = pd.DataFrame(sales_data)

# Calculating profit by subtracting Expenses from Revenue
sales_df['Profit'] = sales_df['Revenue'] - sales_df['Expenses']

# Finding the day with the highest profit
max_profit_day = sales_df.loc[sales_df['Profit'].idxmax()]

print(f"Day with the highest profit: {max_profit_day['Day']} (Profit: {max_profit_day['Profit']})")
```

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Predict the Output...

- Predict the output of the following code snippets.

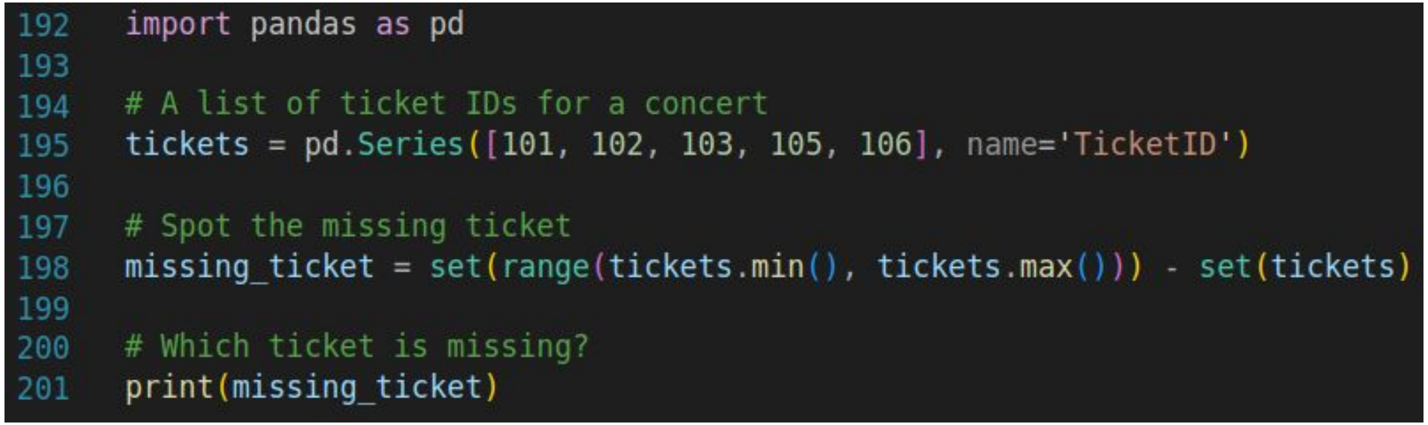


```python
import pandas as pd

# Welcome to our virtual zoo!
zoo_data = {'Animal': ['Lion', 'Tiger', 'Bear'], 'Name': ['Leo', 'Stripes', 'Baloo'], 'Age': [5, 3, 7]}
zoo = pd.DataFrame(zoo_data)

# Guess who's the oldest?
oldest = zoo.sort_values(by='Age', ascending=False).iloc[0]

# Who is it?
print(oldest['Name'])
```

- Snippet 1: What will be printed when we run this snippet?
  - A) **Leo**
  - B) **Stripes**
  - C) **Baloo**
  - D) **Error**

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Predict the Output… (Cont'd)



```python
import pandas as pd

# A list of ticket IDs for a concert
tickets = pd.Series([101, 102, 103, 105, 106], name='TicketID')

# Spot the missing ticket
missing_ticket = set(range(tickets.min(), tickets.max())) - set(tickets)

# Which ticket is missing?
print(missing_ticket)
```

- Snippet 2: What will be printed when we run this snippet?
  - A) **104**
  - B) **107**
  - C) **101**
  - D) **Error**

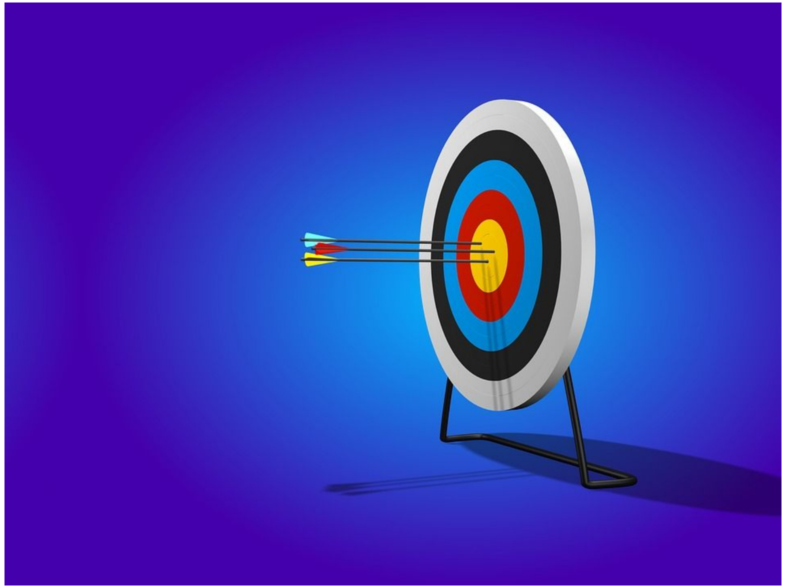
SUMMER SPRINGBOARD
Look Inward. Go Upward.

Takeaway

- **Pandas** is accessible and intuitive
- Despite its simplicity, **Pandas** is incredibly powerful, capable of handling large datasets and complex operations

More Practice!!!

- Work through the rest of the practice problems in today's Google Colab notebook



SUMMER SPRINGBOARD
Look Inward. Go Upward.

Final Project Check-In

SUMMER SPRINGBOARD
Look Inward. Go Upward.

Final Project Check-In

- Final projects will be due soon! As we close out for the day, let’s take some time to discuss the following questions in groups or as a class:
    - What is your idea for your final project?
    - What have you been able to complete so far for your problem?
    - Is there anything that you have been struggling with so far?
- If there is time remaining, use it to make some more progress on your project

SUMMER SPRINGBOARD
Look Inward. Go Upward.