Learn Pandas: Comprehensive Reference Guide

These exercises are based on this YouTube video. ## Setup: Create a Custom Virtual Environment

```
python3 -m venv vEnvPandas      # Create virtual environment named vEnvPandas
source vEnvPandas/bin/activate  # Activate the environment
pip3 install -r requirements.txt  # Install dependencies from file
```

- Requirements file source: GitHub Link ## DataFrames Overview
- **DataFrame**: The primary data structure in pandas, similar to a 2D table.
- Pandas also supports **Series** (1D), but not 3D directly. ## Lambda Functions in Python ### Basic Syntax

```
lambda arguments: expression
```

Example:

```
square = lambda x: x * x
print(square(4))  # Output: 16
```

Lambda with `if-else` (Ternary)

```
check = lambda x: 'Even' if x % 2 == 0 else 'Odd'
print(check(5))  # Output: Odd
```

Lambda with Multiple `if-elif-else`

```
grade = lambda x: 'A' if x >= 90 else 'B' if x >= 80 else 'C' if x >= 70 else 'F'
print(grade(85))  # Output: B
```

Lambda in Pandas

```
df['result'] = df['score'].apply(lambda x: 'Pass' if x >= 60 else 'Fail')
```

Creating DataFrames and Series

```
import pandas as pd
import numpy as np
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake', 'cat', 'dog',
        'dog'],
        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(data, index=labels)
print(df)
s = pd.Series([1, 3, 5, np.nan, 6, 8])
```

- Show version: `pd.show_versions()`
- Data info: `df.info()`, `df.describe()`, `df.head()` ## Accessing and Slicing Data ### Rows and Columns

```
df.iloc[:3]  # First 3 rows
df[['animal', 'age']]  # Specific columns
df.loc[:, ['animal', 'age']]  # Same as above
df.loc[df.index[[3, 4, 8]], ['animal', 'age']]  # Specific rows and columns
```

Conditions

```
df[df['visits'] > 3]
df[df['age'].isnull()]  # or df[df['age'].isna()]
df[(df['animal']=='cat') & (df['age']<3)]
df[df['age'].between(2, 4)]
df['animal'] = df['animal'].replace('snake', 'python')
```

Change Values

```
df.loc['f', 'age'] = 1.5
df['priority2'] = df['priority'] == 'yes'
```

Add/Remove Rows and Columns

```
df.loc['k'] = [5.5, 'dog', 'no', 2]  # Add
df.drop('k', inplace=True)  # Delete row
df.drop(['priority'], axis=1, inplace=True)  # Delete column
```

Sorting and Renaming

```
df.sort_values(['age','visits'], ascending=[False, True])
df.rename({'priority2':'priority'}, axis=1)
```

Pivot Tables

```
dfNew = df.pivot_table(index='animal', columns='visits', values='age', aggfunc='mean')
```

Additional DataFrame Operations

Create Array DataFrame

```
df_a = pd.DataFrame([[1,2,3],[3,4,5],[7,8,9]]*4, columns=["A","B","C"])
```

Summarizing

```
df.head(), df.tail(), df.index, df.columns, df.shape
df.info(), df.describe(), df_a.nunique()
```

Reading Data

```
pd.read_parquet('path.parquet')
pd.read_excel('path.xlsx')
pd.read_csv('path.csv')
```

Random Samples

```
coffee.sample(10)
```

Summary of `.loc` vs `.iloc`

Feature	`.loc`	`.iloc`
Index Type	Label-based	Integer-based

Includes	Stop label **inclusive**	Stop index **exclusive**
Usage	`df.loc[row_label, col_label]`	`df.iloc[row_pos, col_pos]`

- Get cell: `df.at[1, 'Day']` or `df.iat[1, 1]`
- Assign value: `coffee.loc[1:4, 'Units Sold'] = 100` ## For Loops

```
for index, row in coffee.iterrows():
    print(index, row['Units Sold'])
```

Filtering Data

```
bios.loc[bios['height_cm'] > 215, ['name', 'born_country', 'height_cm']]
bios[(bios['height_cm'] > 210) & (bios['born_country'] == 'USA')]
bios[bios['name'].str.contains("Keith|Patric", case=False)]
bios[bios['born_country'].isin(["USA","FRA","GBR"]) & bios['name'].str.startswith("La")]
bios.query("born_country == 'USA' and born_city == 'Seattle'")
```

Modify Columns

Using `np.where`

```
coffee['New_Price'] = np.where(coffee['Coffee Type'] == 'Espresso', 10, 20)
```

Split and Apply

```
bios_new["first_name"] = bios_new['name'].str.split(' ').str[0]
bios_new['height_category'] = bios['height_cm'].apply(lambda x: 'Short' if x <165 else
        'Medium' if x < 185 else 'Tall')
```

Apply Function Across Rows

```
def categorize_athlete(row):
    if row['height_cm'] < 175 and row['weight_kg'] < 70:
        return 'Light Weight'
    elif row['height_cm'] < 185 and row['weight_kg'] <= 80:
        return 'Middle Weight'
    else:
        return 'Heavy Weight'
bios_new['Category'] = bios_new.apply(categorize_athlete, axis=1)
```

- Constant value: `coffeeX['LK'] = 5`
- Deleting: `coffee.drop(0)`, `coffee.drop(columns='Price', inplace=True)`
- Rename: `coffeeY.rename(columns={'LK':'Price'}, inplace=True)`
- Copy: `coffeeY = coffeeX.copy()`
- Convert date: `bios_new["born_date"] = pd.to_datetime(bios_new['born_date'], errors='coerce')` ## Merge and Concat

```
pd.merge(bios, nocs, left_on='born_country', right_on='NOC', how='left')
pd.concat([usa, gbr], axis=0)
```

Handling Missing Data

```
coffee.fillna(100, inplace=True)
coffee['Units Sold'] = coffee['Units Sold'].interpolate()
coffee.dropna(inplace=True)
coffee[coffee['Units Sold'].isna()]
```

Grouping

```
coffee.groupby(['Coffee Type']).agg({'Units Sold': 'sum', 'price': 'sum', 'revenue':
        'sum'})
coffee.groupby(['Coffee Type','Day']).agg({'Units Sold': 'sum', 'price': 'sum', 'revenue':
        'sum'})
```

Pivot and Count

```
coffee.pivot(columns='Coffee Type', index='Day', values='revenue')
pd.pivot_table(df, values='Temperature', index='City', columns='Date')
bios.groupby([bios['birth_year'], bios['birth_month']])
        ['name'].count().reset_index().sort_values('name', ascending=False).head(30)
```

Ranking, Cumulative Sum

```
bios['height_rank'] = bios['height_cm'].rank()
coffee['cumulative_revenue'] = coffee['revenue'].cumsum()
coffee['rolling_sum'] = coffee['revenue'].rolling(3).sum()
```

Plotting

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.hist(bios['height_cm'].dropna(), bins=30, log=True, edgecolor='black')
plt.title('Histogram of Heights')
plt.xlabel('Height (cm)')
plt.ylabel('Frequency (log scale)')
plt.grid(True)
plt.show()
```

Advanced Examples

```
df['animal'].value_counts()
df.idxmax(axis=1)  # Label of max
df.values.argmax(axis=1)  # Index of max
df.nlargest(3, 'age')
df.shift()
df.to_numpy()
df.agg(lambda x: np.mean(x) * 5.6)
df.transform(lambda x: x * 101.2)
s.str.lower()
```

Time Series

```
rng = pd.date_range("1/1/2012", periods=100, freq="s")
ts = pd.Series(np.random.randint(0, 500, len(rng)), index=rng)
ts.resample("5Min").sum()
```