NINJA DATA

PYTHON 3

Python Cheat Sheet Nowicjusz

Podstawy Pythona przed pandas: zmienne, tekst, listy, warunki, pętle i funkcje.

ŚCIĄGA | NOWICJUSZ | WOJOWNIK | MISTRZ

01 START — Zmienne i typy

- **print**: wypisuje wynik. Pierwsza rzecz do sprawdzania, co się dzieje.

```
print("Cześć, dane!")
print("Suma:", 2 + 3)
```

- **Zmienna**: nazwa dla wartości. Bez deklarowania typu.

```
miasto = "Gdańsk"
sprzedaz = 1250.50
klienci = 42
```

- **Typy**: int (całkowite), float (z przecinkiem), str (tekst), bool (True albo False), None (brak wartości).

```
type(klienci)      # <class 'int'>
aktywny = True
rabat = None
```

- **Zmiana typu**: tekst z pliku zwykle trzeba zamienić na liczbę.

```
int("42")          # 42
float("12.5")      # 12.5
str(2026)          # "2026"
```

WSKAZÓWKA Nazwy zmiennych małymi literami, słowa przez podkreślnik: `cena_netto`, `liczba_klientow`.

02 LICZBY — Działania i porównania

- **+ - * /**: dzielenie / zawsze daje float.

```
7 / 2              # 3.5
```

- **//, %, ****: dzielenie całkowite, reszta, potęga.

```
7 // 2             # 3
7 % 2              # 1
2 ** 10            # 1024
```

- **round, abs, min, max**: zaokrąglenie i proste funkcje.

```
round(19.987, 2)   # 19.99
```

- **Porównania dają True albo False**: == != > < >= <=

```
cena = 120
cena >= 100        # True
```

- **and, or, not**: łączysz warunki.

```
cena > 50 and kraj == "PL"
```

03 TEKST — Napisy (str)

- **f-string**: wstawiasz zmienne w tekst. Najwygodniejszy sposób.

```
imie = "Ala"
print(f"{imie} ma {klienci} klientów")
print(f"Kwota: {sprzedaz:,.2f} zł")
```

- **Metody tekstu**: wynik to nowy tekst, oryginał się nie zmienia.

```
" Gdańsk ".strip()       # "Gdańsk"
"sql".upper()            # "SQL"
"Ala ma kota".replace("kota", "psa")
```

- **len, in**: długość tekstu, czy zawiera fragment.

```
len("Python")            # 6
"Pro" in "Laptop Pro"    # True
```

- **split, join**: tekst na listę i z powrotem.

```
"PL;DE;CZ".split(";")    # ['PL', 'DE', 'CZ']
", ".join(["a", "b"])    # "a, b"
```

- **Indeksy i wycinki**: liczone od 0, ostatni ma -1.

```
kod = "PL-2026-001"
kod[0]                   # "P"
kod[-3:]                 # "001"
kod[3:7]                 # "2026"
```

04 LISTY — Kilka wartości po kolei

- **Lista**: wartości w nawiasach kwadratowych, można zmieniać.

```
ceny = [120, 45, 300, 89]
ceny.append(150)         # dodaje na końcu
```

- **len, sum, min, max, sorted**: szybkie liczenie.

```
sum(ceny) / len(ceny)    # średnia
sorted(ceny, reverse=True)
```

- Wycinki działają jak w tekście.

```
ceny[:3]                 # pierwsze 3
```

- **List comprehension**: nowa lista z warunkiem w jednej linii.

```
drogie = [c for c in ceny if c > 100]
brutto = [round(c * 1.23, 2) for c in ceny]
```

05 WARUNKI
if, elif, else

- Wcięcie (4 spacje) mówi Pythonowi, co należy do warunku.

```
if sprzedaz >= 10000:
    status = "Cel osiągnięty"
elif sprzedaz >= 8000:
    status = "Blisko"
else:
    status = "Poniżej celu"
```

- Warunek w jednej linii, gdy są tylko dwie opcje.

```
segment = "Premium" if cena >= 500 else "Standard"
```

WSKAZÓWKA Zapomniany dwukropek albo złe wcięcie to najczęstszy błąd na starcie: `SyntaxError` i `IndentationError`.

06 PĘTLE
for i while

- **for**: robisz to samo dla każdego elementu.

```
for c in ceny:
    print(c * 1.23)
```

- **range**: liczby od 0 do n-1, albo od, do i krok.

```
for i in range(3):       # 0, 1, 2
    print(i)
list(range(2, 10, 2))    # [2, 4, 6, 8]
```

- **enumerate**: numer i wartość naraz.

```
for nr, miasto in enumerate(miasta, start=1):
    print(nr, miasto)
```

- **while**: powtarzasz, dopóki warunek jest prawdziwy.

```
proba = 1
while proba <= 3:
    proba += 1
```

- **break, continue**: koniec pętli albo przejście do kolejnego kroku.

```
for c in ceny:
    if c < 0:
        continue        # pomija ujemne
```

07 SŁOWNIKI
Klucz i wartość

- **dict**: wartości pod nazwami, jak jeden wiersz tabeli.

```
klient = {"id": 7, "miasto": "Kraków",
          "wartosc": 560.0}
klient["miasto"]          # "Kraków"
klient["segment"] = "VIP"  # nowy klucz
```

- **get**: wartość albo zapasowa, gdy klucza nie ma (bez błędu).

```
klient.get("rabat", 0)     # 0
```

08 FUNKCJE
Własne narzędzia

- **def, return**: raz piszesz, używasz wiele razy.

```
def brutto(netto, vat=0.23):
    return round(netto * (1 + vat), 2)
brutto(100)             # 123.0
brutto(100, vat=0.08)   # 108.0
```

09 BŁĘDY
try i except

- **try, except**: program nie przerywa się na złej wartości.

```
try:
    kwota = float("12,5")
except ValueError:
    kwota = None
```

- **Częste komunikaty**: NameError (literówka w nazwie), TypeError (zły typ, np. tekst + liczba), KeyError (brak klucza), IndexError (za duży indeks).

WSKAZÓWKA Czytaj błąd od dołu: ostatnia linia mówi, co się stało, linia wyżej, gdzie.

10 MODUŁY
import i pip

- **import**: gotowe narzędzia z biblioteki standardowej.

```
import math
from datetime import date
math.sqrt(16)          # 4.0
```

- **pip**: instalacja bibliotek w terminalu, potem import.

```
pip install pandas
```

Uczysz się Pythona? Odhaczaj tematy dzień po dniu. **Tracker 100 dni Python** sklep.ninjadata.pl/tracker-100-dni-nauki-python/