Python One Line Codes – Code n Use

Pandas library in Python is super powerful because you can do so much in one line. Pandas often lets you replace multiple lines of loops and conditions with **one-liner vectorized operations** → faster and cleaner

🚀 Pandas One-Liners by Category

📁 Creating and Importing Data from External Tools

From CSV

```
df = pd.read_csv("file.csv")
```

From Excel

```
df = pd.read_excel("file.xlsx", sheet_name="Sheet1")
```

From SQL

```
df = pd.read_sql("SELECT * FROM table", con=connection)
```

From dictionary

```
df = pd.DataFrame({"name": ["Alice", "Bob"], "age": [25, 30]})
```

🧹 Handling Missing Values

```
df.dropna()                          # Drop all rows with NaN

df.fillna(0)                         # Replace NaN with 0

df["col"].fillna(df["col"].mean())   # Fill NaN with column mean

df.dropna(axis=1)                    # Drop columns with NaN

df.isna().sum()                      # Count NaN per column
```

🔍 Filtering, Sorting, and Working with Data Elements

```
df[df["age"] > 30]                   # Filter rows

df.query("age > 30 and city == 'NY'") # SQL-like filtering

df.sort_values("salary", ascending=False)  # Sort by column
```

```
df["col"].unique()                    # Unique values

df["col"].value_counts()              # Frequency count

df["col"].apply(lambda x: x**2)       # Apply function
```

📊 Creating Cross-Tabs and Pivot Tables

```
pd.crosstab(df["gender"], df["purchased"])      # Cross-tab

df.pivot_table(values="sales", index="region",
               columns="year", aggfunc="sum")   # Pivot table

df.groupby("region")["sales"].sum()             # Group by
```

🔗 Working with Multiple DataFrames (Append, Merge)

```
pd.concat([df1, df2])                  # Append

df1.merge(df2, on="id", how="inner")               # SQL-like join

df1.join(df2.set_index("id"), on="id")             # Join on index
```

🧠 Logical Operations and Control Flow

```
df["new"] = df["age"].apply(lambda x: "Adult" if x >= 18 else "Child")

df["is_high_salary"] = df["salary"] > 50000
```

🔁 For Loops and Iteration

(Avoided in Pandas, but still possible)

```
for idx, row in df.iterrows():
  print(row["name"], row["age"])
```

⏰ Working with Dates and Time

```
df["date"] = pd.to_datetime(df["date"])            # Convert to datetime

df["year"] = df["date"].dt.year                    # Extract year
```

```
df["month"] = df["date"].dt.month                    # Extract month

df["weekday"] = df["date"].dt.day_name()             # Extract weekday

df[(df["date"] >= "2024-01-01") & (df["date"] <= "2024-12-31")]  # Filter by range
```

🚀 Pandas One-Liners: Advanced Tasks

📈 Creating Charts (via Pandas + Matplotlib)

```
df["sales"].plot(kind="line")                # Line chart of sales

df["sales"].plot(kind="hist", bins=20)       # Histogram

df.plot(x="month", y="sales", kind="bar")    # Bar chart

df.plot.scatter(x="age", y="income")         # Scatter plot

df["category"].value_counts().plot.pie()     # Pie chart
```

📊 Statistical Analysis

```
df.describe()                                # Summary statistics

df["col"].mean()                             # Mean

df["col"].median()                           # Median

df["col"].mode()                             # Mode

df["col"].var()                              # Variance

df["col"].std()                              # Standard deviation

df.corr()                                    # Correlation matrix

df.cov()                                     # Covariance matrix
```

Python One Line Codes – Code n Use

🤖 Model Building (with scikit-learn)

```
from sklearn.linear_model import LinearRegression

X = df[["feature1", "feature2"]]          # Features
y = df["target"]                          # Target variable

model = LinearRegression().fit(X, y)      # Fit model
y_pred = model.predict(X)                 # Predict
df["predictions"] = y_pred                # Save predictions in df
```

🛢️ Running SQL on Pandas DataFrames

```
import pandasql as ps

q = "SELECT name, age FROM df WHERE age > 30"
result = ps.sqldf(q, locals())            # Run SQL query on DataFrame
```

🗓️ Time Series Analysis

```
df.set_index("date")["sales"].resample("M").sum()   # Monthly sales

df["sales"].rolling(7).mean()                       # 7-day moving average

df["sales"].expanding().mean()                      # Expanding mean
```

🔍 Data Transformation Tricks

```
df["col"].map(str.upper)                    # Apply string method

df.rename(columns={"old":"new"}, inplace=True)   # Rename columns

df.drop_duplicates()                        # Drop duplicate rows

df.sort_values(["col1","col2"])             # Sort by multiple cols
```