PODSTAWY INFORMATYKI
JĘZYK PROGRAMOWANIA PYTHON

dr inż. Paweł Stąpór
Kontakt: p. 3.26C
e-mail: stapor@tu.kielce.pl
Katedra Informatyki Stosowanej

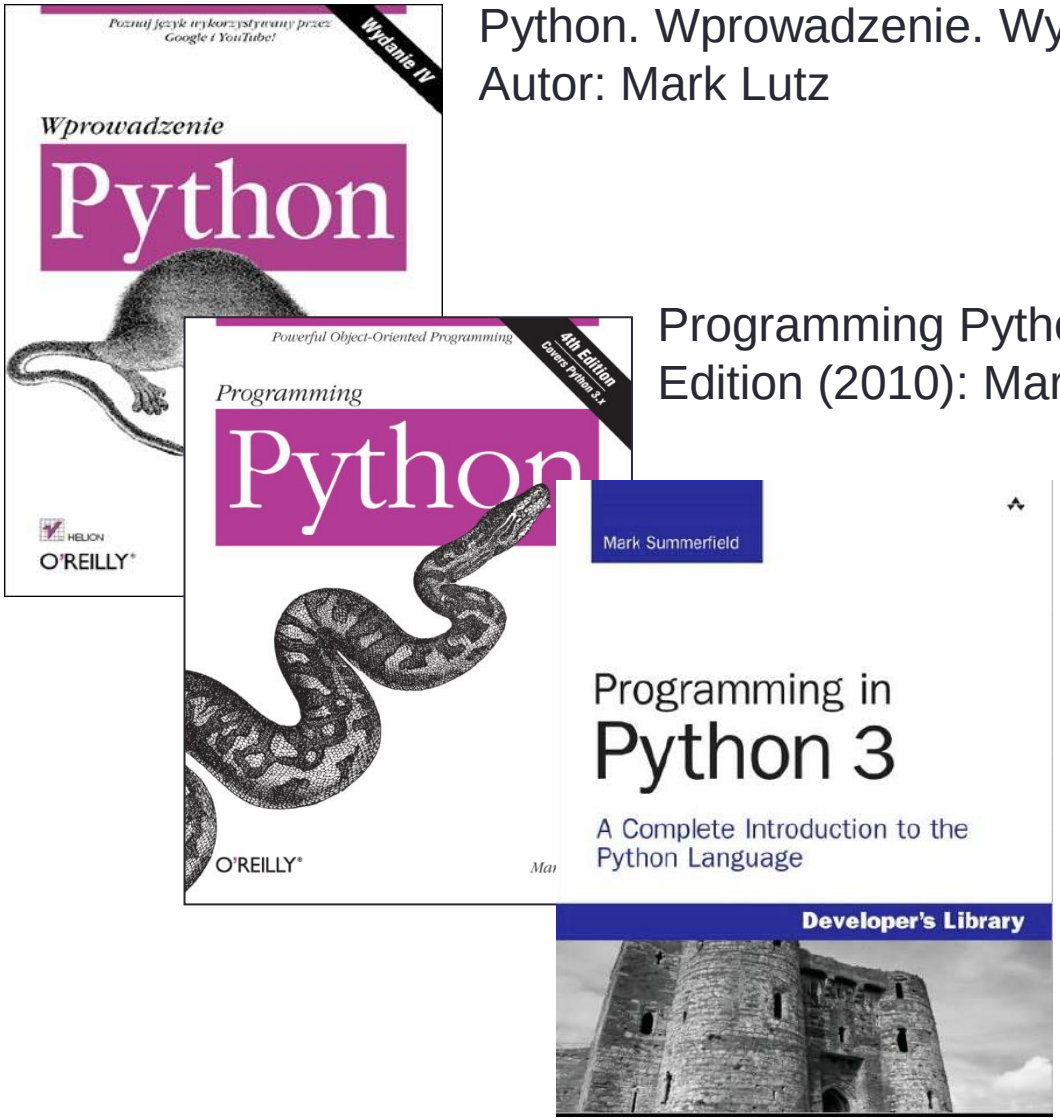
Dlaczego Python?

- Jakość
- Wydajność
- Przenośność
- Obsługa bibliotek
- Integracja
- Przyjemność

www.python.org

Źródła

https://docs.python.org/3/



Python. Wprowadzenie. Wydanie IV
Autor: Mark Lutz

Programming Python, 4th Edition (2010): Mark Lutz

Programming in Python 3
A Complete Introduction to the Python Language
Second Edition
Mark Summerfield

Komputer i program

- Komputer to elektroniczne urządzenie przetwarzające i przechowujące informacje zgodnie z wykonywanym *programem*
- Program komputerowy to szczegółowy zestaw instrukcji określający działanie komputera
- Programowanie to proces tworzenia kodu źródłowego programu w wybranym *języku programowania*
- Język programowania to zestaw zasad tekstowego lub graficznego opisu algorytmu za pomocą przyjętych elementów języka. Po implementacji algorytmu w wybranym języku opisu tekstowego powstaje *kod źródłowy programu*

Języki programowania

- **Kod maszynowy**
- Komputer jako urządzenie elektroniczne działa na zasadzie przekazywania podzespołom sygnałów binarnych – „1” (np. od 2,7 V do 5 V) lub „0” (np. od 0 V do 1,8 V). ***Kod maszynowy*** jest odpowiednikiem sygnałów binarnych, reprezentujących ciąg instrukcji obsługiwanych przez sprzęt (zapis kodu za pomocą instrukcji procesora).
- **Języki niskopoziomowe**
- Przedstawiają one instrukcje udostępniane przez system komputerowy w postaci prostych oznaczeń (o ograniczonej liczbie, zakodowane w procesorze). Do języków niskopoziomowych należą ***Assemblery.***
- **Języki wysokopoziomowe**
- Są to języki o wyższym poziomie abstrakcji, bardziej zrozumiałe dla człowieka i *niezależne od architektury elektrycznej sprzętu komputerowego.*
- Np. wyświetlenie tekstu na ekranie używając assemblera wymaga napisania kilkudziesięciu instrukcji. W języku wysokopoziomowym wystarczy wpisać np. *print(‘tekst’)*.

Języki wysokopoziomowe

- Języki wysokopoziomowe dzielimy na dwie grupy:
- Interpretowalne
- Kompilowalne.

- **Języki interpretowalne** nie wymagają kompilacji – HTML, PHP, JavaScript, *Python*.

- **Języki kompilowalne** wymagają procesu kompilacji kodu źródłowego do postaci kodu maszynowego, aby program mógł być wykonywany – Pascal, C, C++, Java.

Charakterystyka języka Python

- Python jest interpretowalnym językiem programowania wysokiego poziomu stworzonym przez holenderskiego programistę Guido van Rossuma w 1990 roku.
- Nazwa języka wywodzi się z programu telewizyjnego BBC „Latający cyrk Monty Pythona”
- Obecnie Python rozwijany jest jako projekt Open Source, zarządzany przez Python Software Foundation, będącą organizacją non-profit
- Język oferuje możliwości programowania obiektowego strukturalnego i funkcyjnego
- Został użyty do budowy takich serwisów jak Google i Youtube

Jak Python wykonuje programy?

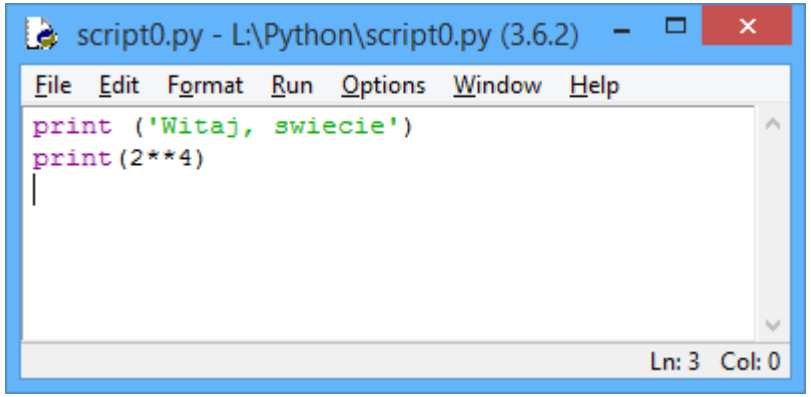
- Python to pakiet oprogramowania zwany interpreterem. Interpreter to rodzaj programu, który wykonuje inne programy. Kiedy pisze się kod w Pythonie, interpreter tego języka odczytuje program i wykonuje jego instrukcje. W rezultacie interpreter jest warstwą logiki oprogramowania znajdującą się pomiędzy kodem a urządzeniami naszego komputera
- Bez względu na formę kod w Pythonie musi zawsze być wykonywany przez interpreter. A żeby było to możliwe, konieczne jest zainstalowanie interpretera Pythona
- Szczegóły instalacji Pythona będą różne na różnych platformach. Dla systemu Windows wystarczy uruchomić plik instalacyjny pobrany np. z *www.python.org*

Wykonywanie programu

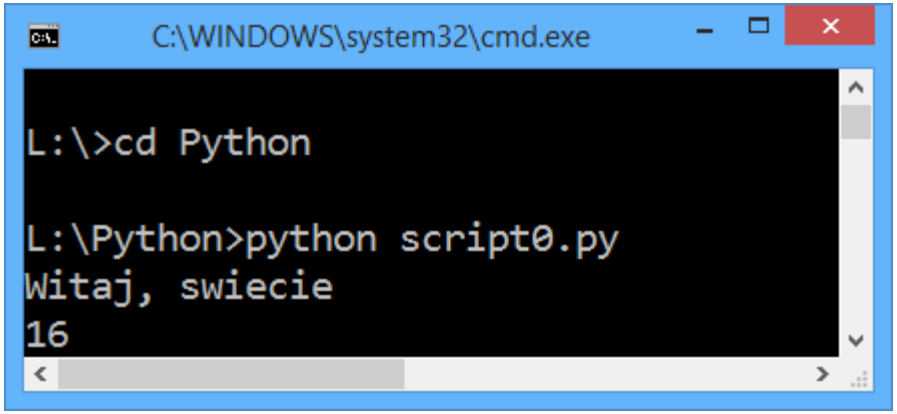
- W najprostszej formie program w Pythonie jest zwykłym plikiem tekstowym zawierającym instrukcje tego języka
- Taki plik można utworzyć w dowolnym edytorze tekstu
- Zgodnie z konwencją pliki Pythona otrzymują nazwy z rozszerzeniem *.py*
- Po wpisaniu instrukcji do pliku tekstowego należy ten plik wykonać – co oznacza wykonanie instrukcji od góry do dołu, jedna po drugiej za pomocą interpretera
- Pliki programów Pythona można wykonywać wykorzystując wiersz poleceń, klikając dwukrotnie ich ikony, uruchamiając je w zintegrowanym środowisku programistycznym IDE a także jako samodzielne aplikacje

Wykonywanie programu

Stworzymy plik *script0.py*



Wykonanie skryptu z linii poleceń

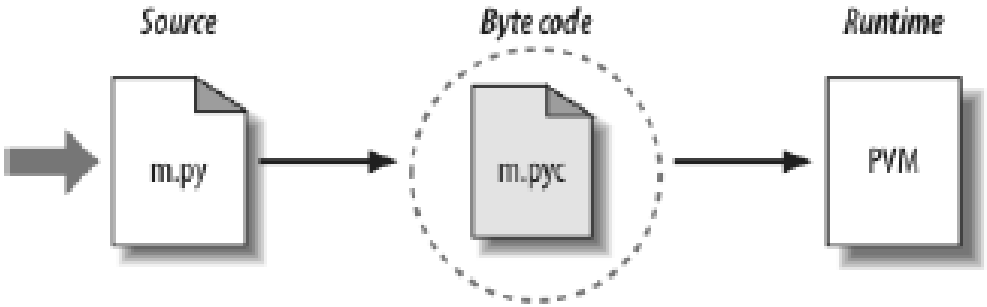


Wykonywanie programu

- Kiedy nakazujemy Pythonowi uruchomić skrypt, zanim kod zacznie się wykonywać, Python przeprowadza kilka kroków. Kod ten jest najpierw kompilowany na tak zwany *kod bajtowy* a następnie przesyłany do czegoś o nazwie *„maszyna wirtualna”*
- Kompilacja kodu źródłowego do kodu bajtowego to krok tłumaczenia kodu na inny format, a kod bajtowy jest niskopoziomową, niezależną od platformy reprezentacją kodu źródłowego
- Proces ten jest prawie całkowicie ukryty, efektem jest pojawienie się w pewnych przypadkach plików z rozszerzeniem *.pyc (.pyc* oznacza skompilowane źródło *.py)*

Wykonywanie programu

- Po skompilowaniu programu do kodu bajtowego (lub załadowaniu kodu bajtowego z istniejących plików *.pyc*) jest on przesyłany do wykonania do tzw. *maszyny wirtualnej (PVM)*
- Maszyna wirtualna jest silnikiem wykonawczym (*runtime engine*) odpowiedzialnym za samo wykonanie skryptów. Z technicznego punktu widzenia jest ostatnim etapem interpretera Pythona



Sposoby wykonywania programu

- Interaktywny wiersz poleceń
- Importowanie i przeładowywanie modułów
- Wywołanie *exec*
- Opcje menu interfejsu IDLE
- Kliknięcie ikony
- Zamrożone pliki binarne

Interaktywny wiersz poleceń można uruchomić na wiele sposobów – na przykład w IDE czy z konsoli systemowej

Najbardziej naturalnym rozpoczęciem sesji interpretera jest wpisanie polecenia *python* bez żadnych argumentów

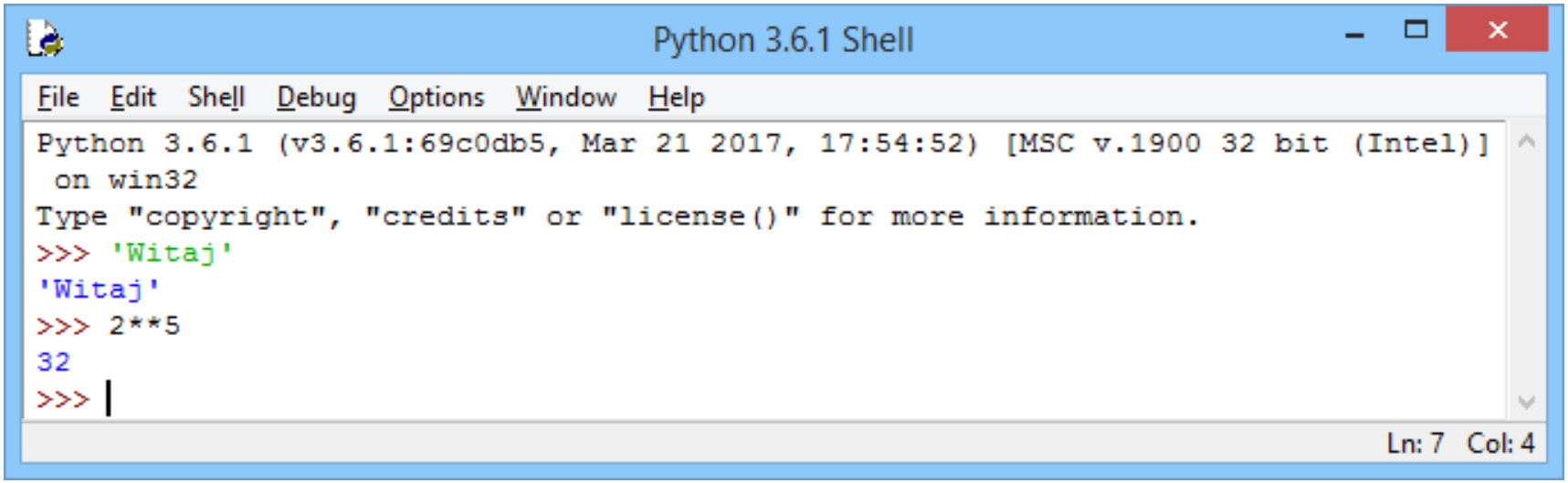
Systemowy wiersz poleceń

- Jeśli zmienna środowiskowa powłoki PATH nie zawiera katalogu instalacyjnego Pythona, słowo *python* należy zastąpić pełną ścieżką do pliku wykonywalnego Pythona
- Użytkownik zachęcany jest do wpisania nowej instrukcji za pomocą znaków zachęty >>>
- W sesji interaktywnej wynik wykonania kodu wyświetlany jest po naciśnięciu przycisku *Enter*
- By wyjść z sesji interaktywnej i powrócić do wiersza poleceń powłoki należy w systemie Windows użyć *Ctrl+Z*

```
L:\Python>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)]
 on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Witaj')
Witaj
>>> print(2**3)
8
```

Interaktywny wiersz poleceń - IDLE

- Sesja interaktywna automatycznie wyświetla wyniki, zazwyczaj nie trzeba w jawny sposób nakazywać ich wypisania za pomocą instrukcji *print*
- Sesję interaktywną możemy również uruchomić w środowisku IDLE
- Opuszczenie sesji w IDLE: Ctrl+D lub zamknięcie okna sesji

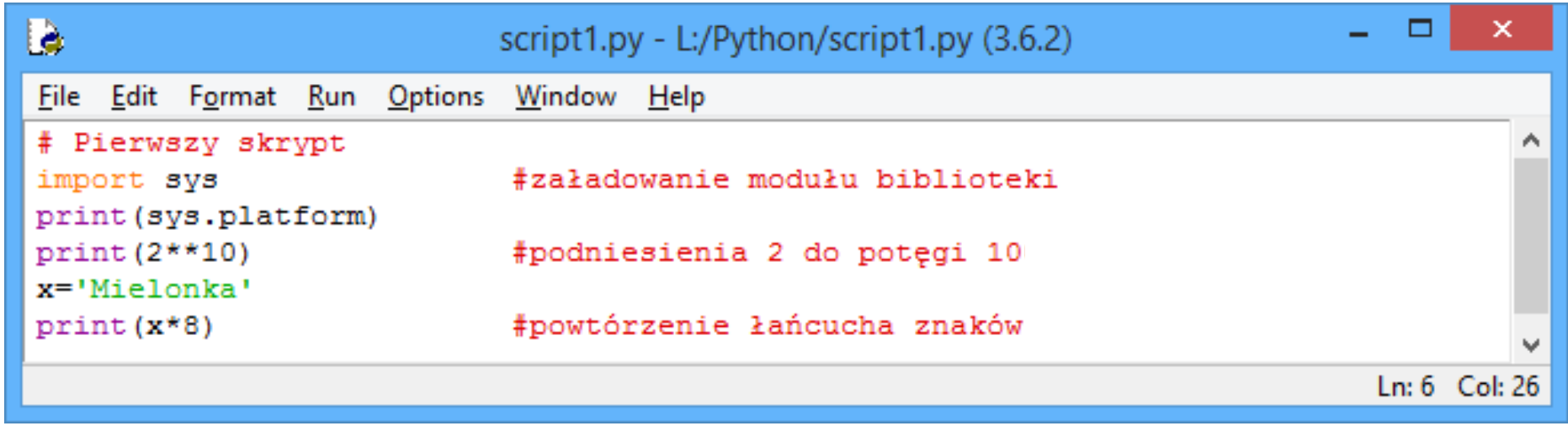


Skrypty

- Sesja interaktywna wykonuje kod i zwraca wyniki, jednak nie zapisuje kodu w pliku
- Aby zapisać program na stałe, trzeba wpisać kod do plików, które są zazwyczaj znane jako *moduły*
- Moduły to po prostu pliki tekstowe zawierające instrukcje Pythona. Po ich utworzeniu można nakazać interpreterowi wykonanie tych instrukcji dowolną liczbę razy i na różne sposoby – za pomocą wiersza poleceń, kliknięcia ikony czy z wykorzystaniem opcji Run interfejsu IDLE
- Bez względu na sposób uruchomienia Python wykonuje kod z góry do dołu za każdym razem kiedy wykonuje plik modułu

Skrypty

- Pliki modułów wykonywane w sposób bezpośredni są nazywane *skryptami* – co jest nieformalnym terminem oznaczającym zazwyczaj plik programu najwyższego poziomu
- Pojęcie modułu zarezerwowane jest dla plików importowanych z innych plików
- Pierwszy skrypt – należy uruchomić edytor tekstowy (Notatnik, IDLE lub jakikolwiek inny), wpisać instrukcje Pythona np.



I zapisać pod nazwą np. script1.py

Wykonanie skryptu za pomocą systemowego wiersza poleceń

- Po zapisaniu możemy zażądać wykonania go, podając pełną nazwę pliku jako pierwszy argument polecenia *python* w systemowym wierszu polecenia

```
L:\Python>python script1.py
win32
1024
MielonkaMielonkaMielonkaMielonkaMielonkaMielonkaMielonkaMielonka
```

- Możemy za pomocą składni powłoki przekierować wynik wykonania skryptu Pythona np. do pliku

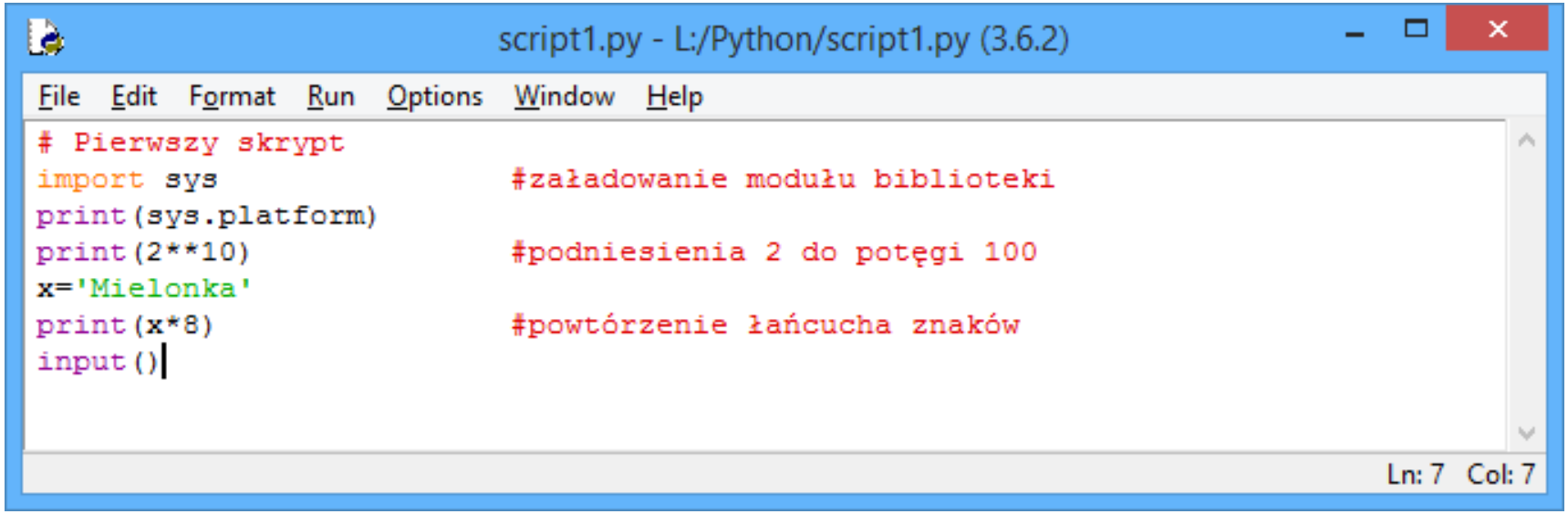
```
L:\Python>python script1.py > saveit.txt

L:\Python>
```

Kliknięcie ikony pliku

- W systemie Windows otwieranie plików za pomocą klikania jest łatwe – dzięki rejestrowi

- Aby zatrzymać okno z wynikiem działania programu można użyć instrukcji *input()*



Importowanie i przeładowywanie modułów

- Importowanie jako jeden ze sposobów uruchamiania programów
- Jeśli rozpoczniemy sesję interaktywną (z systemowego wiersza poleceń czy z IDLE, czy w inny sposób) możemy uruchomić wcześniej utworzony skrypt np. script1.py za pomocą instrukcji *import*.

```
>>> import script1
win32
1024
MielonkaMielonkaMielonkaMielonkaMielonkaMielonkaMielonkaMielonka
```

Ale musimy pamiętać o ścieżkach do skryptu

```
>>> import os
>>> os.getcwd()
'C:\\Users\\Kwiatek\\AppData\\Local\\Programs\\Python\\Python36-32'
>>> os.chdir('F:\Python')
```

Importowanie i przeładowywanie modułów

- Importowanie działa jednak tylko raz na sesję (a dokładniej mówiąc na proces), przy kolejnym importowaniu nic się nie dzieje
- Jeśli chcemy zmusić Pythona do wykonania skryptu jeszcze raz w tej samej sesji (bez jej zatrzymywania i ponownego uruchamiania) musimy zamiast tego wywołać funkcję *reload*
- *Należy ją jednak wcześniej zaimportować z modułu biblioteki standardowej imp (od wersji 3.x Pythona)*

```
>>> from imp import reload
>>> reload(script1)
win32
1024
MielonkaMielonkaMielonkaMielonkaMielonkaMielonkaMielonkaMielonka
```

Importowanie i przeładowywanie modułów

- Funkcja *reload* jest funkcją wywoływaną, natomiast import jest instrukcją
- Funkcja *reload* oczekuje podania w nawiasach okrągłych nazwy przeładowywanego modułu dlatego wcześniej moduł należy poprawnie zaimportować
- Możemy również zaimportować moduł imp,

>>>import imp w postaci : imp.reload(script1)

a następnie wywołać funkcję reload:

>>>imp.reload(script1)

Więcej o modułach

- Atrybuty

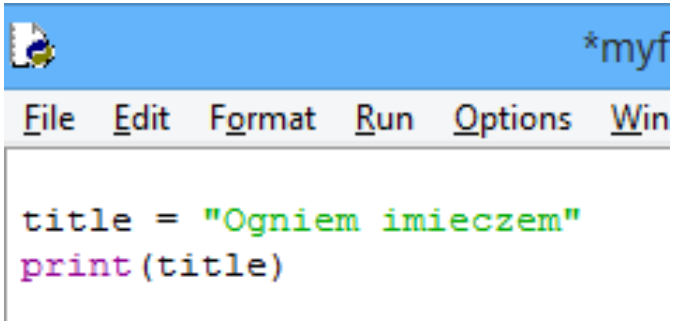
Importowanie i przeładowanie modułów stanowi naturalną opcję uruchamiania ponieważ polecenie *import* na samym końcu wykonuje pliki. W szerszym kontekście moduły pełnią rolę *biblioteki* narzędzi, czyli są pakietem nazw zmiennych tzw. *przestrzenią nazw (*ang. *namespace).* Nazwy w tym pakiecie to *atrybuty*.

Atrybut - jest nazwą zmiennej przywiązaną do określonego modułu

Plik importujący uzyskuje dostęp do wszystkich nazw (zmiennych) przypisanych na najwyższym poziomie importowanego pliku modułu

Więcej o modułach

- Nazwy z pliku modułu można pobrać z zewnątrz za pomocą instrukcji *import* i *from* a także dzięki wywołaniu funkcji *reload*.
- Przykład pliku modułu *myfile.py*



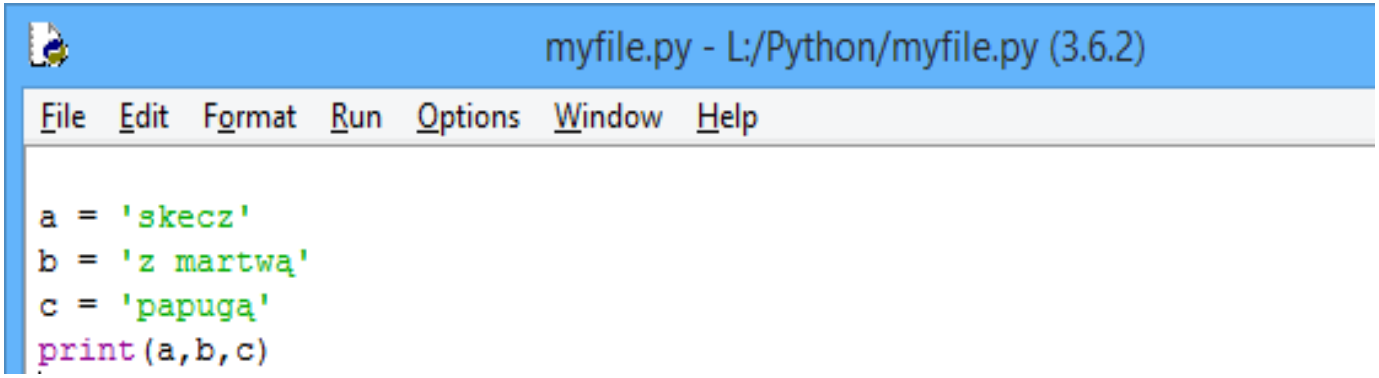

```
>>> import myfile
>>> print(myfile.title)
Ogniem i mieczem
>>>
```

Moduł zawiera jedną instrukcję przypisania. Kiedy zaimportujemy ten plik jego kod jest wykonywany. Instrukcja przypisania tworzy atrybut o nazwie *title*

- Składnia z kropką – w postaci *obiekt.atrybut* – pozwala na pobranie atrybutu do dowolnego obiektu.

Więcej o modułach

- W praktyce pliki modułów mogą definiować więcej niż jedną nazwę, która może być wykorzystana w tym pliku a także poza nim np.



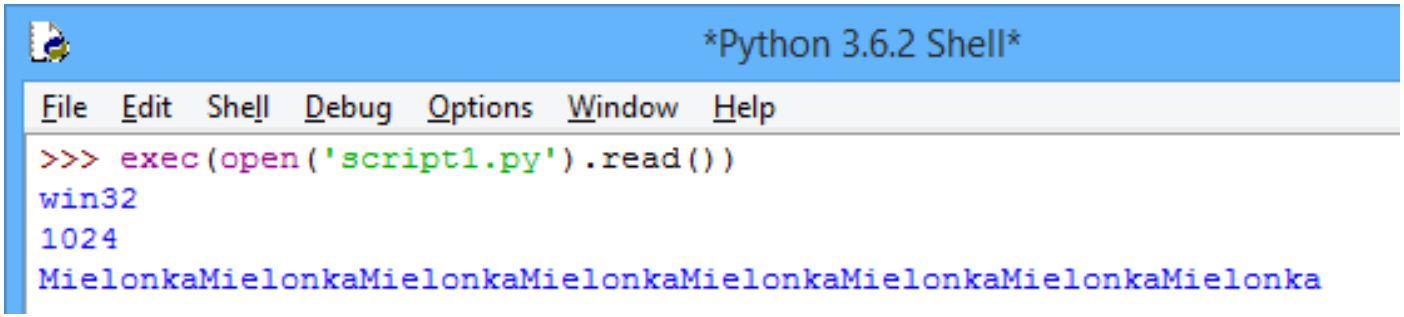
Wykorzystanie funkcji exec

- *Import* wykonuje plik raz na proces i nadaje mu osobną przestrzeń nazw modułu, tak by przypisania nie zmieniły zmiennych w bieżącym zakresie
- *From* kopiuje atrybuty modułu w taki sposób że stają się one zmiennymi w kodzie importującym

```
>>> from myfile import a,b,c
>>> a
'skecz'
>>> b
'z martwą'
>>> c
'papugą'
>>>
```

Wykorzystanie funkcji exec

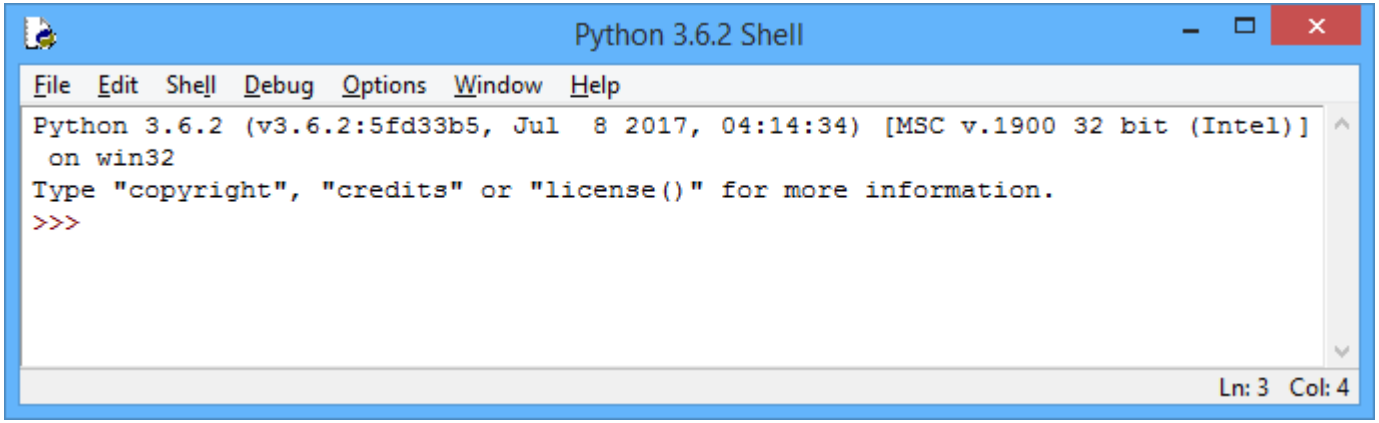
- Wywołania wbudowanej funkcji *exec* jest innym sposobem uruchamiania plików z sesji interaktywnej bez konieczności importowania i przeładowywania ich.



- Za każdym wywołaniem funkcji *exec* plik wykonywany jest od nowa, tak jakbyśmy wkleili jego kod w miejsce, gdzie wywołuje się *exec*

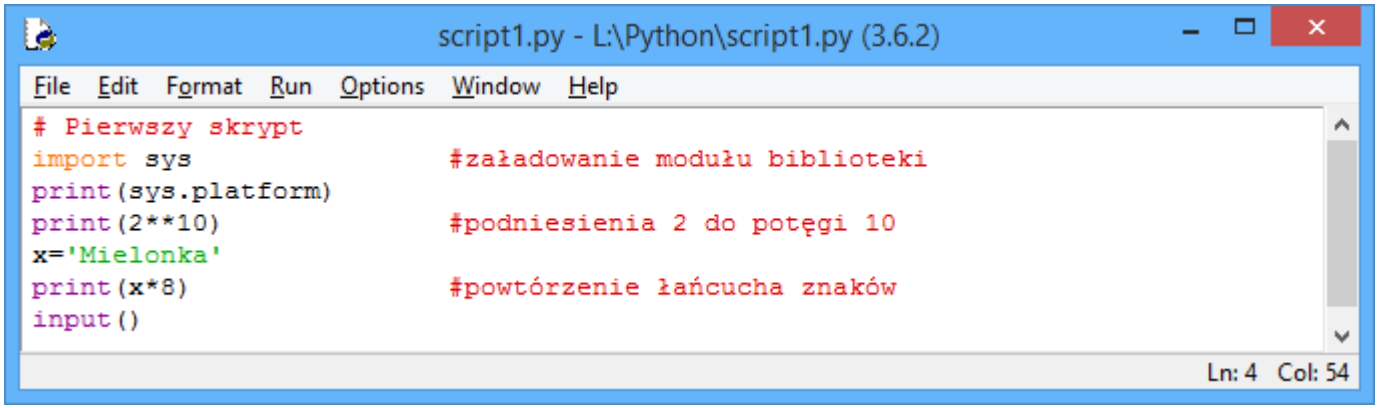
Interfejs użytkownika IDLE

- IDLE – to graficzny interfejs użytkownika (GUI) służący do programowania w Pythonie, który jest standardową częścią systemu Python. Zazwyczaj określa się go mianem *zintegrowanego środowiska programistycznego* (IDE)
- *IDLE* pozwala na edycję, wykonywanie i przeglądanie programów w Pythonie a także ich debugowanie.



Podstawy IDLE

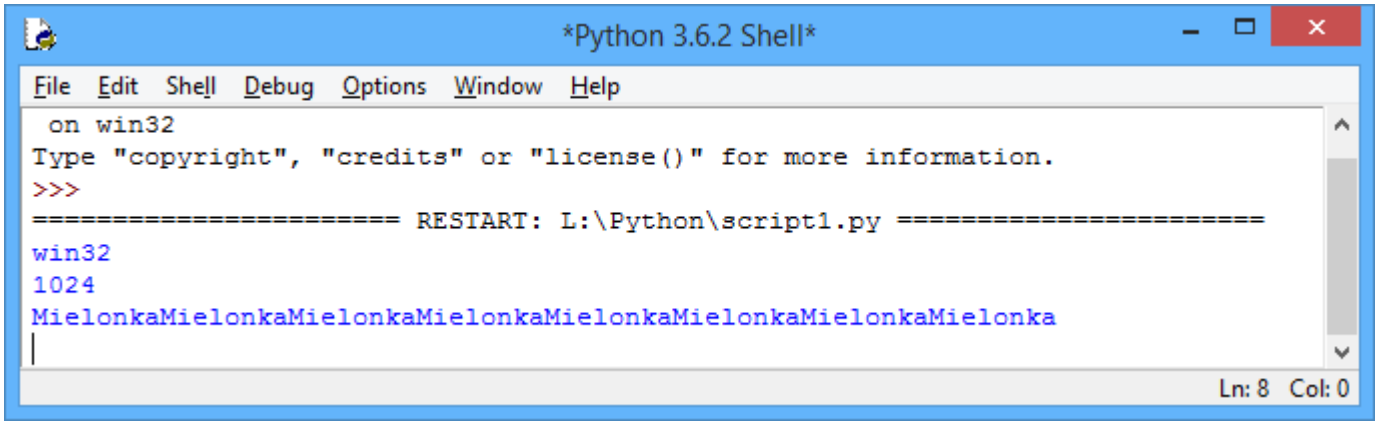
- By wykonać plik edytowany w IDLE, należy otworzyć okno edytora tekstowego pliku



- Następnie z menu *Run* tego okna wybrać opcję *Run Module* (lub użyć skrótu klawiszowego F5)

Podstawy IDLE

- Kiedy plik wykonujemy w ten sposób, dane wyjściowe oraz ewentualne komunikaty o błędach pokazują się w głównym oknie interaktywnym (oknie powłoki Pythona)



- Komunikat RESTART informuje, że proces wykonywania kodu rozpoczął się ponownie w celu wykonania edytowanego skryptu
- Skorzystanie z opcji Run Module w IDLE zawsze wykonuje najbardziej aktualną wersję pliku