Python for Data Science Cheat Sheet! 🎉

Unlock your data superpowers with this quick & colorful guide to Python essentials! 🚀

1. Core Python Fundamentals ✨

Python is the magical wand for data science! Here are the spells (basics) you'll need:

Data Types: Your Info Ingredients 🍲

- **Numbers:** `int` (whole), `float` (decimals). E.g., `age = 30`, `price = 9.99`
- **Strings:** `str` (text). E.g., `city = "DataTown"`
- **Booleans:** `bool` (**True**/**False**). E.g., `is_awesome = True`
- **Lists:** Ordered, mutable. Like a dynamic shopping list! E.g., `fruits = ['apple', 'banana']`
- **Tuples:** Ordered, immutable. A fixed recipe! E.g., `coords = (10, 20)`
- **Dictionaries:** Key-value pairs. Your data's personal directory! E.g., `person = {'name': 'Alice', 'age': 25}`
- **Sets:** Unordered, unique items. No duplicates allowed! E.g., `unique_ids = {1, 5, 2}`

Control Flow: Guiding Your Code's Journey 🗺️

- **Conditionals:** `if`, `elif`, `else` to make decisions.
- **Loops:** `for item in iterable:` (for iterating), `while condition:` (for repeating).
- **List Comprehensions:** A super-speedy way to build lists! `[x*2 for x in range(5) if x > 0]`

Functions: Your Reusable Superpowers ⚡

- Define with `def my_power(arg):`.
- Send results back with **return value**.

---

2. NumPy (Numerical Python) 🧠

Think of NumPy as Python's muscle for numbers, especially when dealing with big arrays!

```
import numpy as np
```

Array Creation: Building Your Data Towers 🏗️

- `np.array([1,2,3])`: Craft an array from a list.
- `np.zeros((2,3))`, `np.ones((2,2))`: Make arrays full of zeros or ones.
- `np.arange(0,10,2)`: Create a sequence with a step.
- `np.random.rand(2,2)`: Random numbers for simulations!

Attributes & Operations: Knowing Your Array's Secrets 🕵️

- `arr.shape`: What shape is your data? `(rows, cols)`
- `arr + 5`, `arr * arr`: Super-fast math across the entire array!
- `np.dot(arr1, arr2)`: The fancy matrix multiplication.
- `arr.sum()`, `arr.mean(axis=0)`: Quick stats (total, average).
- `arr[0]`, `arr[0,1]`: Pinpoint specific data points.
- `arr[arr > 3]`: Filter data based on conditions.
- `arr.reshape(new_shape)`: Change your array's shape (carefully!).

---

3. Pandas (Data Analysis Library) 🐼

Pandas is your friendly spreadsheet-in-Python, perfect for cleaning, transforming, and analyzing tabular data!

```
import pandas as pd
```

Data Structures: Your Data Containers 📦

- `pd.Series(list)`: A single column of data with labels.
- `pd.DataFrame(dict)`: The main event! A 2D table (like Excel or a SQL table).

I/O & Inspection: Peek at Your Data! 👀

- `pd.read_csv('file.csv')`: Load data from a CSV.
- `df.head()`, `df.tail()`: Glimpse the top/bottom rows.
- `df.info()`, `df.describe()`: Get quick summaries and stats.
- `df.shape`, `df.columns`: See dimensions and column names.

Selection & Manipulation: Data Gymnastics! 🤸

- `df['col']`, `df[['col1', 'col2']]`: Grab specific columns.
- `df.iloc[0]`, `df.loc[0, 'col1']`: Precisely select rows & columns.
- `df[df['col'] > val]`: Filter rows like a pro!
- `df.isnull().sum()`, `df.dropna()`, `df.fillna(value)`: Tidy up messy (missing) data.
- `df['new_col'] = ...`: Add sparkling new columns.
- `df.rename(columns={'old':'new'})`: Give columns new, clearer names.
- `df.groupby('col')['agg_col'].mean()`: Group data for powerful insights!
- `pd.merge(df1, df2, on='key')`: Join datasets like puzzle pieces.

---

4. Matplotlib & Seaborn 🎨

Turn raw numbers into dazzling stories with these visualization wizards!

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Matplotlib Basics: Your Art Supplies 🖌️

- `plt.plot(x, y)`, `plt.scatter(x, y)`: **Line graphs** for trends, **scatter** for relationships.
- `plt.bar(cat, val)`, `plt.hist(data)`: **Bar charts** for comparisons, **histograms** for distributions.
- `plt.title('Awesome Plot')`, `plt.xlabel('Time')`: Label everything clearly!
- `plt.show()`: Unveil your masterpiece!

Seaborn for Stats Plots: Instant Eye-Candy! 💖

- `sns.set_style('whitegrid')`: Make your plots look sleek instantly.
- `sns.regplot(x, y, data)`: **Scatter plot** with a handy **regression line**.
- `sns.histplot(data, kde=True)`: Beautiful **histograms**, often with a smooth **density curve**.
- `sns.boxplot(x, y, data)`: Show **data distributions** and spot **outliers** across categories.
- `sns.heatmap(corr_matrix, annot=True)`: Visualize **correlations** with a vibrant **color map**!
- `sns.pairplot(df, hue='cat_col')`: Explore **relationships** between *all* pairs of variables.

---

5. Scikit-learn (Machine Learning) 🤖

This is where your data comes alive with predictive power! Scikit-learn is your go-to for building smart models.

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

The ML Magic Formula (Workflow):

1. **Load Data:** Get your **features** (`X`) and what you want to predict (`y`).
2. **Split Data:** Divide into **train** and **test** sets so your model isn't cheating! (`train_test_split`).
3. **Preprocess:** Clean up and transform your **features** (e.g., `StandardScaler` to normalize).
4. **Choose Model:** Pick your algorithm (e.g., **LinearRegression** for numbers, **LogisticRegression** for categories).
5. **Train Model:** `model.fit(X_train, y_train)` — Teach the model to learn patterns.
6. **Predict:** `model.predict(X_test)` — Let the model guess outcomes on new data.
7. **Evaluate:** Check how good your model is! (**mean_squared_error** for regression, **accuracy_score** for classification).

---

6. Other Useful Libraries 🌟

The Python universe has even more awesome tools for your data adventures:

- **SciPy:** For all sorts of scientific heavy-lifting (**optimization**, **signal processing**).
- **Statsmodels:** Dive deep into **statistical modeling** and **tests**.
- **Plotly/Bokeh:** Craft **interactive**, **web-ready visualizations** that shimmer!
- **TensorFlow/PyTorch:** For the ultimate challenge: **Deep Learning**! 🧠
- **NLTK/spaCy:** Tame text data and uncover linguistic secrets (**NLP**).
- **Requests/Beautiful Soup:** Your toolkit for **web-scraping** adventures! 🕸️

Full Python Code Reference 💡

Here are all the code snippets, gathered in one place for quick copying and reference. Happy coding! 🧑‍💻👨‍💻

Core Python

```
# Data Types
x = 10; y = 3.14
name = "Alice"; message = 'Hello, World!'
is_valid = True; is_empty = False
my_list = [1, 'a', 3]; my_list.append(4); my_list[0] = 0
my_tuple = (1, 'a'); first = my_tuple[0]
my_dict = {'key': 'value'}; print(my_dict['key'])
my_set = {1, 2, 3}; my_set.add(4)

# Control Flow
if x > 0: print("Positive")
elif x == 0: print("Zero")
else: print("Negative")

for i in range(3): print(i)
count = 0; while count < 2: print(count); count += 1

squares = [x**2 for x in range(3)] # [0, 1, 4]

# Functions
def greet(name): return f"Hello, {name}!"
print(greet("Charlie"))
```

NumPy

```
import numpy as np

# Array Creation
arr = np.array([1, 2, 3])
zeros = np.zeros((2, 3))
ones = np.ones((2, 2))
arange_arr = np.arange(0, 5, 1) # [0, 1, 2, 3, 4]
linspace_arr = np.linspace(0, 1, 3) # [0.0, 0.5, 1.0]
rand_arr = np.random.rand(2, 2)
randint_arr = np.random.randint(0, 10, size=(2, 2))

# Attributes
print(arr.shape)
print(arr.dtype)
print(arr.ndim)

# Operations & Indexing
a = np.array([1, 2, 3]); b = np.array([4, 5, 6])
print(a + b)
print(np.dot(a, b))
matrix = np.array([[1, 2], [3, 4]])
print(matrix.sum())
print(matrix.mean(axis=0))
print(arr[0])
print(matrix[0, 1])
print(arr[1:3])
print(matrix[:, 0])
print(arr[arr > 2])
arr_2d = arr.reshape(1, 3)
print(matrix.T)
```

Pandas

```
import pandas as pd
import numpy as np # For np.nan

# Data Structures
s = pd.Series([1, 2, 3])
data = {'col1': [1, 2], 'col2': ['A', 'B']}
df = pd.DataFrame(data)

# I/O & Inspection
df_csv = pd.read_csv('data.csv')
df.to_csv('output.csv', index=False)
print(df.head())
df.info()
print(df.describe())
print(df.shape)
print(df.columns)
print(df.dtypes)

# Selection & Filtering
print(df['col1'])
print(df[['col1', 'col2']])
print(df.iloc[0])
print(df.loc[0])
print(df[df['col1'] > 1])

# Data Cleaning & Manipulation
df_with_nan = pd.DataFrame({'A': [1, np.nan], 'B': [3, 4]})
print(df_with_nan.isnull().sum())
df_cleaned = df_with_nan.dropna()
df_filled = df_with_nan.fillna(0)
df['new_col'] = df['col1'] * 2
df_dropped = df.drop('col2', axis=1)
df_renamed = df.rename(columns={'col1': 'feature_A'})
df['col1_sq'] = df['col1'].apply(lambda x: x**2)
grouped_df = df.groupby('col2')['col1'].mean()

df1 = pd.DataFrame({'key': ['A', 'B'], 'val1': [1, 2]})
df2 = pd.DataFrame({'key': ['B', 'C'], 'val2': [3, 4]})
merged_df = pd.merge(df1, df2, on='key', how='inner')
concatenated_df = pd.concat([df1, df2], axis=0)
```

Matplotlib & Seaborn

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np # For example data

# Matplotlib Basics
x_data = [1, 2, 3]; y_data = [10, 15, 7]
plt.plot(x_data, y_data, marker='o', color='purple')
plt.title("Line Plot Example")
plt.xlabel("X Value"); plt.ylabel("Y Value")
plt.show() # Display the plot

plt.scatter(x_data, y_data, color='teal')
plt.show()

categories = ['A', 'B', 'C']; values = [5, 8, 3]
plt.bar(categories, values, color=['#8b5cf6', '#a78bfa', '#c4b5fd'])
plt.show()

data_hist = np.random.randn(100)
plt.hist(data_hist, bins=10, edgecolor='black', color='#facc15')
plt.show()

# Seaborn for Stats Plots
sns.set_style("whitegrid")
# To run these, you'd typically need to load a dataset, e.g.:
# tips = sns.load_dataset("tips")

# sns.regplot(x="total_bill", y="tip", data=tips, color='#ec4899')
# plt.show()

# sns.histplot(tips['total_bill'], kde=True, color='#22d3ee')
# plt.show()

# sns.boxplot(x="day", y="total_bill", data=tips, palette='light:purple')
# plt.show()

# # Create a dummy correlation matrix for heatmap example
# dummy_corr = pd.DataFrame(np.random.rand(3,3), columns=['A','B','C'], index=['A','B','C'])
# dummy_corr.iloc[0,1] = 0.8; dummy_corr.iloc[1,0] = 0.8
# dummy_corr.iloc[2,0] = -0.5; dummy_corr.iloc[0,2] = -0.5
# sns.heatmap(dummy_corr, annot=True, cmap='coolwarm', fmt=".2f")
# plt.title("Dummy Correlation Heatmap")
# plt.show()
```

Scikit-learn

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, accuracy_score
from sklearn.preprocessing import StandardScaler
import numpy as np # For example data

# Example Data
X = np.array([[1], [2], [3], [4], [5]]) # Features
y = np.array([2, 4, 5, 4, 6])          # Target (Regression)
y_class = np.array([0, 0, 1, 1, 0])    # Target (Classification)

# 1. Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# 2. Preprocess (Optional - Scaling)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 3. Choose & Train Model (Regression Example)
reg_model = LinearRegression()
reg_model.fit(X_train_scaled, y_train)

# 4. Predict & Evaluate (Regression)
y_pred_reg = reg_model.predict(X_test_scaled)
rmse = np.sqrt(mean_squared_error(y_test, y_pred_reg))
print(f"Regression RMSE: {rmse:.2f}")

# # Classification Example:
# X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X, y_class, test_size=0.4, random_state=42)
# scaler_c = StandardScaler()
# X_train_c_scaled = scaler_c.fit_transform(X_train_c)
# X_test_c_scaled = scaler_c.transform(X_test_c)
# class_model = LogisticRegression(random_state=42)
# class_model.fit(X_train_c_scaled, y_train_c)
# y_pred_class = class_model.predict(X_test_c_scaled)
# accuracy = accuracy_score(y_test_c, y_pred_class)
# print(f"Classification Accuracy: {accuracy:.2f}")

# Common Models:
# from sklearn.cluster import KMeans
# from sklearn.ensemble import RandomForestClassifier
# from sklearn.svm import SVC
```

Created with ❤️ by @preettheman