# Wstęp do programowania
# semestr zimowy 2025/2026

Dr Anna Muranova
UWM w Olsztynie

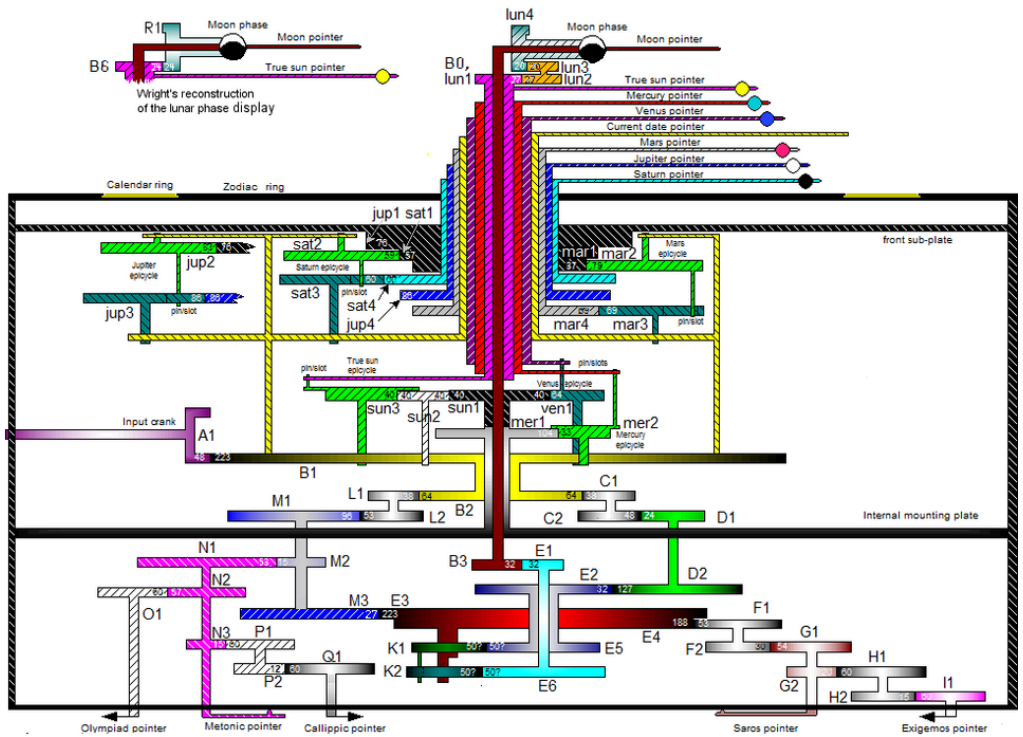
Wykład 1

# Informatyka

Informatyka (niem. Informatik, ang. computer science, computing, computer engineering, IT, ICT) – nauka ścisła oraz techniczna zajmująca się przetwarzaniem informacji, w tym również technologiami przetwarzania informacji oraz technologiami wytwarzania systemów przetwarzających informacje. Zajmuje się rozwiązywaniem problemów obliczeniowych, opisem procesów algorytmicznych oraz tworzeniem programów komputerowych i częściowo sprzętu komputerowego (z wyłączeniem zagadnień materialnych i energetycznych). Informatyka początkowo stanowiła część matematyki, z czasem wyodrębniła się do oddzielnej dyscypliny. W języku polskim termin „informatyka" zaproponował w październiku 1968 r. Romuald Marczyński (w Zakopanem, na ogólnopolskiej konferencji podczas wykładu zatytułowanego „Informatyka, czyli maszyny matematyczne i przetwarzanie informacji"), na wzór francuskiego informatique i niemieckiego Informatik.

# Historia informatyki 1

Jako pierwszy przykład komputera jest również wymieniany mechanizm z Antykithiry zbudowany między 150 a 100 rokiem p.n.e. Mechanizm służył do obliczania ruchu Słońca, Księżyca, Merkurego, Wenus, Marsa, Jowisza i Saturna, przewidywania zaćmień Księżyca i Słońca.




Źródło obrazków: Wikipedia
`https://pl.wikipedia.org/wiki/Mechanizm_z_Antykithiry`

# Historia informatyki 2

Pierwsze próby konstrukcji maszyn mechanicznych mogących dokonywać automatycznych obliczeń podjęto w wiekach XVII–XIX. Maszyny liczące konstruowali m.in. Wilhelm Schickard (1623 lub 1624), Blaise Pascal (tzw. Pascalina; 1642 lub 1641–1645), Samuel Morland (1666 lub 1668) czy Gottfried Wilhelm Leibniz (1671).

Źródło obrazku: Wikipedia `https://pl.wikipedia.org/wiki/Pascalina`

## Historia informatyki 3

W latach 30. XIX wieku Charles Babbage skonstruował uniwersalną automatyczną maszynę analityczną. Jej podstawy opracował w 1822 roku. Była ona wyposażona w pamięć i jednostkę liczącą sterowaną programem zapisanym na karcie perforowanej. Współpracowniczką Babbage'a była lady Ada Lovelace, której przypisywane jest napisanie w 1843 roku ciągu szczegółowych instrukcji, które pozwalały w sposób systematyczny wygenerować liczby Bernoulliego. Zestaw instrukcji uchodzi za pierwszy zapis programu komputerowego. W XIX w. prototypy maszyn analitycznych opracowali również George Boole i Augustus De Morgan. Boole zasłynął również z wprowadzenia metod algebraicznych do logiki, współtworząc logikę matematyczną.

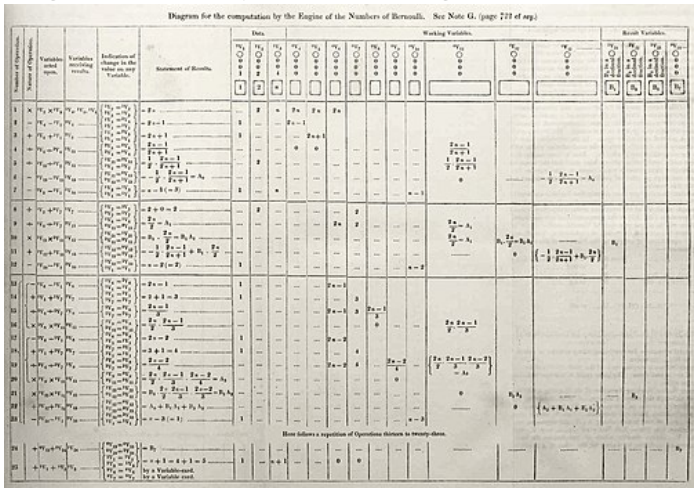
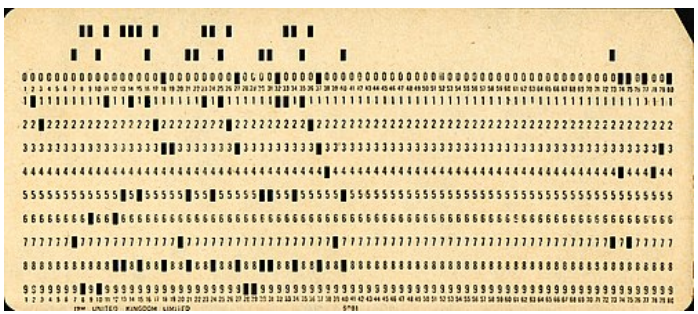
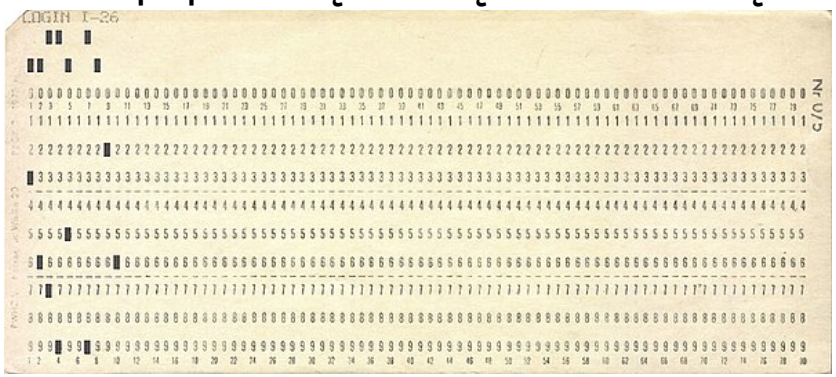


Diagram będący pierwszym opublikowanym algorytmem przeznaczonym do wykonania na maszynie analitycznej.
Źródło obrazku: Wikipedia `https://pl.wikipedia.org/wiki/Ada_Lovelace`

# Historia informatyki 4

Karta dziurkowana, karta perforowana – nośnik danych stosowany do zapisu informacji w maszynach z automatycznym przetwarzaniem danych. Używana do programowania komputerów począwszy od ich konstrukcji aż do lat 80. XX wieku, w ostatnim okresie stosowana z papierową taśmą dziurkowaną.





Karta dziurkowana IBM 12/80 oraz karta dziurkowana używana do wprowadzania programów do komputerów oraz danych w byłym Centrum Informatycznym Politechniki Wrocławskiej
Źródło obrazków `https://pl.wikipedia.org/wiki/Karta_dziurkowana`

## Zasady działania kart dziurkowanych 1

Kodowanie informacji:

- na prostokątnej, papierowej karcie kolejne pozycje są albo przedziurkowane albo pozostawione nienaruszone;
- każda pozycja na karcie reprezentuje pojedynczy bit informacji;
- każda kolumna reprezentuje jeden znak;
- w kolumnie znajduje się 12 pozycji;
- w formacie firmy IBM kolumn było 80, można zatem było zakodować na niej 80 znaków.

Pojedyncza dziurka w kolumnie służyła do kodowania cyfr, dwie do kodowania liter alfabetu (wielkie litery), trzy do użycia znaków specjalnych. Od 1964 r. dopuszczono stosowanie nawet 6 dziurek w kolumnie.
Karty były dziurkowane za pomocą maszyny nazywanej dziurkarką, następnie weryfikowano poprawność przy pomocy sprawdzarki (początkowo robił to człowiek). Dopiero potem karty wkładano do czytnika kart, który był jednym z peryferyjnych urządzeń wejścia do komputera. Wszystkie te działania wymagały nadzoru, bądź współpracy człowieka, operatora maszyny. Karty miały, w celu zapobieżenia przypadkom nieprawidłowego („do góry nogami" lub „na lewą stronę") umieszczenia w czytniku, obcięty narożnik – np. górny lewy; karty nieprawidłowo umieszczone w pliku natychmiast można było dzięki temu dostrzec.

## Zasady działania kart dziurkowanych 2

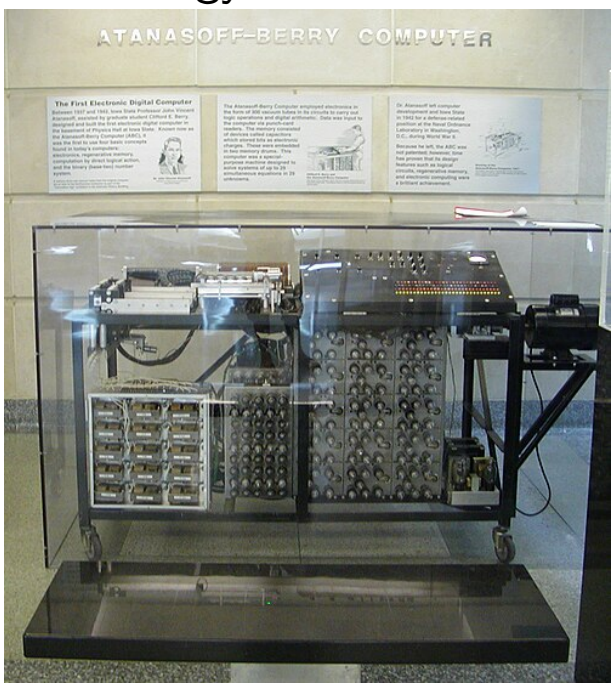
W celu zwiększenia czytelności dla człowieka na górze karty był wydrukowany napis odpowiadający wydziurkowanej treści.
System wprowadzania informacji do komputera za pomocą kart był – jak na czasy, w których go stosowano – stosunkowo nowoczesny i szybki (fotoelektryczne czytniki działały z prędkością kilkunastu kart na sekundę, co przy 80-znakowej karcie oznaczało wczytywanie w tempie do ok. półtora tysiąca znaków na sekundę, czyli znacznie szybciej niż w przypadku papierowych taśm perforowanych). Zmorą tego systemu były jednak przypadki przekłamań wynikających z resztek po dziurkowaniu „confetti", albo nawet pojedynczych włókien papieru, powstających szczególnie wówczas, gdy nóż perforatora był już stępiony), które czasami pozostawały wewnątrz otworu i fałszowały odczyt. Nierzadkie też bywały przypadki pomieszania kolejności kart, które mogły się zdarzyć w razie rozsypania stosu kart i nieprawidłowego ich późniejszego poukładania.

## Historia informatyki 5

W 1931 roku Kurt Gödel wykazał, że każde rozumowanie matematyczne można przedstawić w formie algorytmu. W 1936 roku Alan Turing przedstawił koncepcję abstrakcyjnego modelu urządzenia wykonującego algorytmy (tzw. maszynę Turinga). Pomysł Turinga zakładał nieskończenie długą taśmę podzieloną na kwadratowe komórki ułożone w szeregu.
Maszyna składa się z bloku sterowania, głowicy odczytującej i zapisującej oraz nieskończenie długiej taśmy. W każdej komórce taśmy może mieścić się jeden symbol. Maszyna zawsze jest ustawiona nad jednym z pól i znajduje się w jednym z Q stanów. Zależnie od kombinacji stanu maszyny i symbolu napotkanego na taśmie maszyna zapisuje nową wartość w polu, zmienia stan, a następnie może przesunąć się o jedno pole w prawo lub w lewo. Taka operacja nazywana jest rozkazem. Maszyna Turinga jest sterowana listą zawierającą dowolną liczbę takich rozkazów. Czasem dopuszcza się też stan M+1, który oznacza zakończenie pracy maszyny. Lista rozkazów dla maszyny Turinga może być traktowana jako jej program.
W 1937 roku Claude Shannon zaprezentował syntezę technologii elektromechanicznej, algebry Boole'a i systemu binarnego. John von Neumann (przez wiele lat nazywany „ojcem komputerów") opracował ideę maszyny cyfrowej, która miała opierać się m.in. na binarnym układzie arytmetycznym i rozdziale programu i danych w pamięci.

## Historia informatyki 6

W latach 1937–1942 John Vincent Atanasoff i jego asystent Clifford Berry zbudowali komputer Atanasoff-Berry Computer (ABC), będący pierwszym elektronicznym komputerem cyfrowym. Choć ABC był pierwszym komputerem cyfrowym, aż do 1973 roku patent na urzędzenie tego typu należał do komputera ENIAC z 1946 roku. Za wynalezienie elektronicznego komputera Atanasoff został w 1990 roku odznaczony medalem National Medal of Technology.




Źródło obrazku: Wikipedia
`https://pl.wikipedia.org/wiki/Atanasoff-Berry_Computer`

## Atanasoff-Berry Computer, ABC

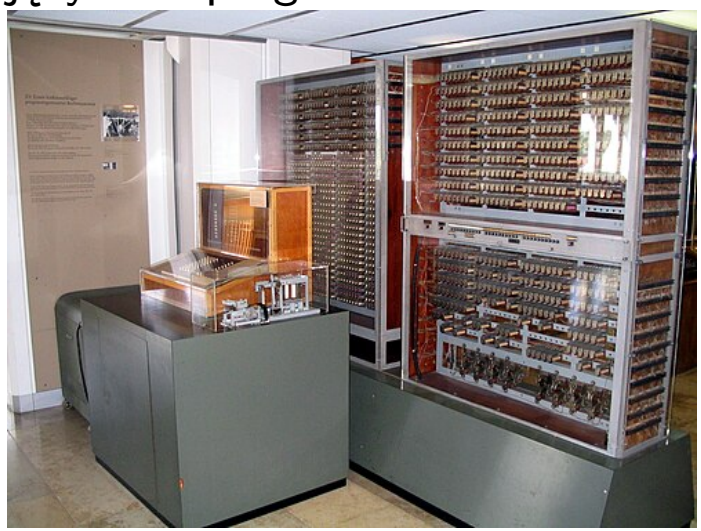
ABC – maszyna do rozwiązywania układów równań algebraicznych liniowych. Z powodu wykorzystania lamp elektronowych uważana jest za pierwszy działający prototyp specjalizowanego elektronicznego komputera cyfrowego. Została skonstruowana w 1942 roku.
Maszyna – wielkości sporego biurka – pracowała wolno i wymagała nadzoru człowieka-operatora, była około 1000 razy szybsza od stosowanych wówczas urządzeń mechanicznych. Zawierała 270 lamp elektronowych, z czego ok. 210 działało na rzecz jednostki centralnej, 30 służyło obsłudze czytnika danych i perforatora kart, a pozostałe odgrywały rolę pomocniczą.
W roku 1948 została zdemontowana bez wiedzy Atanasoffa przez pracowników Iowa State University, zachowało się tylko kilka oryginalnych elementów.
W październiku 1973 roku komputer ABC stał się powodem unieważnienia patentu na komputer ENIAC. Po wieloletnim procesie sądowym wytoczonym przez Atanasoffa twórcom ENIAC-a, sędzia federalny Earl R. Larson podjął decyzję unieważniającą patent ENIAC-a i przyznał Atanasoffowi miano wynalazcy elektronicznego komputera cyfrowego (ang. electronic digital computer).

## Historia informatyki 7: Lata 40. i 50. XX wieku

Podczas II wojny światowej udoskonalono maszyny matematyczne, głównie na potrzeby wojskowe. Alan Turing pracował nad maszynami obliczeniowymi, które umożliwiłyby złamanie kodów Enigmy. Efektem prac była tzw. bomba Turinga, stanowiąca rozwinięcie bomby kryptologicznej skonstruowanej przez Mariana Rejewskiego. Następnie Turing pracował nad Colossusem, pierwszą w pełni programowaną maszyną cyfrową do łamania szyfrów, uruchomioną w grudniu 1943 roku.
W 1944 roku na Uniwersytecie Harvarda zakończono prace nad komputerem elektromechanicznym ogólnego zastosowania ASCC (również znanym jako Harvard Mark I). Komputer przeprowadzał trzy operacje dodawania i odejmowania albo jedną operację mnożenia na sekundę. Głównym konstruktorem Mark I był Howard Aiken. W pracach nad serią komputerów Mark (I, II[w innych językach] i III[w innych językach]) aktywnie uczestniczyła Grace Hopper. Na zbudowanym w 1947 roku komputerze Mark II zarejestrowano pierwszy błąd komputerowy: w złączach znaleziono owada (ang. bug), który powodował awarie doprowadzające do generowania nieprawidłowych wyników obliczeń. Grace Hopper przypisuje się stworzenie i upowszechnienie terminu „bug" jako nazwę błędu sprzętu lub oprogramowania

## Historia informatyki 8: Lata 40. i 50. XX wieku

6 maja 1949 roku na Uniwersytecie Cambridge uruchomiono komputer lampowy z programowalną pamięcią EDSAC. Kierownikiem prac nad budową EDSAC był Maurice Wilkes. Był to pierwszy komputer, w którym przebieg programu i obróbka danych była zakodowana w pamięci urządzenia.
W III Rzeszy głównym konstruktorem maszyn liczących był Konrad Zuse. W 1938 roku stworzył maszynę liczącą Z1, która wykorzystywała system binarny i liczby zmiennoprzecinkowe. W 1941 roku Zuse uruchomił Z3. Był to pierwszy elektroniczny komputer w historii. Cztery lata później Zuse ukończył pracę nad językiem programowania Plankalkül.



Replika Z3 w Deutsches Museum w Monachium
Źródło obrazku: `https://pl.wikipedia.org/wiki/Z3`

# ENIAC

15 lutego 1946 roku w Stanach Zjednoczonych John William Mauchly i John Presper Eckert skonstruowali komputer ENIAC. Składający się z około 18 tys. lamp elektronowych ENIAC umożliwiał zsumowanie 5 tys. liczb w ciągu sekundy. Główną wadą ENIAC – a była jego skomplikowana budowa (ustawienie programu obliczeń trwało kilka godzin i polegało na połączeniu wielu kabli i ręcznym przełączaniu sześciu tysięcy przełączników), stosowane setek kart perforowanych przed uruchomieniem programu, duża energochłonność, zawodność lamp próżniowych oraz wydzielanie bardzo dużej ilości ciepła przez nie. Opinia publiczna dowiedziała się o istnieniu komputera latem 1946 roku. Powszechnie przyjmuje się, że ENIAC był pierwszym elektronicznym komputerem w historii, choć faktycznie wcześniej powstały modele Atanasoffa i Berry'ego, Zusego oraz brytyjski Colossus. Informacje o Colossusie odtajniono w 1975 roku. W październiku 1955 roku ENIAC został wyłączony.

# IBM

Prace nad komputerami prowadził również IBM. 7 sierpnia 1944 roku IBM wraz z Howardem Aikenem skonstruował cyfrową maszynę liczącą ASCC (ang. Automatic Sequence Controlled Calculator). W 1948 roku IBM uruchomiło swoją pierwszą maszynę liczącą. SSEC[w innych językach] (Selective Sequence Electronic Calculator) był kalkulatorem umożliwiającym wybór sekwencji obliczeń. Komputer był stosowany w celach naukowych, przy jego pomocy wyprodukowano tablice ruchu Księżyca, których użyto podczas misji Apollo 11.
International Business Machines Corporation (IBM) – amerykańska spółka publiczna, będąca jednym z najstarszych przedsiębiorstw informatycznych na świecie.
Obecnie przedsiębiorstwo oferuje rozwiązania w oparciu o usługi doradcze i informatyczne oraz oprogramowanie i sprzęt. Jego główną siedzibą jest Armonk w pobliżu Nowego Jorku.
W 2020 roku zatrudniało 383 tys. pracowników na całym świecie i osiągnęło przychód w wysokości 73,6 mld USD oraz zysk netto w wysokości 5,5 mld USD. Akcje przedsiębiorstwa notowane są na giełdzie New York Stock Exchange od 11 listopada 1915.

## Historia informatyki: Lata 60. i 70. XX wieku

Pierwsze komputery stosowano do obliczeń naukowo-technicznych. Ich szersze zastosowanie w ośrodkach naukowych, wojsku, a następnie wśród przedsiębiorców, nastąpiło po 1951 roku, gdy opracowano metody przetwarzania danych. W 1950 roku taśma magnetyczna zaczęła wypierać taśmy dziurkowane i karty perforowane. 14 czerwca 1951 roku uruchomiono UNIVAC I, zaprojektowany przez Johna Mauchly'ego i Johna Eckerta pierwszy komputer dostępny w wolnej sprzedaży. W tym samym okresie IBM rozpoczęła produkcję komputerów serii 700 ogólnego przeznaczenia.

# Dalej


Źródło: `https://in.pinterest.com/pin/423549539927490709/`

# Urządzenia

Wraz z rozwojem samych komputerów rozwijała się także cała branża z nimi związana.
1953 – pierwsza drukarka
1964 – mysz komputerowa
Internet: Projekt globalnej sieci komputerów opisał w 1960 roku J. Licklider. Twórcą koncepcji Internetu jest Paul Baran, który w 1962 roku opublikował 12-tomową pracę, będącą projektem wytrzymałych, rozproszonych (nie gwiaździstych) sieci cyfrowych transmisji danych, zdolnych przetrwać przewidywaną wówczas III wojnę światową, wykonanym na zlecenie Amerykańskich Sił Zbrojnych.

# Przechowywanie danych 1

- **Lata 70. – Dyskietki (Floppy Disks)** Wprowadzone przez IBM 8-calowe dyskietki stanowiły elastyczny nośnik magnetyczny do komputerów mainframe. Późniejsze, mniejsze wersje (3,5 cala) umożliwiały łatwe przenoszenie plików, dystrybucję oprogramowania i tworzenie kopii zapasowych. Ich pojemność wynosiła od 360 KB do 1,44 MB.
- **Lata 80. – Dyski twarde (Hard Disk Drives, HDD)** Zastąpiły dyskietki, oferując znacznie większą pojemność – początkowo w megabajtach, a później w gigabajtach i terabajtach. Dzięki niskim kosztom i dużej pojemności stały się podstawowym nośnikiem danych, używanym do dziś.
- **Lata 2000. – Płyty CD i DVD** Stały się popularnym medium do przechowywania muzyki, filmów i oprogramowania. Ich pojemność sięgała od 650 MB (CD) do 4,7 GB (DVD). Choć rewolucjonizowały rozrywkę, z czasem zostały wyparte przez bardziej pojemne technologie.

# Przechowywanie danych 2

- 🔑 **XXI wiek – Pamięci USB i karty SD** Przenośne, wygodne nośniki danych, które początkowo oferowały pojemność w megabajtach, a dziś sięgają nawet terabajta. Karty SD i pendrive'y umożliwiły szybki transfer danych i znacząco wpłynęły na rozwój urządzeń mobilnych.
- ⚡ **Współczesność – Dyski SSD (Solid State Drives)** Zapewniają znacznie szybszy odczyt i zapis niż HDD, co przekłada się na krótszy czas uruchamiania systemu, lepszą wydajność i niezawodność. Stosowane są zarówno w komputerach osobistych, jak i w profesjonalnych stacjach roboczych oraz centrach danych.
- ☁️ **Era obecna – Przechowywanie w chmurze (Cloud Storage)** Umożliwia zdalny dostęp do danych z dowolnego miejsca na świecie. Oferuje praktycznie nieograniczoną pojemność, elastyczność oraz niższe koszty utrzymania infrastruktury, stając się dominującym rozwiązaniem w przechowywaniu danych.

Źródło: `https://medium.com/@khankhushboo919/from-floppy-to-the-cloud-exploring-the-evolution-of-datatypes-c594a349`

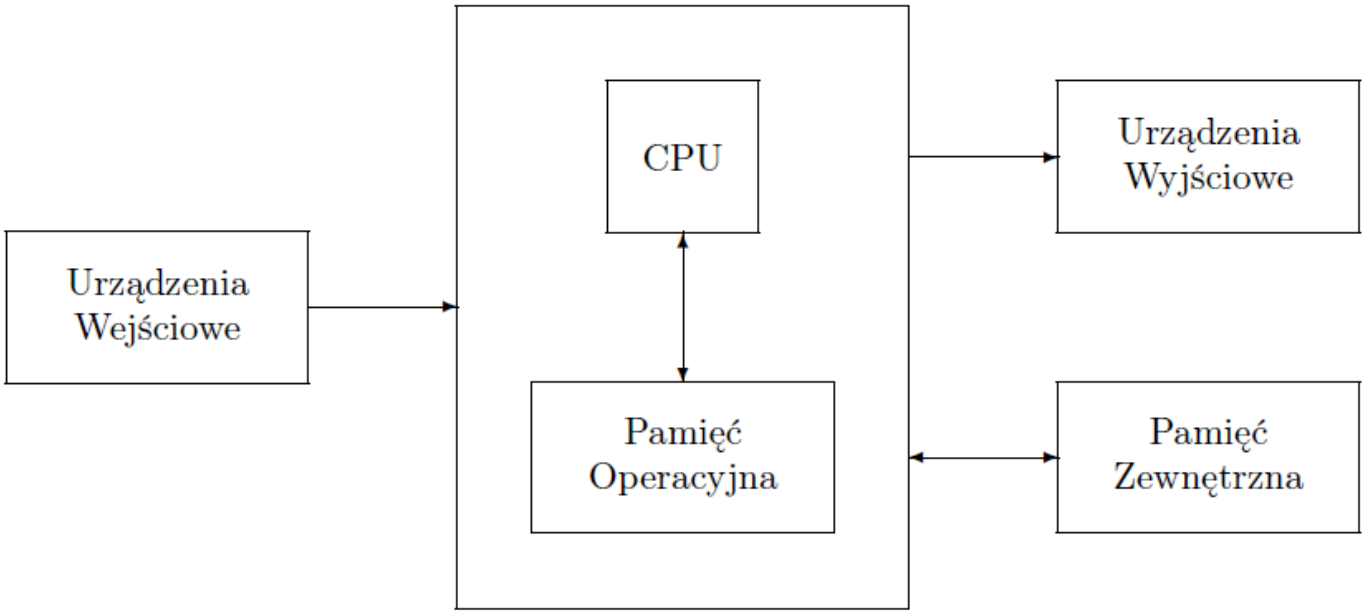
# Komputery

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)

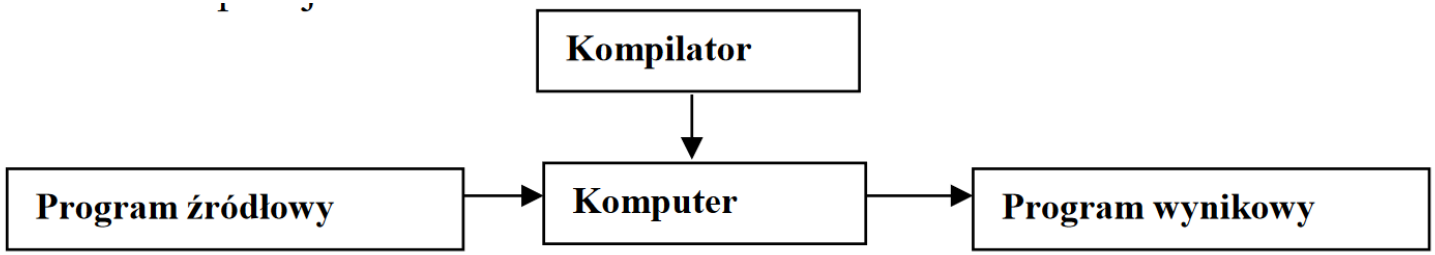
Program napisany w języku programowania może być przetłumaczony na język wewnętrzny komputera przez program nazywany translatorem. Wyróżnia się dwa rodzaje translatorów:

- kompilatory
- interpretatory

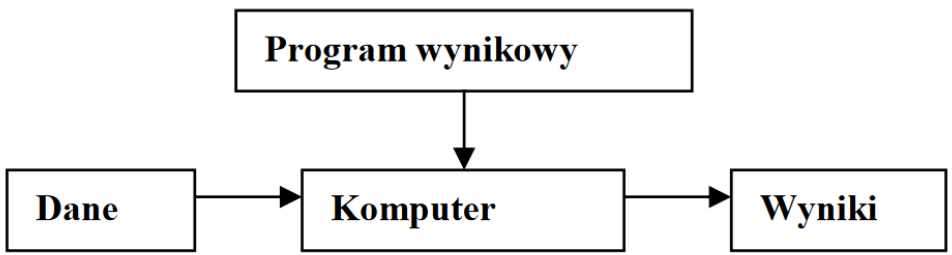
# Kompilator

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.

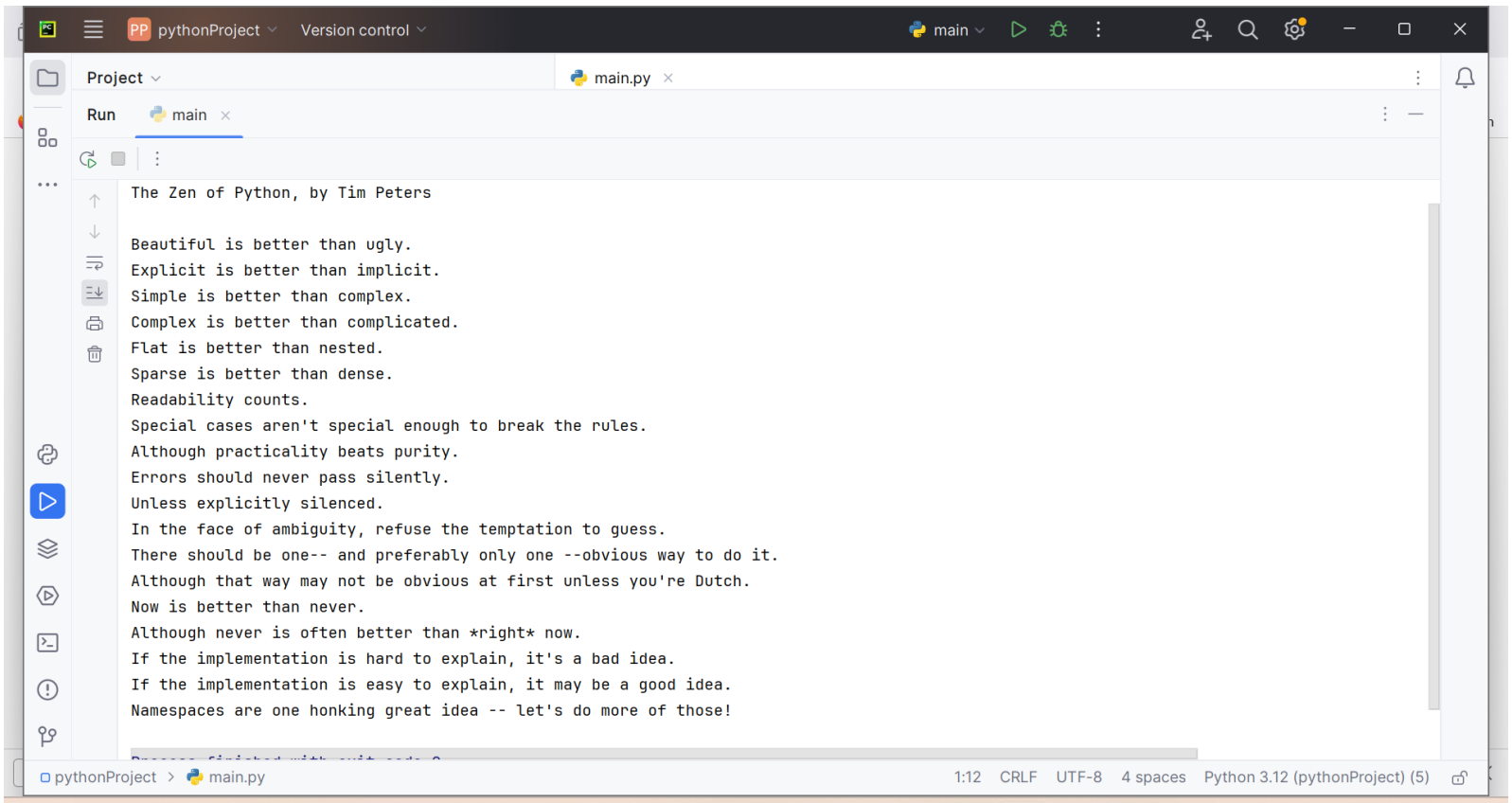
## Paradygmaty programowania (Python)

- **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:

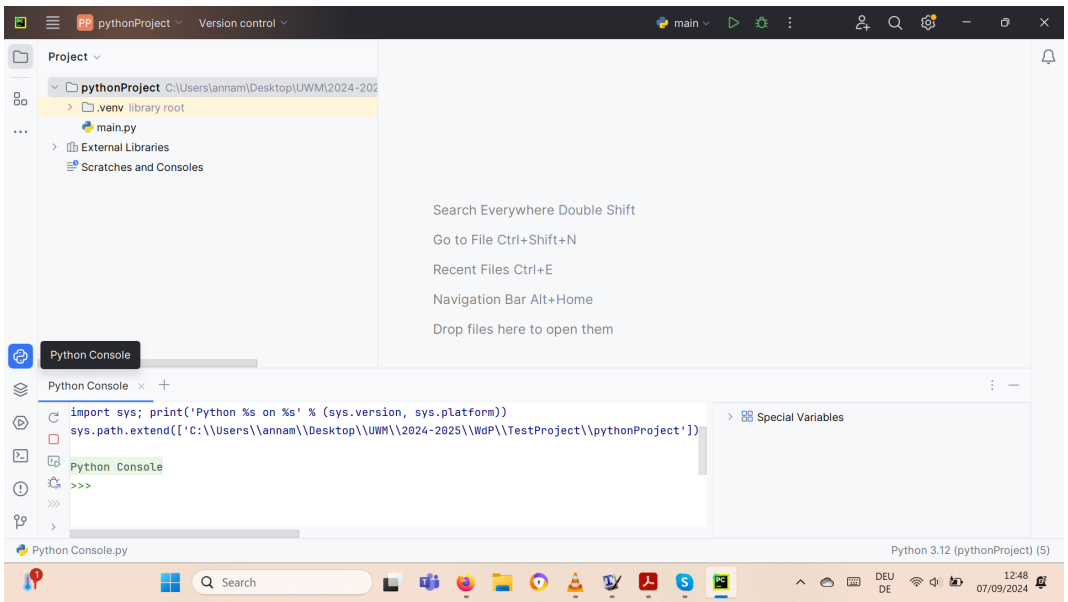
- tryb interaktywny (linia poleceń) – wprowadzane są pojedyncze instrukcje i natychmiast wykonywane
- tryb skryptowy (aplikacyjny)– grupa instrukcji zapisywana jest w pliku nazywanym skryptem ( rozszerzenie `.py`) a następnie sekwencyjnie wykonywana.

# 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
123456789101112131415161718192O*5
123456789101112131415161718192O**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
```

# Python 2. vs Python 3.

- Utworzenie drugiej gałęzi rozwoju 3.x. Początkowe obie gałęzie były rozwijane niezależnie, lecz wsparcie Pythona 2.x zostało zakończone w roku 2020.
- Python 2.x cały czas jest wykorzystywany w niektórych narzędziach, np. w ArcGis Desktop
  `https://support.esri.com/en/technicalarticle/000013224`

| Polecenie | Python 2 | Python 3 |
|---|---|---|
| `3/2` | 1 | 1.5 |
| `3//2` | 1 | 1 |
| `3/2.0` | 1.5 | 1.5 |
| `3//2.0` | 1.0 | 1.0 |

Spróbować Python 2 można tu:
`https://onecompiler.com/python2`

## Potęgowanie

```
2**2 #4
2**.5 #1.4142135623730951
2**1/2 #1.0
2**(1/2) #1.4142135623730951
4**(1/2) #2.0
-4**(1/2) #-2.0
(-4)**(1/2) #(1.2246467991473532e-16+2j)
(-4)**(-1/2) #(3.061616997868383e-17-0.5j)
16**(1/4) # 2.0
type(16**(1/4)) #<class 'float'>
16**0.5  #4.0
16**0.25 #2.0
type(16**(0.25)) #<class 'float'>
```

## Typ `str`

Kolejną ważną zmienną są string'i czyli napisy
`str` – string, napisy, łańcuchy znaków.
Obecnie odchodzi się od określenia "tablica znaków".
Definiujemy je za pomocą apostrofów ('...') lub cudzysłowów ("..."). Inaczej `Hello World!` nie zadziała:

```
Hello World
"Hello World"
'Hello World'
"napis"
'napis'
"to jest 'napis'"
'i to tez jest "napis"'
'doesn't'
"doesn't"
'doesn\'t'
```

Każdy string musi zaczynać się i kończyć tym samym znakiem – nie ważne czy użyjemy apostrofów ('...') czy cudzysłowów ("...").

## Uwaga!

Stringi:

- można używać pojedynczych apostrofów jak i podwójnych cudzysłowów
- ważne, aby stosować wybraną notację konsekwentnie
- jedyny wyjątek to gdy wewnątrz `stringu` chcemy użyć cudzysłów np. `print('Oglądam film "Player One"')`

# Co jeszcze możemy zrobić ze stringami?

Ciągi znaków możemy dodawać (inaczej łączyć, inaczej konkatenować) czy mnożyć:

```
'Kto to jest?'+'To Adam'
'Kto to jest?'+'"To Adam" odpowiedziała Magda'
'herbata' * 3
'herbata' + 3
'herbata' + "3"#'herbata3'
'herbata' + str(3)
len('herbata')#7
len('herbata'*3)
```

## Przypisywanie wartości do zmiennej

- Zmienna:
  najprościej: przechowuje pewną wartość
- Operator przypisania:
  = przypisuje prawą stronę do lewej (!), często mylony z operatorem logicznym równa się ==

```
x = 5
y = "Piotr"
a = 4.5
A = 56
x, y, z = "Orange", "Banana", "Cherry"
x = y = z = "Apple"
z = 'Cherry'
y#'Apple'
b#NameError: name 'b' is not defined
a = 5
b = 10
a * b
c = a * b
c
```

## Instrukcja `print()`

`print()` – polecenie wyświetlania.

```
print("Hello World!")
```

Możesz połączyć dwa lub więcej napisów w jednym poleceniu oddzielając je przecinkiem, a spacja między nimi doda się automatycznie:

```
print("Witaj", "co u Ciebie?")
```

`print()` pozwala na wyświetlanie różnych typów, nie tylko napisów.

```
print(3)
print(3, 6, 8)
print("Ania ma:", 3, "lata")
print()
print('doesn\'t')
print('Hello \nworld')
print('Hello' , 'world')
print('Hello' , 'world' , sep=' ')
print('Hello' , 'world', sep='\n')
```

# Znaki specjalne

| Znak specjalny | Znaczenie |
|---|---|
| \n | znak nowej linii, dodanie „entera” |
| \t | dodanie tabulacji ”=8 spacji |
| \’ | apostrof |
| \" | cudzysłów |
| \\ | ukośnik |

```
print("To jest\nnowe")
print("To jest\n\tnowe")
```

## Instrukcja `print()`

`print()` – polecenie wyświetlania.

```
print("Hello World!")
```

Możesz połączyć dwa lub więcej napisów w jednym poleceniu oddzielając je przecinkiem, a spacja między nimi doda się automatycznie:

```
print("Witaj", "co u Ciebie?")
```

`print()` pozwala na wyświetlanie różnych typów, nie tylko napisów.

```
print(3)
print(3, 6, 8)
print("Ania ma:", 3, "lata")
print()
print('doesn\'t')
print('Hello \nworld')
print('Hello' , 'world')
print('Hello' , 'world' , sep=' ')
print('Hello' , 'world', sep='\n')
```

## Instrukcja `print()`

```
print(5+3)
print(4*5.2)
print(9-7)
print(25%7)
print(4/5)
print(4//5)
print(4/5.0)
print(4//5.0)
print(3**0)
print(0**0)
print(4/0)
```

## print() ze zmiennymi

```
a = 5
b = 7
print(a + b)
print(s = 3)
c, d = 2, 4
print(c/d**2)
print(c/(d**2))
print((c/d)**2)
slowo = 'Ala ma'
print(slowo + 'kota')
print(slowo, 'kota')
print(slowo, 'kota', sep= '*')
print(slowo*2)
print(slowo, 3)
print(slowo+3)
```

# Zaawansowane opcje `print()`

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush =False)
```

- `objects` – to co ma być wyświetlone
- `sep` – separator, domyślnie znak spacji
- `end` – co co ma być wyświetlone na końcu, domyślnie znak końca linii
- `file` – określa gdzie mają być `objects` wyświetlone, domyślnie `sys.stdout` (domyślny ekran)
- `flush` – określa czy „wyjście" ma być buforowane przed przekazaniem do `file`, domyślne `False`

# Przykłady

```
print(1, 2, 3, 4)
print(1, 2, 3, 4, sep = '*')
print(1, 2, 3, 4, sep = '*', end = '&')
print('x', 'y', 'z', sep='', end='')
print('a', 'b', 'c', sep='', end='')
print('a', 'b', '\n', 'c')
print('sdf', 3456, -2, sep='\t')
```