NINJA DATA

NUMPY 2, MATPLOTLIB 3, SEABORN 0.13

Python Cheat Sheet Mistrz

Poza pandas: NumPy do liczenia, wykresy w matplotlib i seaborn, statystyka w SciPy.

ŚCIĄGA | NOWICJUSZ | WOJOWNIK | MISTRZ

01 NUMPY
Tablice zamiast pętli

- **import, array**: tablica liczb jednego typu, liczy całą naraz.

```
import numpy as np
ceny = np.array([120, 45, 300, 89])
ceny * 1.23        # brutto dla każdej
```

- **arange, reshape**: ciąg liczb w nowym kształcie (.shape pokaże wymiary).

```
a = np.arange(12)  # 0, 1, ..., 11
m = a.reshape(3, 4)
```

- **arange, linspace**: gotowe ciągi liczb.

```
np.arange(0, 10, 2)    # [0 2 4 6 8]
np.linspace(0, 1, 5)   # 5 punktów od 0 do 1
```

- **Agregacje**: axis=0 po kolumnach, axis=1 po wierszach.

```
m.sum(axis=0)      # suma każdej kolumny
m.mean(axis=1)     # średnia każdego wiersza
```

- **median, percentile, std**: rozkład w trzech liczbach.

```
np.median(ceny)
np.percentile(ceny, 90)
ceny.std()
```

02 WARUNKI
Maski, where, select

- **Maska**: warunek daje tablicę True i False, wybiera wiersze.

```
ceny[ceny > 100]       # [120 300]
(ceny > 100).sum()     # ile spełnia
```

- **np.where**: if na całej tablicy.

```
np.where(ceny >= 100, "drogie", "tanie")
```

- **np.select**: kilka progów, pierwszy spełniony wygrywa.

```
np.select([ceny >= 200, ceny >= 100],
          ["Premium", "Średnia"],
          default="Budżet")
```

- **NaN**: braki w liczbach. Zwykłe funkcje dają nan, wersje nan* je pomijają.

```
x = np.array([1.0, np.nan, 3.0])
np.isnan(x).sum()      # 1
np.nanmean(x)          # 2.0
```

03 LICZENIE
Wektorowo i losowo

- **Broadcasting**: działanie liczby z tablicą albo wiersza z macierzą.

```
z = (ceny - ceny.mean()) / ceny.std()
```

- **cumsum, diff, round**: narastająco, różnice, zaokrąglenie.

```
np.cumsum([10, 20, 30])  # [10 30 60]
np.diff([10, 20, 35])    # [10 15]
np.round(z, 2)
```

- **default_rng**: losowe liczby z ziarnem, wynik powtarzalny.

```
rng = np.random.default_rng(42)
rng.normal(100, 15, size=1000)
rng.integers(1, 7, size=10)  # kostka
```

- **corrcoef**: korelacja dwóch serii (od -1 do 1).

```
np.corrcoef(reklama, sprzedaz)[0, 1]
```

WSKAZÓWKA Pętla `for` po milionie wierszy liczy sekundy, NumPy ułamek sekundy. Gdy piszesz pętlę po liczbach, sprawdź, czy nie ma gotowej funkcji.

04 STATYSTYKA
SciPy

- **ttest_ind**: czy średnie dwóch grup się różnią (test A/B).

```
from scipy import stats
wynik = stats.ttest_ind(grupa_a, grupa_b,
                        equal_var=False)
wynik.pvalue       # < 0.05: różnica istotna
```

- **pearsonr, spearmanr**: korelacja z p-value.

```
r, p = stats.pearsonr(reklama, sprzedaz)
```

- **zscore**: ile odchyleń od średniej, szybki test odstających.

```
odst = np.abs(stats.zscore(ceny)) > 3
```

WSKAZÓWKA Korelacja nie mówi, co jest przyczyną. Istotna różnica nie znaczy duża: zawsze podaj też samą różnicę średnich.

05 MATPLOTLIB

Wykres krok po kroku

- **subplots**: figura i oś. Na osi (ax) rysujesz i ustawiasz wszystko.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(8, 4))
```

- **plot, bar, barh**: linia w czasie, słupki do porównań.

```
ax.plot(miesiace, sprzedaz, marker="o")
slupki = ax.bar(kategorie, wartosci)
ax.barh(produkty, wartosci)  # długie nazwy
```

- **hist, scatter**: rozkład i zależność dwóch liczb.

```
ax.hist(ceny, bins=20)
ax.scatter(reklama, sprzedaz, alpha=0.5)
```

- **Opisy**: tytuł, osie, legenda, siatka.

```
ax.set_title("Sprzedaż 2026")
ax.set_xlabel("Miesiąc")
ax.set_ylabel("PLN")
ax.legend()
ax.grid(axis="y", alpha=0.3)
```

- **Mniej szumu**: bez górnej i prawej ramki, wartości na słupkach.

```
ax.spines[["top", "right"]].set_visible(False)
ax.bar_label(slupki, fmt="%.0f")
```

- Kilka wykresów obok siebie.

```
fig, axs = plt.subplots(1, 2, figsize=(10, 4))
axs[0].hist(ceny)
axs[1].boxplot(ceny)
```

- **Zapis**: ostry PNG do raportu albo prezentacji.

```
fig.tight_layout()
fig.savefig("wykres.png", dpi=200)
plt.show()
```

06 PANDAS + WYKRES

Szybki podgląd

- **df.plot**: wykres prosto z tabeli, pod spodem matplotlib.

```
df.plot(x="Data", y="Wartosc", ax=ax)
(df.groupby("Kategoria")["Wartosc"].sum()
   .sort_values().plot.barh())
```

07 SEABORN

Wykresy z tabeli

- **Start**: motyw dla wszystkich wykresów.

```
import seaborn as sns
sns.set_theme(style="whitegrid")
```

- **histplot, boxplot**: rozkład i porównanie grup. Każda funkcja przyjmuje też ax=ax.

```
sns.histplot(data=df, x="Cena", kde=True)
sns.boxplot(data=df, x="Kategoria", y="Cena")
```

- **barplot**: estimator mówi, co liczyć, errorbar=None bez wąsów.

```
sns.barplot(data=df, x="Kategoria",
            y="Wartosc", estimator="sum",
            errorbar=None)
```

- **scatterplot, lineplot**: hue koloruje według kolumny.

```
sns.scatterplot(data=df, x="Cena",
                y="Ilosc", hue="Kraj")
sns.lineplot(data=df, x="Data",
             y="Wartosc", hue="Kraj")
```

- **heatmap**: mapa korelacji wszystkich kolumn liczbowych.

```
kor = df.corr(numeric_only=True)
sns.heatmap(kor, annot=True, fmt=".2f",
            cmap="RdBu_r", center=0)
```

WSKAZÓWKA Jeden wykres, jedna myśl. Tytuł mówi wniosek („Online rośnie szybciej"), a nie temat („Sprzedaż wg kanału").

08 KTÓRY WYKRES

Szybki wybór

- **Zmiana w czasie**: linia.
- **Porównanie kategorii**: słupki poziome, posortowane.
- **Rozkład jednej liczby**: histogram albo boxplot.
- **Zależność dwóch liczb**: scatter.
- **Wiele korelacji naraz**: heatmap.

Nowe ściągi i materiały dostajesz najpierw w newsletterze. Zapisz się sklep.ninjadata.pl/newsletter/