Wstęp do programowania
semestr zimowy 2024/2025

Dr Anna Muranova
UWM w Olsztynie

Wykład 1

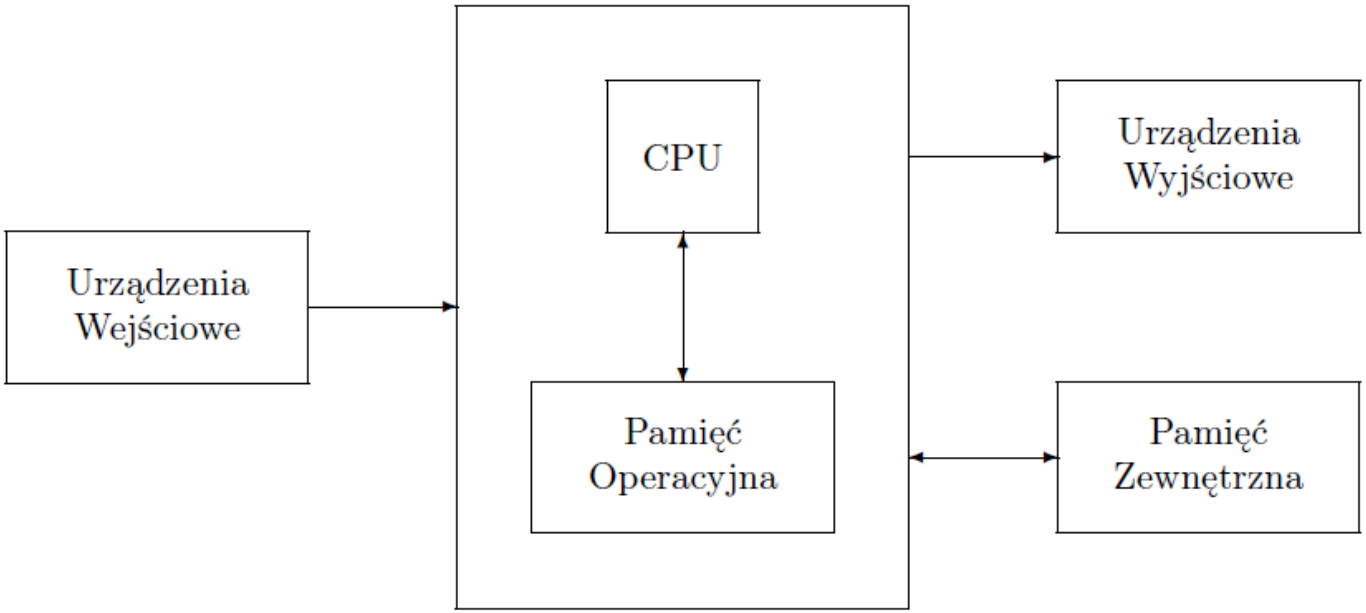
Komputery

Pojęcia podstawowe

- **Komputer** to elektroniczne urządzenie, które przechowuje i przetwarza informacje zgodnie z wykonywanym programem.
- Większość współczesnych komputerów oparta jest na tzw. architekturze von Neumanna (od nazwiska Johna von Neumanna).
- Cechą charakterystyczną tej architektury jest to, ze programy oraz dane do programów znajdują się w tej samej pamięci, zwanej **pamięcią operacyjną**
- Podstawowe elementy komputera to:
  - procesor (CPU – od ang. Central Processor Unit)
  - pamięć operacyjna (RAM – od ang. Random Access Memory)
  - urządzenia wejścia/wyjścia
  - pamięć zewnętrzna

Komputery

Funkcjonalny schemat komputera



Programy

- **Program komputerowy** to szczegółowy zestaw instrukcji określający działanie komputera.
- **Programowanie** to proces tworzenia kodu źródłowego programu w wybranym języku programowania.
- **Język programowania** to zbiór reguł określających, które ciagi symboli tworzą program komputerowy oraz jakie obliczenia opisuje ten program.

Programy

Postacie programu

- **Kod zródłowy** to ciąg instrukcji i deklaracji zapisany w zrozumiałym dla człowieka języku programowania.
- **Kod maszynowy** to postać programu komputerowego (nazywana wykonywalną lub binarną) przeznaczona do bezpośredniego lub prawie bezpośredniego wykonania przez procesor.
- **Postać binarna programu** jest dopasowana do konkretnego typu procesora i wyrażona w postaci rozumianych przez niego kodów rozkazów oraz ich argumentów

Języki Programowania

Realizowany przez komputer program składa się z rozkazów – instrukcji maszynowych w kodzie zero-jedynkowym. Tak zapisany program w języku wewnętrznym jest bardzo nieczytelny. Użytkownicy formułują algorytmy w językach programowania wyższego poziomu. Języki programowania posiadają:

- ściśle określony alfabet tj. zbiór możliwych słów
- ściśle określone reguły składniowe (syntaktyka)
- jednoznaczne reguły interpretujące znaczenie poszczególnych napisów (semantyka)

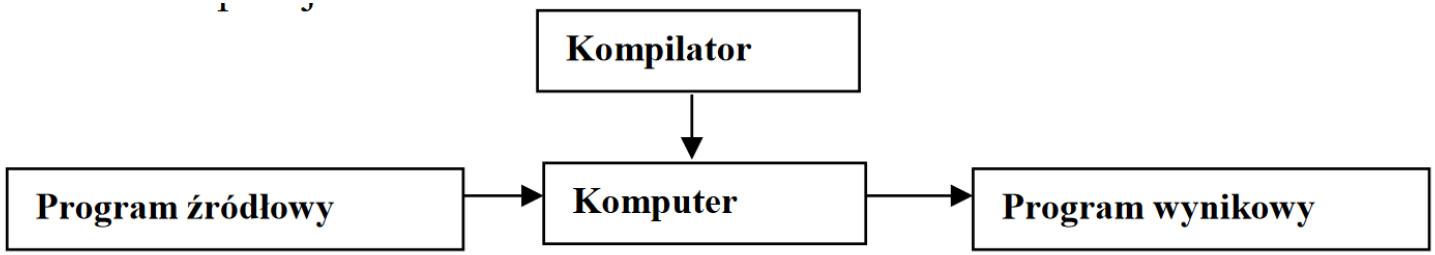
Program napisany w języku programowania może być przetłumaczony na język wewnętrzny komputera przez program nazywany translatorem. Wyróżnia się dwa rodzaje translatorów:

- kompilatory
- interpretatory

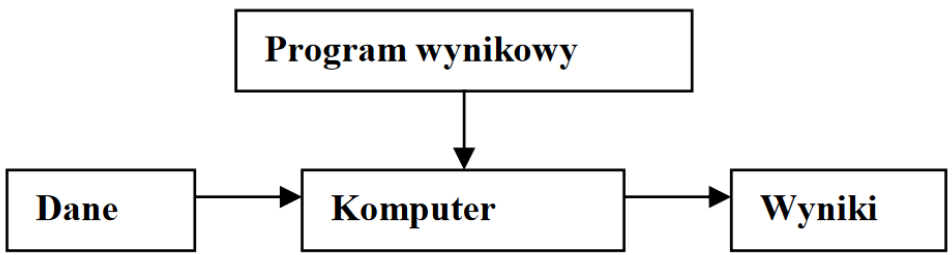
Kompilator

W przypadku wykorzystania kompilatora obliczenia prowadzone są w dwu fazach:

- 1^o faza kompilacji



- 2^o faza kompilacji



Interpretator

W przypadku wykorzystania do obliczeń interpretatora każda instrukcja programu źródłowego, po przetłumaczeniu, jest natychmiast wykonywana. Nie tworzona jest wersja wynikowa programu (najczęściej posiadająca rozszerzenie `.exe`). Za każdym razem wykonanie programu wymaga ponownego tłumaczenia.

Język Python wyposażony jest w translator typu interpretator. co daje większą łatwość modyfikacji gotowego programu, lecz obniża efektywność działania w stosunku do języków kompilowanych, takich jak C.

Paradygmaty programowania 1

Paradygmaty programowania są różnymi sposobami lub podejściami do tworzenia programów lub korzystania z języków programowania. Każdy paradygmat ma swoje własne struktury, cechy i zalecenia dotyczące rozwiązywania typowych problemów programistycznych.

- **Programowanie modularne** – paradygmat programowania zalecający stosowanie nadrzędności modułów w stosunku do procedur i bloków tworzących program. Moduł grupuje funkcjonalnie związane ze sobą dane oraz procedury i jest reprezentacją obiektu jednokrotnie występującego w programie.
- **Programowanie generyczne (uogólnione)** pozwala na pisanie kodu programu, w językach typowanych statycznie (C++), bez wcześniejszej znajomości typów danych, na których kod ten będzie pracował.
- **Programowanie proceduralne** – paradygmat programowania zalecający dzielenie kodu na procedury, czyli fragmenty wykonujące ściśle określone operacje. Procedury nie powinny korzystać ze zmiennych globalnych (w miarę możliwości), lecz pobierać i przekazywać wszystkie dane (czy też wskaźniki do nich) jako parametry wywołania.

- **Programowanie imperatywne**
- **Programowanie obiektowe**
- **Programowanie funkcyjne**
- **Programowanie deklaratywne**
- **Programowanie agentowe**
- **Programowanie współbieżne**

i inne.

Język Python

- Poprawna wymowa: pajton.
- Język Python stworzył we wczesnych latach 90. Guido van Rossum – jako następcę języka ABC.
- Najbardziej popularny języki programowania w latach 1965 – 2024: `https://youtu.be/hL85pP3KZ7c?si=cg5qMWoOAE9MsTLA`
- Nazwa języka pochodzi od serialu komediowego emitowanego w latach siedemdziesiątych przez BBC – „Monty Python's Flying Circus" (Latający cyrk Monty Pythona). Projektant, będąc fanem serialu i poszukując nazwy krótkiej, unikalnej i nieco tajemniczej, uznał tę za świetną.
- Python wspiera różne paradygmaty programowania: obiektowy, imperatywny oraz w mniejszym stopniu funkcyjny.

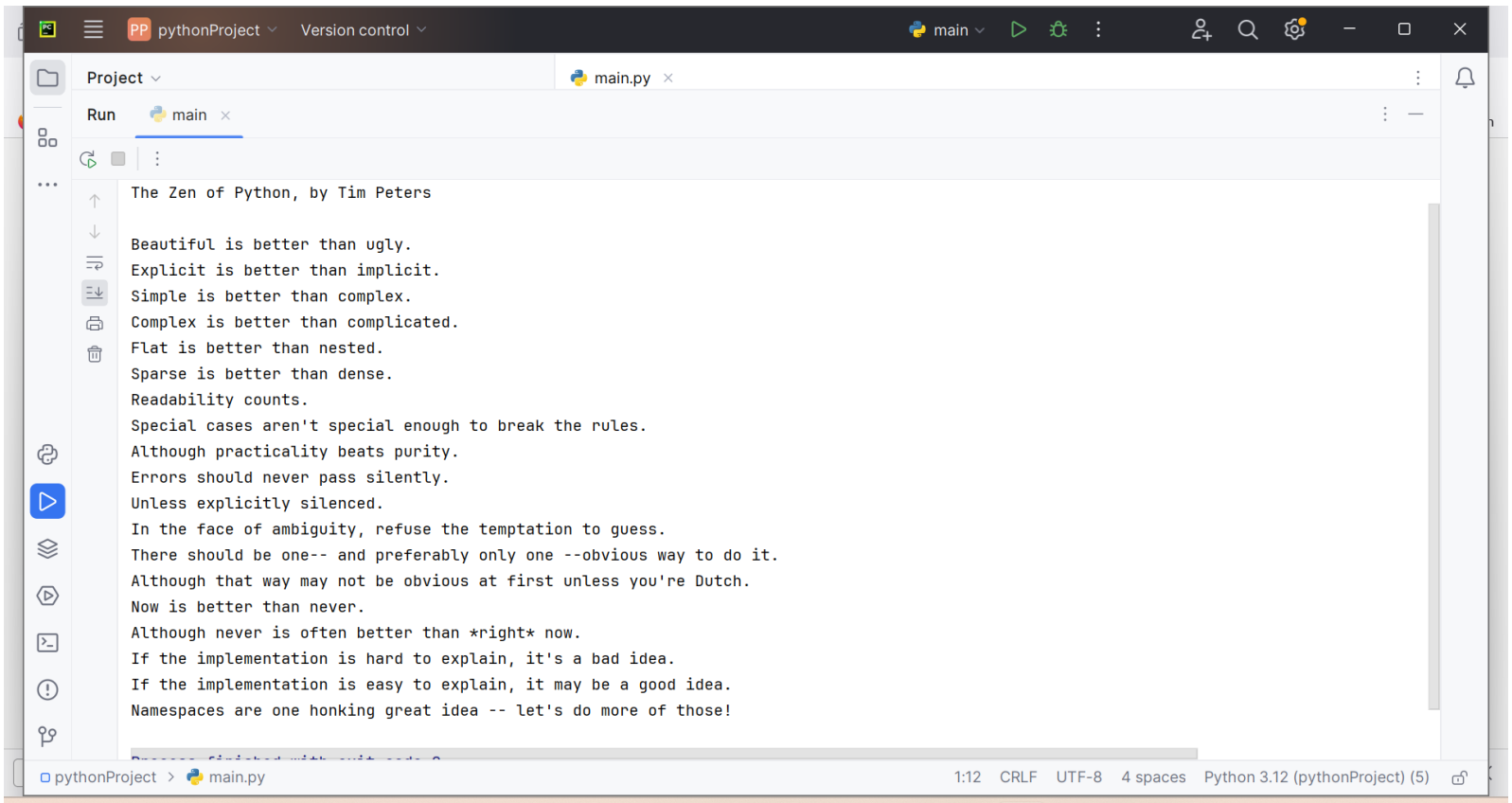
Paradygmaty programowania (Python)

- **Programowanie obiektowe** – paradygmat programowania, w którym programy definiuje się za pomocą obiektów – elementów łączących stan (czyli dane, nazywane najczęściej atrybutami) i zachowanie (czyli procedury, tzn: metody). Obiektowy program komputerowy wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań.
- **Programowanie imperatywne** – paradygmat programowania, który opisuje proces wykonywania jako sekwencję instrukcji zmieniających stan programu. Podobnie jak tryb rozkazujący w języku naturalnym wyraża żądania jakichś czynności do wykonania. Programy imperatywne składają się z ciągu komend do wykonania przez komputer. Rozszerzeniem (w sensie wbudowanych funkcji) i rodzajem (w sensie paradygmatu) programowania imperatywnego jest programowanie proceduralne.
- **Programowanie funkcyjne** – filozofia i metodyka programowania będąca odmianą programowania deklaratywnego, w której wykorzystuje się to, że funkcje należą do typów pierwszoklasowych. Kładzie się nacisk na obliczanie wartości funkcji (często rekurencyjnych), ich kompozycje oraz funkcje wyższego rzędu.

Zen of Python

Zen of Python to lista założeń w okół których był tworzony ten język. Co ciekawe, zasady te są dostępne z poziomu samego języka. Spróbuj uruchomić taki plik:

```
import this
```



Zen of Python 1

```
Piękne jest lepsze niż brzydkie.
Wyrażone wprost jest lepsze niż domniemane.
Proste jest lepsze niż złożone.
Złożone jest lepsze niż skomplikowane.
Płaskie jest lepsze niż wielopoziomowe.
Rzadkie jest lepsze niż gęste.
Czytelność się liczy.
Sytuacje wyjątkowe nie są na tyle wyjątkowe, aby łamać reguły.
Choć praktyczność przeważa nad konsekwencją.
Błędy zawsze powinny być sygnalizowane.
Chyba że zostaną celowo ukryte.
W razie niejasności powstrzymaj pokusę zgadywania.
```

Zen of Python 2

```
Powinien być jeden -- i najlepiej tylko jeden --
            oczywisty sposób na zrobienie danej rzeczy.
Choć ten sposób może nie być oczywisty jeśli nie jest się Holendrem.
Teraz jest lepsze niż nigdy.
Chociaż nigdy jest często lepsze niż natychmiast.
Jeśli rozwiązanie jest trudno wyjaśnić, to jest ono złym pomysłem.
Jeśli rozwiązanie jest łatwo wyjaśnić, to może
        ono być dobrym pomysłem.
Przestrzenie nazw to jeden z niesamowicie genialnych pomysłów --
        miejmy ich więcej!
```

Interpretator + IDE (Zintegrowane środowisko programistyczne)

Zintegrowane środowisko programistyczne, IDE (od ang. *integrated development environment*) – program lub zespół programów (pakiet, środowisko) służących do tworzenia, modyfikowania, testowania i konserwacji oprogramowania.
Programy będące zintegrowanymi środowiskami programistycznymi charakteryzują się tym, że udostępniają złożoną, wieloraką funkcjonalność obejmującą edycję kodu źródłowego, kompilowanie kodu źródłowego, tworzenie zasobów programu (tzn. szablonów/ekranów/okien dialogowych, menu, raportów, elementów graficznych jak ikony, obrazy), tworzenie baz danych, komponentów i innych.

Oprogramowanie: interprteter

Interpretator:
Python w wersji 3.12.
`https://www.python.org/`

- W systemie Linux: W Ubuntu Python 3 jest domyślnie zainstalowany
- Interpreter Pythona 3 można bezpłatnie pobrać ze strony
  `https://www.python.org/downloads/release/python-3120/`

  Pobieramy odpowiedni plik poprzez link:
  `Windows x86-64 executable installer`
  albo poprzez link
  `Windows x86 executable installer`

  Instalujemy Pythona klikając na pobrany plik.

Oprogramowanie: IDE

IDE:
PyCharm: `https://www.jetbrains.com/pycharm/`
Inne IDE:

- IDLE (domyślny), dokumentacja: `https://docs.python.org/3/library/idle.html`
- Spyder IDE: `https://www.spyder-ide.org/`
- Visual Studio: `https://visualstudio.microsoft.com/pl/vs/features/python/`
- Visual Studio Code + odpowiednie rozszerzenia: `https://code.visualstudio.com/`
- Atom + ide-python: `https://atom.io/packages/ide-python`
- i wiele innych ...

Tryby pracy

Praca w środowisku programistycznym Pythona może odbywać się na dwa sposoby:

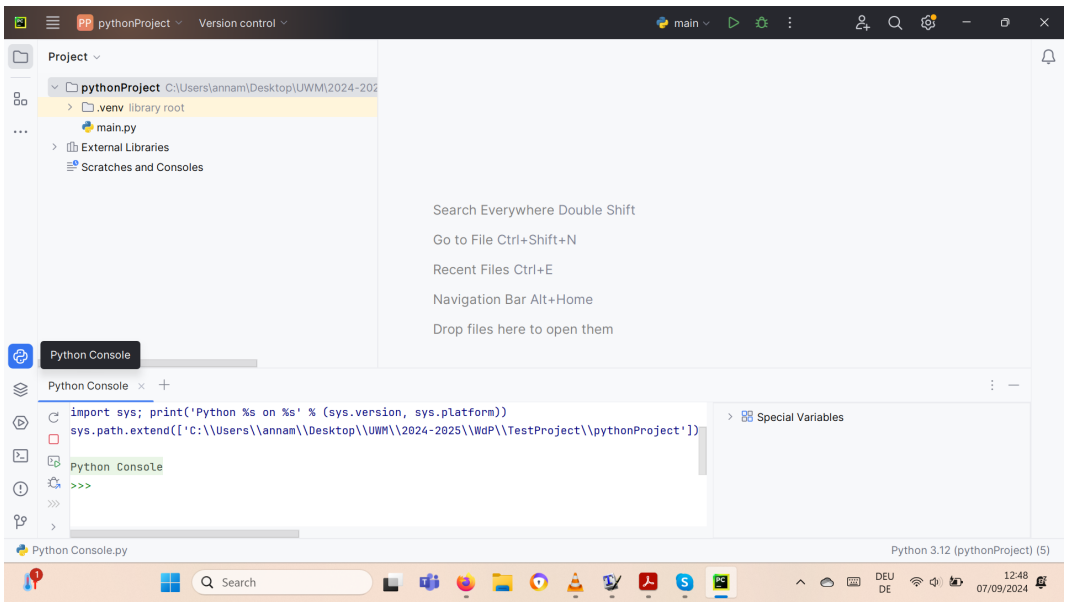
- tryb interaktywny (linia poleceń) – wprowadzane są pojedyncze instrukcje i natychmiast wykonywane
- tryb skryptowy (aplikacyjny)– grupa instrukcji zapisywana jest w pliku nazywanym skryptem (rozszerzenie `.py`) a następnie sekwencyjnie wykonywana.

Tryb interaktywny

Uruchomienie konsoli w PyCharm:

Main menu –> Tools –> Python or Debug Console

albo



Tryb interaktywny

Podstawowe operacje arytmetyczne:

$+$	dodawanie
$-$	odejmowanie
$*$	mnożenie
$/$	dzielenie
$**$	potęgowanie

Porządek: potęgowanie, mnożenie i dzielenie, dodawanie i odejmowanie.
Przykłady:

```
5+3
5+3*2
2*5**2
2-5**2
-3**2
10-3*2**2
125/5**2
4/0
```

Nawiasy

Dla zmiany kolejności operacje używają się nawiasy:

```
5+3*2
(5+3)*2
(2*5)**2
(2-5)**2
(-3)**2
10-(3*2)**2
(10-3*2)**2
(10-3)*2**2
(125/5)**2
```

Liczbowy typu danych

- `int`: całkowity
- `float`: zmiennoprzecinkowy
- `complex`: zespolony – nie omawiamy teraz.

Sprawdzanie typu

```
type(15)#<class 'int'>
type(0.5)#<class 'float'>
type(3/2)#<class 'float'>
type(15.0)#<class 'float'>
```

wszystko jest obiektem

Typ int

- bez kropki dziesiętne
- może być dowolnie długi (ograniczenie ilość pamięci)

```
123456789101112131415161718192O+1
123456789101112131415161718192l*5
123456789101112131415161718192l**2
10000**1000
10000**10000
```

- Jaki system liczbowy? Domyślnie dziesiętny

```
101
0x101
0o101
0b101
0X101
0O101
0B101
```

Typ `float`

Liczby rzeczywiste zazwyczaj zapisuje się w pamięci komputera za pomocą tzw. techniki zmiennego przecinka (z ang. *floating point*), stąd często stosowana nazwa – liczba zmiennoprzecinkowa. Praktyczną konsekwencją jest to, że liczba jest zapisywana z określoną dokładnością.

```
125/2 #62.5
125/5 #25.0
7.4 #7.4
4.#4.0
.4 #0.4
float(32)
```

Uwaga:

```
0.3-0.1*3 #-5.551115123125783e-17
```

Uwaga: Zajętość pamięci wszystkich typów danych oraz zakres danych typu float mogą się różnić w zależności od komputera, na którym jest uruchamiany program.

Typ `float`: zapis wykładniczy

```
3e4 # 30000.0
.3e4 #3000.0
2e-2 #0.02
1.79e308 #1.79e+308
1.79e308+1 #1.79e+308
1.8e308 #inf
5e-324 #5e-324
5e-324+1 #1.0
1e-325 #0.0
```

Typ `float` oraz `int`

//	dzielenie liczb całkowitych lub iloraz przy dzieleniu z resztą
%	reszta

```
3/2 #1.5
3//2 #1
type(3/2) #<class 'float'>
type(3//2) #<class 'int'>
2/2 #1.0
2//2 #1
3.5//2 #1.0
3%2 #1
5%3 #2
5.5%2 #1.5
```