# Wprowadzenie

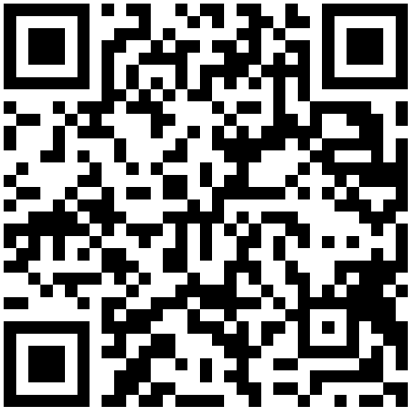
*Podstawowy warsztat AI*

Cel: zapewnienie elementarnej znajomości narzędzi potrzebnych w AI oraz na studiach matematycznych.

- Wstęp do Pythona (na potrzeby laboratoriów z Algebry liniowej i Analizy matematycznej).
- Obsługa Linuksa.
- Systemy składania tekstu (LaTeX).
- Systemy kontroli wersji ($\mathrm{git}$ + GitHub).
- (Inne narzędzia AI.)

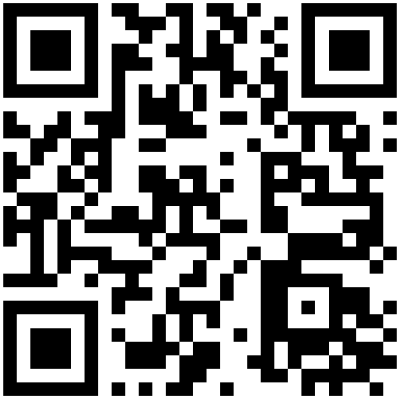
# Wprowadzenie

Strona wykładu: https://math.uni.wroc.pl/~jagiella/pwai/



# Wprowadzenie

(powtórzenie) Wczorajsza ankieta: https://forms.office.com/e/mrUsT9vWJT



# Wprowadzenie

Wstępne* warunki zaliczenia przedmiotu:

- Cotygodniowe, punktowane listy zadań.
- Mały projekt na zakończenie.

Listy zadań:

- Będą pojawiać się na stronie laboratoriów (link na stronie wykładu) po każdym wykładzie (w tym dziś) i obowiązują na laboratoria w tygodniu następnym.
- Dokładny sposób rozliczania danej listy będzie na niej podany.

Informacje o projekcie zostaną podane później.

Możliwe będzie zwolnienie z części lub całości list zadań (ale nie z projektu) na indywidualnych zasadach. Szczegóły sposobu zwolnień oraz sposobu wystawienia ocen zostaną podane w ciągu dwóch tygodni od rozpoczęcia semestru.

# Python

Tygodnie 1-5: crash course Pythona.
Python – popularny* język programowania ogólnego zastosowania.

- Python 1 (pierwsza wersja w 1994), Python 2 (2000), **Python 3 (2008)**, obecnie Python 3.13 (2024). (Debiut Pythona 3.14: we wtorek.)
- Szeroko używany w analizie danych i w konsekwencji w uczeniu maszynowym.
- Eksplozja popularności około 2020 (wraz z eksplozją AI).
- Język wyboru dla projektów uczenia maszynowego, AI.

Pewne powody popularności:

- Wysokopoziomowy język z prostą składnią.
- Dostepny na wielu platformach.
- Bogaty ekosystem bibliotek i frameworków dla statystyki, analizy danych, ML.
- Nie wymaga wielkiego nakładu pracy do eksperymentowania i pisania prototypów.
- Dużo zasobów wiedzy (podręczniki, tutoriale, blogi, interaktywne kursy, . . . )

# Python

Środowisko na wykładzie:

1. „Oficjalny interpreter" (CPython): https://python.org
2. PyCharm (IDE): https://www.jetbrains.com/pycharm/
3. Później: Jupyter Notebook/Lab: https://jupyter.org/

*(krótka demonstracja interpretera, ewaluacji wyrażeń, przypisań)*

Instrukcje i niuanse instalacji oraz używanie ww. narzędzi: na laboratoriach.

Uwagi według punktów:

1. CPython jest de facto główną implementacją Pythona, istnieje jednak wiele innych (https://wiki.python.org/moin/PythonImplementations).
2. PyCharm jest już zainstalowany na komputerach w pracowniach. Oprócz wersji bezpłatnej dla studentów uczelni wyższych dostępna jest też wersja edukacyjna (z dodatkowymi opcjami). Alternatywne środowiska: VS Code + plugin.

# Terminologia

Języki programowania – opis przez składnię (syntax) i semantykę (semantics).

**Składnia** – reguły stanowiące, co jest poprawnym wyrażeniem lub instrukcją w języku.
*Np. reguły konstrukcji poprawnych wyrażeń arytmetycznych (operatory muszą pojawiać się między podwyrażeniami, nawiasy otwarte muszą zostać zamknięte, etc.).*

**Semantyka** – znaczenie poprawnych wyrażeń języka.
*Np. „a + b" to wyrażenie, które wylicza się do sumy a i b.*

Dwa rodzaje „napisów" w języku: instrukcje (statements) i wyrażenia (expressions).

**Wyrażenie** – wylicza się do pewnej wartości (konkretnego obiektu).

**Instrukcja** – powoduje zmianę stanu programu, nie posiada wartości (np. przypisanie).

# Wyrażenia arytmetyczne

W Pythonie pracujemy z obiektami (liczby, napisy, wartości logiczne, bardziej złożone byty – więcej szczegółów na dalszych wykładach).

Wyrażenia arytmetyczne: wyrażenia reprezentujące (wyliczające) liczby.

Skonstruowane z:

- operatorów arytmetycznych, liczb, nawiasów,
- nazw reprezentujących obiekty,
- pewnych podwyrażeń (o tym później).

Operatory arytmetyczne (w kolejności wykonywania):

1. ** (potęgowanie),
2. +, - (jednoargumentowy plus i minus),
3. *, /, %, // (mnożenie, dzielenie, modulo, dzielenie całkowite),
4. +, -,
5. ...

# Więcej o przypisaniu

**Nazwa** – identyfikator obiektu. Dowolnemu obiektowi można nadać nazwę i używać jej w kodzie, reprezentując ten obiekt.

```
r = 5
area = 3.14 * r ** 2
```

Wartość wyrażenia z prawej strony wyliczana jest raz – w momencie przypisania.
Nazwa działa jak etykietka (alias) i **nie jest** tożsama z samym obiektem.

```
a = 1
a = "Jestem napisem"
```

Semantyka przypisania w innych językach:
JavaScript, Ruby: prawie identyczna
Java, C#: podobna (nazwa $\approx$ reference).
C/C++: zupełnie inna (najlepsze przybliżenie: nazwy jak wskaźniki)

# Wyrażenia ogólnie

Wyrażenia konstruowane z:

- operatorów, stałych, nawiasów,
- nazw reprezentujących obiekty,
- podwyrażeń.

Niektóre operatory (w kolejności wykonywania):

1. arytmetyczne,
2. $<=$, $<$, $>$, $>=$ – nierówności,
3. $==$, $!=$ – równość, nierówność,
4. $\mathrm{not}$ – negacja,
5. $\mathrm{and}$ – koniunkcja,
6. $\mathrm{or}$ – alternatywa.

# Wejście/wyjście

Wejście/wyjście: wbudowane w język input() i print().

```
print(wyrazenie1, wyrazenie2, ...)
```

np.

```
print("Hello", "World!", 12 + 10)
Hello World! 22
```

```
s = input("Podaj napis")
```

s będzie napisem, ale możemy spróbować uzyskać zamiast niego liczbę całkowitą:

```
s = int(s)
```

Można też:

```
n = int(input("Podaj liczbę całkowitą"))
print(n + 1)
```

# Przepływ sterowania

Instrukcja warunkowa.

```
if wyrazenie: instrukcja
```

```
if wyrazenie:
    instrukcja1
    instrukcja2
    ...
```

```
if wyrazenie1:
    instrukcja1
    instrukcja2
    ...
elif wyrazenie2: # dowolnie wiele razy
    instrukcja3
    ...
else:
    instrukcja4
    ...
```