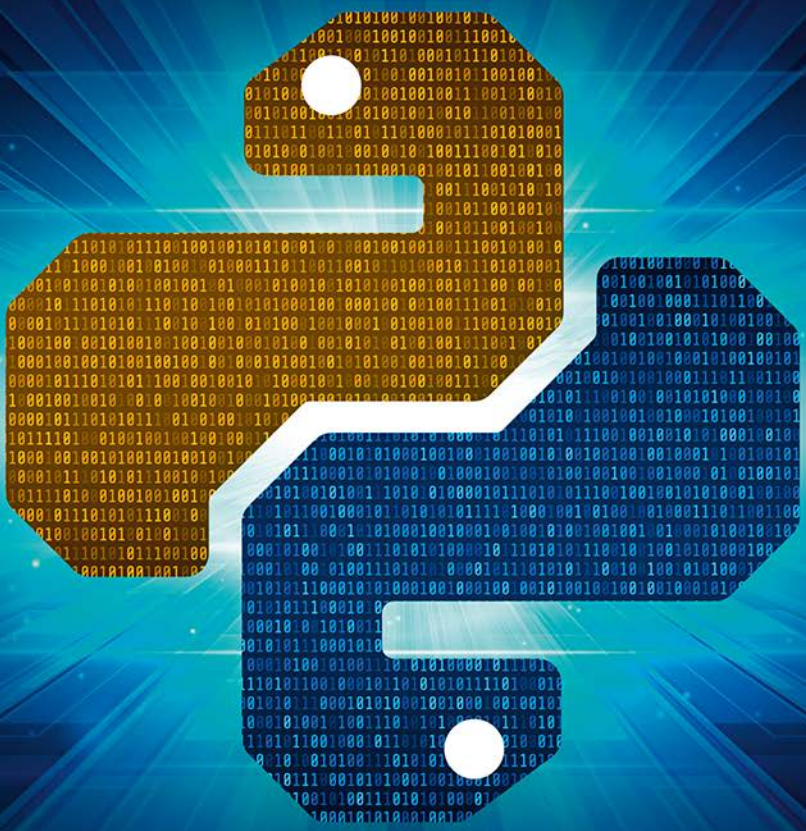
Gniewomir Sarbicki



Python

Kurs dla nauczycieli i studentów

Helion

Opieka redakcyjna: Ewelina Burska

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/pytkna
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-4566-9

Printed in Poland.

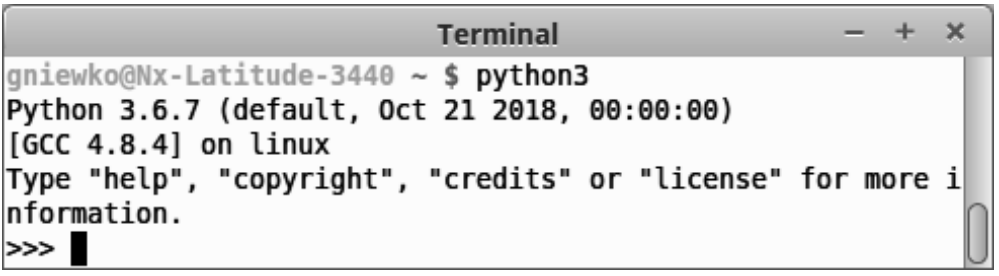
Spis treści

Rozdział 1.

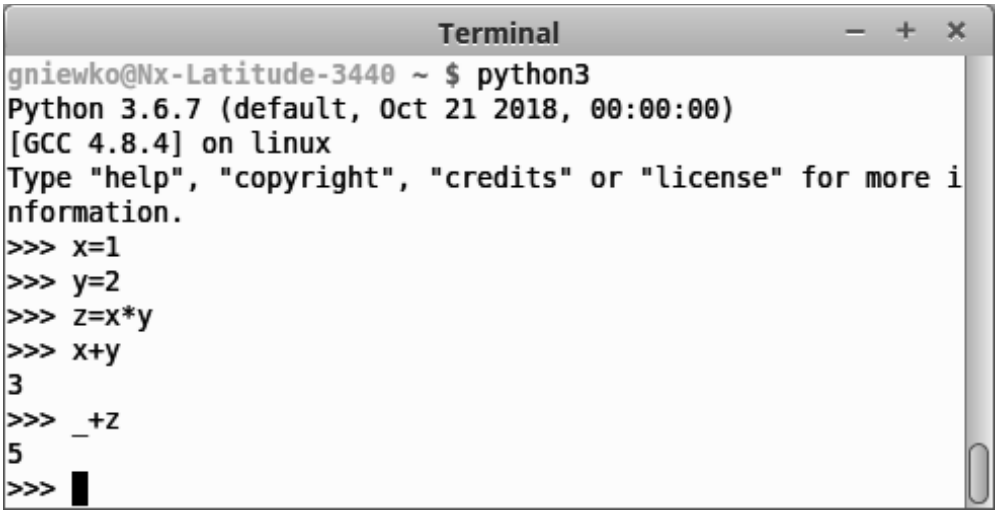
Wprowadzenie

1.1 Interaktywna powłoka, interpreter skryptów, edytory i środowiska

Wywołując polecenie `python3`, przechodzimy do interaktywnej powłoki Pythona.



W sesji powłoki pamiętane są zdefiniowane zmienne, możemy wykonywać na nich operacje. Jeżeli wyniku operacji nie przypiszemy, zostanie on wypisany. Żeby uzyskać dostęp do ostatniego wypisanego wyniku, używamy zmiennej `_` :



Powłoki interaktywne oferujące więcej funkcjonalności to m.in. `ipython`, `bpython`.

Używanie powłoki jest wygodne, gdy wykonujemy po sobie jednolinijkowe instrukcje. W przypadku bardziej złożonych programów umieszczamy je w skrypcie — pliku tekstowym, który następnie przekazujemy jako polecenie dla interpretera. Zapiszmy do pliku `pierwszy.py` następujący ciąg instrukcji:

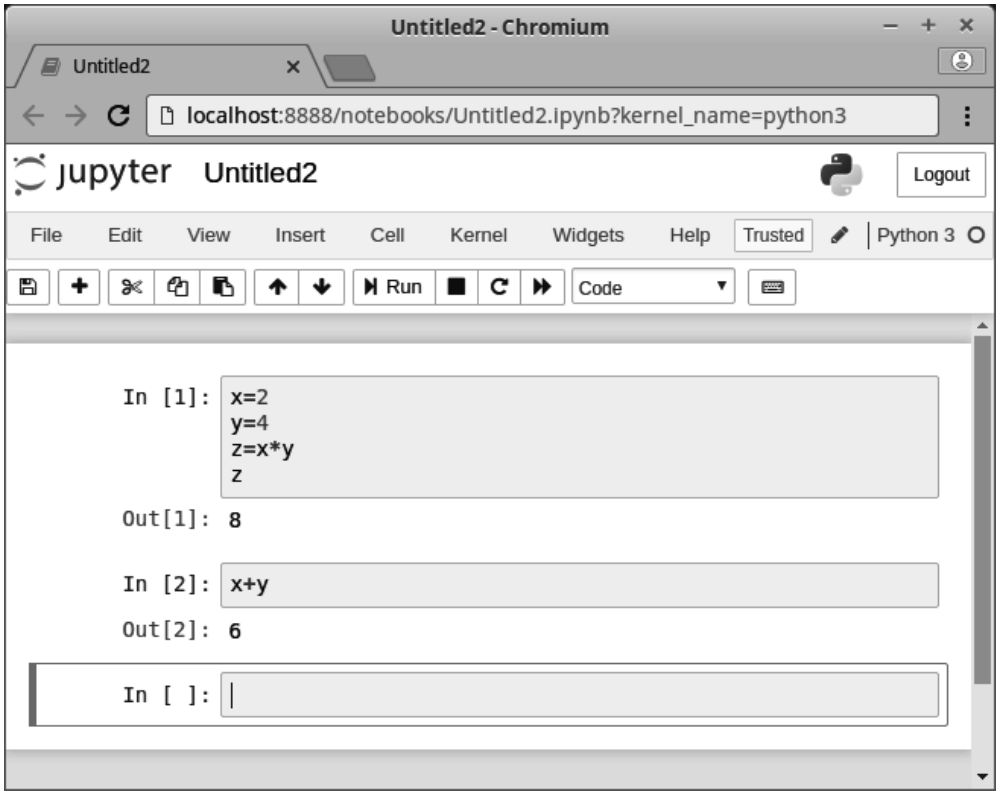
```
x=2
y=4
z=x*y
print(z)
```

i wywołajmy go poleceniem `python3 pierwszy.py`. Inaczej niż podczas pracy w trybie powłoki teraz, żeby wyprowadzić wynik operacji na ekran, trzeba użyć funkcji wbudowanej `print`[1].

Do tworzenia skryptów Pythona można użyć zwykłego edytora tekstu lub narzędzi bardziej zaawansowanych — środowisk programistycznych, które pozwalają edytować kod, uruchamiać go, korzystać z podpowiedzi, wstępnie analizują kod pod kątem błędów i pozwalają zarządzać projektami złożonymi z wielu plików. Najprostszym takim środowiskiem, dającym już wiele funkcjonalności, jest `geany`, a najbardziej zaawansowanym — `pycharm`. Środowisko `geany` jest wystarczająco funkcjonalne dla programów z tej książki.

[1] W Pythonie 2 poprawną składnią będzie `print z`.

Tryb pracy, który łączy zalety pisania wielolinijkowych skryptów i interaktywnej powłoki, oferuje środowisko `jupyter notebook`. Opiera się ono na koncepcji wielolinijkowych komórek — wpisujemy do komórki wielolinijkowy kod, wywołujemy całą komórkę i przechodzimy do następnej.



1.2 Typy liczbowe

Można zauważyć, że w powyższych przykładach nie definiowaliśmy typu zmiennej, jak należałoby zrobić np. w C. W odróżnieniu od C, `x` i `y` nie są obszarami w pamięci, ale nazwami, które wskazują na obiekty w pamięci. O typie obiektu informuje polecenie `type`. Wykonajmy w powłoce następujące instrukcje:

```
>>> x=2
>>> type(x)
<class 'int'>
>>> x=2.0
>>> type(x)
<class 'float'>
```

Stworzony został nowy obiekt `2.0` typu `float`. Zmienna `x` przestała wskazywać na obiekt `2`, a zaczęła na obiekt `2.0` i w ten sposób zmieniła swój typ. Obiektowi `2` zmniejszyła się liczba dowiązań o jeden i ponieważ żadna zmienna już na niego nie wskazywała, został on usunięty z pamięci. Właściwość języka polegająca na tym, że zmienna może wskazywać na obiekty różnego typu, nazywamy *słabym typowaniem*.

Na obiektach typów liczbowych możemy wykonywać następujące operacje liczbowe: `+`, `-`, `*`, `/`, `//` (dzielenie całkowitoliczbowe), `%` (reszta z dzielenia), `**` (potęgowanie)[2]. W Pythonie 3 dzielenie liczb całkowitych prowadzi w ogólności do liczby wymiernej: `1/2 = 0.5`. Dzielenie całkowitoliczbowe realizujemy przez `//`[3]. Wszystkie te działania mają odpowiadające im operatory przypisania:

```
a += b   ⇔  a = a + b
a -= b   ⇔  a = a - b
a *= b   ⇔  a = a * b
a /= b   ⇔  a = a / b
a //= b  ⇔  a = a // b
a %= b   ⇔  a = a % b
a **= b  ⇔  a = a ** b
```

Zmienne typu `int` są liczbami całkowitymi o nieograniczonym zakresie (ich rozmiar w bajtach ogranicza tylko ilość dostępnej pamięci operacyjnej)[4].

Zmienne typu `float` odpowiadają typowi `double` w `C` i są zapisywane na 64 bitach (1 bit znaku, 11 bitów wykładnika, 52 bity mantysy).

Ostatnim typem liczbowym jest typ zespolony `complex`, przechowujący część rzeczywistą i urojoną jako zmienne typu `float`:

```
>>> type(1.0+0.2j)
<class 'complex'>
```

[2] Potęgowanie realizuje również funkcja wbudowana `pow`, która oprócz argumentów *podstawa*, *wykładnik* może przyjmować trzeci argument, *m*, informujący o tym, że potęgowanie wykonujemy modulo *m*. Gdy potęgujemy modulo, `pow(x,y,z)` jest bardziej wydajne niż `x**y % z`.

[3] W Pythonie 2 dzielenie liczb całkowitych jest dzieleniem całkowitoliczbowym. Żeby wymusić wynik wymierny, należy wykonać je jako `1./2` (jeden z argumentów musi być liczbą wymierną).

[4] W Pythonie 2 zmienne typu `int` są zapisywane na 8 bajtach i przyjmują wartości od -2^{63} do $2^{63}-1$. W przypadku przekroczenia tego zakresu wynik przyjmuje typ `long`, odpowiadając typowi `int` w Pythonie 3. Typ `long` poznajemy po literze `L` na końcu zapisu liczby:

```
>>> 2**64
18446744073709551616L
>>> type(_)
<type 'long'>
```

Zmienna typu `complex` ma atrybuty: `real`, `imag`, oznaczające jej część rzeczywistą i część urojoną. Wywołanie `z.conjugate()` zwraca liczbę sprzężoną do `z`.

Funkcja wbudowana `abs` zwraca wartość bezwzględną argumentu (lub jego moduł, gdy jest zespolony).

Operatory rzutowania jednego typu na drugi nazywają się tak jak typ, na który rzutują: `int`, `float`, `complex`.

Ćwiczenie 1. *Zapisz wzór na rozwiązanie równania kwadratowego o współczynnikach zapisanych w zmiennych* `a`, `b`, `c`. *Wzór powinien działać dla dowolnych typów danych (całkowitych, wymiernych, zespolonych).*

1.3 Typy sekwencyjne

Jest 7 typów sekwencyjnych w Pythonie 3, z których najważniejsze to:

- łańcuch znaków (`str`)

  ```
  >>> s='żółw'
  ```

- łańcuch bajtów (`bytes`)

  ```
  >>> b=b'zolw'
  ```

- lista (`list`)

  ```
  >>> l=[1,2,3]
  ```

- krotka (`tuple`)

  ```
  >>> k=(1,2,3)
  ```

Wspólne operacje dla wszystkich typów sekwencyjnych to:

- Indeksowanie: `s[0],b[1],l[-1],k[-2]`. Wartości ujemne oznaczają indeksy liczone od końca (`-1` to ostatni indeks).
- Branie wycinków: `s[x:y]` lub `s[x:y:z]`, gdzie `x` oznacza początek wycinka, `y` oznacza koniec wycinka. Pominięcie wartości `x` oznacza od początku, pominięcie wartości `y` oznacza do końca. `z` oznacza, co który element, wartość ujemna oznacza odliczanie od końca.

- Mnożenie przez liczbę całkowitą dodatnią — powtarzanie: `2*[1,2]=[1,2,1,2]`.
- Dodawanie — oznacza konkatenację: `'ab'+'cd'='abcd'`.
- Operator członkostwa `in`:

```
>>> 'a' in ('a','b','c')
True
>>> 'a' not in ('a','b','c')
False
```

- Długość: `len(s)`.
- Elementy minimalny i maksymalny: `min(u)`, `max(u)`.
- Wyszukiwanie pierwszego indeksu:

```
>>> 'Ala␣ma␣kota'.index('a')
2
```

- Zliczanie elementów:

```
>>> 'Ala␣ma␣kota'.count('a')
3
```

Operatorami rzutowania na typy sekwencyjne są: `str`, `list`, `tuple`. Na typ `bytes` nie możemy rzutować — przekształcając ciąg znaków na ciąg bajtów, musimy podać kodowanie, tak samo, gdy chcemy rzutować w drugą stronę, czyli zdekodować ciąg bajtów do ciągu znaków. Do konwersji ciągu bajtów na ciąg znaków służy metoda `decode`. Do konwersji ciągu znaków na ciąg bajtów służy metoda `encode`. W metodach tych za argument podajemy kodowanie, np. `'utf8'`:

```
>>> b='żółw'.encode('utf8')
>>> type(b)
<class 'bytes'>
>>> b
b'\xc5\xbc\xc3\xb3\xc5\x82w'
>>> s=b.decode('utf8')
>>> type(s)
<class 'str'>
>>> s
'żółw'
```

Skorowidz


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA Helion

Python

Kurs dla nauczycieli i studentów

Weź Pythona na uczelnię!

Python to jeden z tych języków programowania, bez których trudno się dziś obejść. Dzięki słynnej elastyczności oraz rozbudowanemu zestawowi narzędzi i bibliotek można wykorzystywać go w zróżnicowanych projektach i na najróżniejszych platformach, a łatwość opanowania podstaw zachęca do nauki. Sprawdź sam, jak wygodnie i prosto da się w nim pisać własny kod, opracowywać aplikacje desktopowe, projektować strony WWW czy przeprowadzać obliczenia numeryczne.

Książka *Python. Kurs dla nauczycieli i studentów* powstała z myślą o wszystkich, którzy chcą opanować podstawy Pythona i praktycznie wykorzystywać go na co dzień. Kurs dzieli się na dwie części: pierwsza zawiera wprowadzenie do języka, pozwalające ruszyć z własnymi projektami; druga zaś wprowadza ważne, bardziej rozbudowane konstrukcje językowe. Niezwykle istotnym atutem tego podręcznika są ćwiczenia wraz z rozwiązaniami, opracowane na podstawie wieloletnich doświadczeń autora. Można tu znaleźć zarówno materiały do samodzielnej nauki, jak i inspiracje do prowadzenia własnych zajęć z uczniami czy ze studentami. Jeśli chcesz zacząć aktywnie korzystać z niesamowitych możliwości Pythona, ta publikacja jest dla Ciebie!

- **Podstawowe elementy, operacje i funkcje**
- **Programowanie obiektowe i graficzny interfejs użytkownika**
- **Wielowątkowość i komunikacja sieciowa**
- **Obsługa baz danych i współpraca z serwerem Apache**
- **Obliczenia numeryczne**
- **Iteratory, generatory, koprocedury**
- **Współbieżność wykorzystująca podprocesy**

Moc Pythona w Twoich rękach!

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

ISBN 978-83-283-4566-9

9 788328 345669

Cena: 59,00 zł