Cheat Sheet for comprehensive DataCamp Data Scientist with Python Career Track

Python Basics

Variables and Data Types

- **Variables**: `x = 10`, `name = "Alice"`

- **Data Types**:

- `int`: `x = 10`
- `float`: `y = 10.5`
- `str`: `name = "Alice"`
- `bool`: `is_valid = True`
- `list`: `my_list = [1, 2, 3]`
- `tuple`: `my_tuple = (1, 2, 3)`
- `dict`: `my_dict = {"name": "Alice", "age": 30}`
- `set`: `my_set = {1, 2, 3}`

Basic Operations

- **Arithmetic**: `+`, `-`, `*`, `/`, `//` (floor division), `%` (modulus), `**` (exponentiation)

- **Comparison**: `==`, `!=`, `>`, `<`, `>=`, `<=`

- **Logical**: `and`, `or`, `not`

Control Flow

- **If Statements**:

```
if condition:
    # code
elif another_condition:
    # code
else:
    # code
```

- **For Loops**:

```
for i in range(5):
    # code
```

- **While Loops**:

```
while condition:
    # code
```

Data Manipulation with Pandas

DataFrame Basics

- **Create DataFrame**:

```
import pandas as pd
df = pd.DataFrame({
    'column1': [1, 2, 3],
    'column2': ['A', 'B', 'C']
})
```

- **Read/Write Files**:

- `pd.read_csv('file.csv')`
- `df.to_csv('file.csv', index=False)`

DataFrame Operations

- **Select Columns**:

- `df['column1']`
- `df[['column1', 'column2']]`

- **Filter Rows**:

- `df[df['column1'] > 1]`

- **Add/Remove Columns**:

- `df['new_column'] = df['column1'] + df['column2']`
- `df.drop(columns=['column1'], inplace=True)`

- **GroupBy**:

```
df.groupby('column1').mean()
```

- **Merge/Join**:

```
pd.merge(df1, df2, on='key')
```

Data Visualization with Matplotlib and Seaborn

Matplotlib Basics

- **Plot Types**:

- `plt.plot(x, y)`
- `plt.scatter(x, y)`
- `plt.bar(x, height)`
- `plt.hist(data)`

- **Customization**:

- `plt.title('Title')`
- `plt.xlabel('X Label')`
- `plt.ylabel('Y Label')`
- `plt.legend()`

- **Show Plot**:

- `plt.show()`

Seaborn Basics

- **Plot Types**:

- `sns.lineplot(x='x', y='y', data=df)`
- `sns.scatterplot(x='x', y='y', data=df)`
- `sns.barplot(x='x', y='y', data=df)`
- `sns.histplot(data=df, x='column')`

- **Customization**:

- `sns.set_style('darkgrid')`
- `sns.set(rc={'figure.figsize':(10, 5)})`

Machine Learning with Scikit-Learn

Model Training and Evaluation

- **Import Libraries**:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

- **Split Data**:

```
  X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
```

- **Train Model**:

```
  model = LinearRegression()
  model.fit(X_train, y_train)
```

- **Predict**:

```
  y_pred = model.predict(X_test)
```

- **Evaluate**:

```
  mse = mean_squared_error(y_test, y_pred)
```

Model Selection and Tuning

- **Cross-Validation**:

```
  from sklearn.model_selection import cross_val_score
  scores = cross_val_score(model, X, y, cv=5)
```

- **Grid Search**:

```
  from sklearn.model_selection import GridSearchCV
  param_grid = {'C': [0.1, 1, 10]}
  grid_search = GridSearchCV(model, param_grid, cv=5)
  grid_search.fit(X, y)
```

Advanced Topics

Handling Missing Data

- **Drop Missing Values**:

```
  df.dropna(inplace=True)
```

- **Fill Missing Values**:

```
df.fillna(method='ffill', inplace=True)
```

Feature Engineering

- **Create Features**:

```
df['new_feature'] = df['column1'] * df['column2']
```

- **One-Hot Encoding**:

```
df = pd.get_dummies(df, columns=['category_column'])
```

Time Series Analysis

- **Resample**:

```
df.resample('D').mean()
```

- **Shift**:

```
df['shifted'] = df['column'].shift(1)
```

Tips and Tricks

Efficient Code

- **List Comprehensions**:

```
squares = [x**2 for x in range(10)]
```

- **Lambda Functions**:

```
add = lambda x, y: x + y
```

Debugging

- **Print Statements**:

```
print(variable)
```

- **Try-Except Blocks**:

```
try:
    # code
except Exception as e:
    print(f"Error: {e}")
```

Performance Optimization

- **Vectorization**:

```
df['new_column'] = df['column1'] + df['column2']
```

- **Caching**:

```
from functools import lru_cache
@lru_cache(maxsize=None)
def expensive_function():
    # code
```

Useful Libraries

Data Manipulation

- **Pandas**: `import pandas as pd`
- **NumPy**: `import numpy as np`

Data Visualization

- **Matplotlib**: `import matplotlib.pyplot as plt`
- **Seaborn**: `import seaborn as sns`

Machine Learning

- **Scikit-Learn**: `import sklearn`
- **XGBoost**: `import xgboost as xgb`
- **TensorFlow**: `import tensorflow as tf`

Conclusion

- **Practice Regularly**: Hands-on practice is key to mastering data science.

- **Explore Documentation**: Refer to official documentation for detailed explanations and examples.

- **Join Communities**: Engage with online communities like Stack Overflow and GitHub for support and collaboration.

By Ahmed Baheeg Khorshid

ver 1.0