

Wprowadzenie do języka Python

Wykład 1

Sprawy organizacyjne

Sprawy organizacyjne

- Sylabus jest dostępny w systemie USOS.
- Regulamin zajęć dostępny jest na stronie prowadzącego zajęcia <http://wmii.uwm.edu.pl/~piojas/>.
- Forma zaliczenia: egzamin.
- Wykład - 15 godzin.
- <https://github.com/pjastr/WprowadzenieDoJezykaPython2023>

Wymagania wstępne

- Znajomość podstawowych konstrukcji programistycznych.

Ewentualne braki należy opanować w samodzielny zakresie.

W razie problemów zapraszam na konsultacje.

Wstęp do języka Python

Język Python

- Poprawna wymowa: pajton.
- Język Python stworzył we wczesnych latach 90. Guido van Rossum – jako następcę języka ABC.
- 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.

Przełomowy rok - 2008

- 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 np. w ArcGis Desktop <https://support.esri.com/en/technical-article/000013224>

Podstawowe różnice między 2.x a 3.x

- funkcja `print`

Python 2:

```
1 print 'Hello, World!'
2 print('Hello, World!')
3 print "text", ; print 'print more text on the same line'
```

Python 3

```
1 print('Hello, World!')
2 print("some text,", end='')
3 print(' print more text on the same line')
```


Dzielenie zmiennych typu `int`

Python 2:

```
1 print '3 / 2 =', 3 / 2
2 print '3 // 2 =', 3 // 2
3 print '3 / 2.0 =', 3 / 2.0
4 print '3 // 2.0 =', 3 // 2.0
```

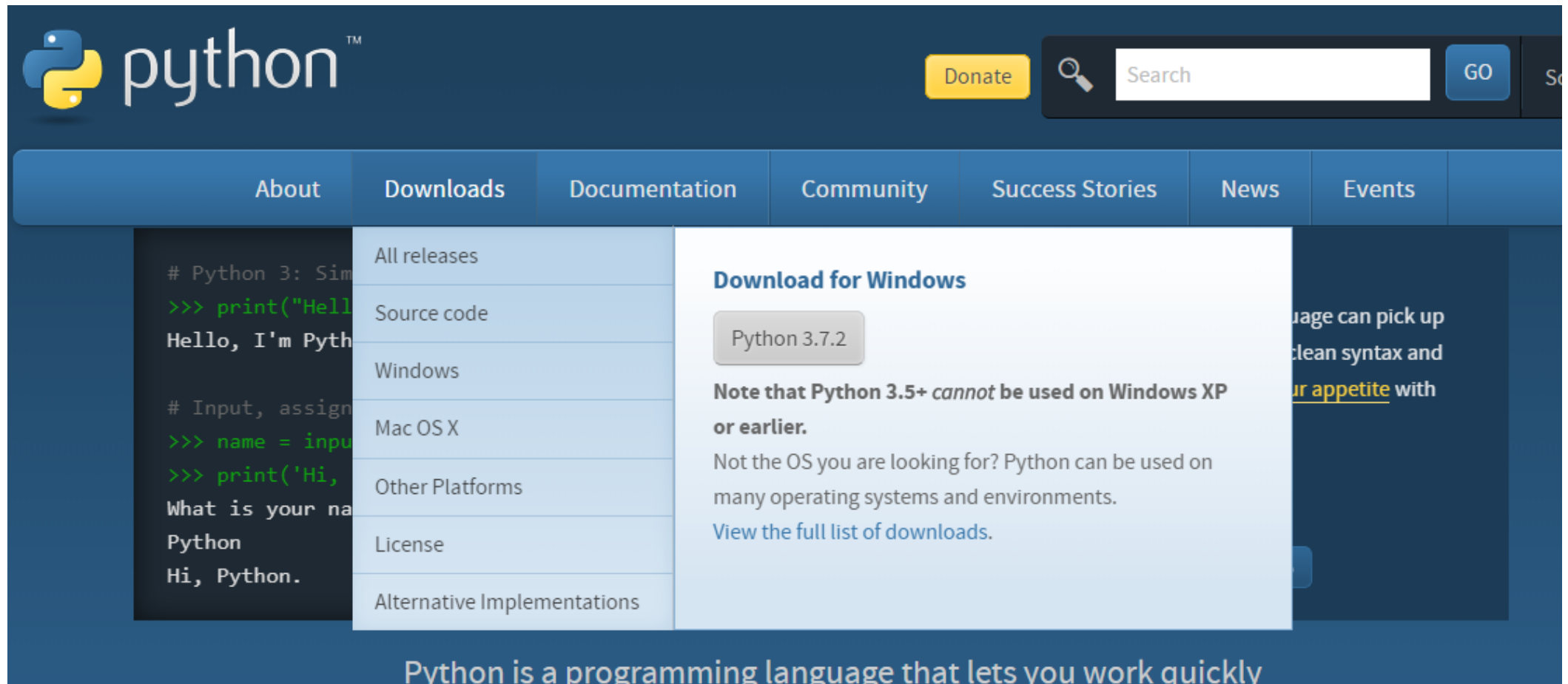
Python 3

```
1 print('3 / 2 =', 3 / 2)
2 print('3 // 2 =', 3 // 2)
3 print('3 / 2.0 =', 3 / 2.0)
4 print('3 // 2.0 =', 3 // 2.0)
```

Warto doczytać np. [tutaj](#).

Instalacja - Windows

- <https://python.org/>



The screenshot shows the Python.org website with the 'Downloads' menu open. The menu options are: All releases, Source code, Windows, Mac OS X, Other Platforms, License, and Alternative Implementations. The 'Download for Windows' section is highlighted, showing a button for 'Python 3.7.2'. A note states: 'Note that Python 3.5+ cannot be used on Windows XP or earlier.' Below this, it says: 'Not the OS you are looking for? Python can be used on many operating systems and environments. View the full list of downloads.'

python™

Donate

Search GO

About Downloads Documentation Community Success Stories News Events

```
# Python 3: Simple
>>> print("Hello, I'm Python")
Hello, I'm Python

# Input, assignment
>>> name = input("What is your name? ")
>>> print('Hi, Python')
What is your name? Python
Hi, Python.
```

Download for Windows

Python 3.7.2

Note that Python 3.5+ cannot be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments. [View the full list of downloads.](#)

Python is a programming language that lets you work quickly

Linux

Sprawdzenie wersji na Ubuntu 18.04:

```
piotrekwd@piotrekwd-VirtualBox:~$ python3
Python 3.6.5 (default, Apr 1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

Ubuntu

Ręczna instalacja:

```
1 sudo apt install python3
```

Wybór IDE do Pythona

- IDLE (domyślny)
- PyCharm <https://www.jetbrains.com/pycharm/> (na ćw. i wykład)
- 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>
- <https://colab.research.google.com/>
- <https://datalore.jetbrains.com/>
- i wiele innych...

Styl PEP8

- wymowa: pi-i-pi-ejt
- standaryzacja kodu używana m.in. przy rozwijaniu nowych funkcjonalności
- używanie daje lepszą organizację i czytelność kod
- pełna wersja <https://www.python.org/dev/peps/pep-0008/>

Znaki odstępu:

- we wcięciach stosujemy spacje (a nie tabulatory)
- każdy poziom wcięcia powinien składać się z 4 spacji
- wiersz powinien składać się z maksymalnie 79 znaków

Puste linie:

- dwie linie między funkcjami najwyższego poziomu i między klasami.
- pojedyncza linia między funkcjami w klasie

Kodowanie:

- dla Pythona 3 sugerowane i domyślne to UTF-8.

Stringi:

- można używać pojedynczych apostrofów jak i podwójnych cudzysłówów
- ważne, aby stosować wybraną notację konsekwentnie
- jedyny wyjątek to gdy wewnątrz stringu chcemy użyć cudzysłów np.

```
1 print('Oglądam film "Player One"')
```


Jak sprawdzać styl PEP8?

- Pycharm - Code - Inspect Code
- moduł flake8

Różnice - Editor/Code Style/Python

Cechy języka Python

- Python wspiera różne paradygmaty programowania: obiektowy, imperatywny oraz funkcyjny.
- Posiada w pełni dynamiczny system typów i automatyczne zarządzanie pamięcią (garbage collector).
- Często używany jako język skryptowy. Interpretery Pythona są dostępne na wiele systemów operacyjnych. Różne implementacje Pythona: CPython (język C), IronPython (platforma .NET), Jython (Java), PyPy (Python).
- Prosta i czytelna składnia ułatwiająca utrzymywanie, używanie i rozumienie kodu.

Zen

```
1 import this
```

The Zen of Python, by Tim Peters

```
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
...
```

PL - <https://pl.python.org/forum/index.php?topic=392.msg1844#msg1844>

Zmienna

- najprościej: przechowuje pewną wartość:

```
1 x = 5  
2 y = "John"
```

Wbudowane typy danych

- numeryczne (liczbowe): int, float, complex
- tekstowe: str
- sekwencji: list, tuple
- odwzorowania (mapping type): dict
- zestawów (set types): set, frozenset
- logiczne: bool
- binarne: bytes, bytearray

Int - typ całkowity

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

```
1 print(123123123123123123123123123123123123123123 + 1)
```

```
123123123123123123123123123123123123123124
```

Jaki system liczbowy?

Domyślnie dziesiętny. Więcej za kilka zajęć przy algorytmach liczbowych.

```
1 print(101)
2 print(101)
3 print(0x101) # zero-x
4 print(0o101) # zero-litera o
5 print(0b101) # zero-b
6 print(0X101) # zero-x
7 print(0O101) # zero-litera o
8 print(0B101) # zero-b
```

```
101
101
257
65
5
257
65
5
```

Sprawdzenie typu

```
1 print(type(234))
```

```
<class 'int'>
```

`<class 'int'>` - wszystko jest obiektem

Float - typ zmiennoprzecinkowy

```
1 print(4.2)
2 print(4.)
3 print(.5)
4 print(.3e4)
5 print(3e4)
6 print(2e-2)
7 print(1.79e308)
8 print(1.8e308)
9 print(5e-324)
10 print(1e-325)
```

```
4.2
4.0
0.5
3000.0
30000.0
0.02
1.79e+308
inf
5e-324
0.0
```

Complex - typ zespolony (dot. liczb zespolonych)

```
1 print(2+3j)
2 print(2+5*1j)
```

(2+3j)

(2+5j)

Uwaga: wyrażenie

```
1 print(4+5*j)
```

skutkuje błędem w większości wypadków.

str - string, napisy, łańcuchy znaków

Obecnie odchodzi się od określenia “tablica znaków”

```
1 print("ABC")  
2 print('abc')
```

ABC

abc

bool - typ logiczny

```
1 print(True)
2 print(False)
```

True

False

Operator przypisania

= przypisuje prawą stronę do lewej (!), często mylony z operatorem logicznym równa się ==

```
1 x = 5
2 x = "Piotr"
3 a = 4.5
4 A = 56
5 x, y, z = "Orange", "Banana", "Cherry"
6 x = y = z = "Orange"
```

input - operacja wejścia

```
1 num = input ("Wprowadź liczbę :")
2 print(num)
3 name1 = input("Wprowadź imię : ")
4 print(name1)
```

- zawsze przyjmuje napis, w razie potrzeby trzeba zrzutować

```
1 x = str(num)
2 y = int(num)
3 z = float(num)
```

print - instrukcja wyjścia

```
1 print(4.2)
2 # print('Mój wiek to' + 36)
3 print('Mój wiek to', 36)
4 a = 36
5 print('Mój wiek to', a)
6 print('hello', 'world')
7 print('hello', 'world', sep='')
8 print('hello', 'world', sep='\n')
```

4.2

Mój wiek to 36

Mój wiek to 36

hello world

helloworld

hello

world

```
1 print(*objects, sep=' ', end='\n', file=sys.stdout,  
2      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**


```
1 print(1, 2, 3, 4)
2 print(1, 2, 3, 4, sep='*')
3 print(1, 2, 3, 4, sep='#', end='&')
```

```
1 2 3 4
1*2*3*4
1#2#3#4&
```

```
1 print('x', 'y', 'z', sep='', end='')  
2 print('a', 'b', 'c' , sep='', end='')
```

xyzabc

```
1 print('a', 'b', '\n', 'c')
```

a b

c

`\t` - przesunięcie do następnego “tab”=8 spacji

```
1 print('sdf', 3456, -2, sep='\t')
```

```
sdf 3456      -2
```

Formatowanie napisów będzie później.

Operacje arytmetyczne

Operator	Opis	Składnia
+	Dodawanie	$x + y$
-	Odejmowanie	$x - y$
*	Mnożenie	$x * y$
/	Dzielenie	x / y
//	Dzielenie całkowite	$x // y$
%	Dzielenie modulo	$x \% y$
**	Potęgowanie	$x ** y$

```
1 print(5+3)
2 print(4*5.2)
3 print(9-7)
4 print(25%7)
```

8

20.8

2

4

```
1 print(4/5)
2 print(4//5)
3 print(4/5.0)
4 print(4//5.0)
```

0.8

0

0.8

0.0

```
1 print(3**0)
2 print(0**0)
```

1

1

```
1 print(4/0)
```

Daje info: **ZeroDivisionError: division by zero.**

Przypisanie z operacją arytmetyczną

Lista zawiera wybrane operacje.

Inna nazwa to złożone operatory przypisania.

Operator	Zapis	Dłuższa wersja
<code>+=</code>	<code>x += 5</code>	<code>x = x + 5</code>
<code>-=</code>	<code>x -= 5</code>	<code>x = x - 5</code>
<code>*=</code>	<code>x *= 5</code>	<code>x = x * 5</code>
<code>/=</code>	<code>x /= 5</code>	<code>x = x / 5</code>
<code>%=</code>	<code>x %= 5</code>	<code>x = x % 5</code>
<code>//=</code>	<code>x //= 5</code>	<code>x = x // 5</code>
<code>**=</code>	<code>x **= 5</code>	<code>x = x ** 5</code>


```
1 a = 5
2 a += 1
3 print(a)
4 a **= 2
5 print(a)
```

6

36

Operatory porównania

Operator	Znaczenie	Przykład
>	Większe niż	$x > y$
<	Mniejsze niż	$x < y$
==	Równe	$x == y$
!=	Nie równa się	$x != y$
>=	Większe lub równe	$x >= y$
<=	Mniejsze lub równe	$x <= y$

Operatory logiczne

Operator	Znaczenie	Przykład
and	i	x and y
or	lub	x or y
not	negacja	not x

Operatory bitowe

Operator	Znaczenie	Przykład
&	i - logiczne	$x \& y$
	lub - logiczne	$x y$
^	albo - logiczne	$x \wedge y$
~	negacja - logiczne	$\sim x$
<<	przesunięcie w lewo	$x \ll y$
>>	przesunięcie w prawo	$x \gg y$

Operator &

```
1 print(4&5)
```

4

x	100	4
<hr/>		
y	101	5
<hr/>		
x&y	100	4

Operator |

```
1 print(4|5)
```

5

x	100	4
<hr/>		
y	101	5
<hr/>		
x y	101	5

Operator ^

```
1 print(4^5)
```

1

x	100	4
<hr/>		
y	101	5
<hr/>		
x^y	001	1

Operator ~

- równoważnie $-(x+1)$

```
1 print(~4)
```

-5

x	100	4
<hr/>		
~x	?	-5

Operator <<

$a \ll b$ - równoważnie $a * \text{pow}(2, b)$

```
1 print(3<<2)
```

12

x	0011	3
<hr/>		
x<<2	1100	12

Operator >>

$a \gg b$ - równoważnie $a // \text{pow}(2, b)$

```
1 print(13>>2)
```

3

x	1101	13
<hr/>		
x>>2	0011	3

Instrukcje warunkowe

Składnia

```
1 if <expr>:  
2     <statement>
```

else

```
1 if <expr>:  
2     <statement(s)>  
3 else:  
4     <statement(s)>
```

```
1 a = 5
2 if a > 0:
3     print("Liczba dodatnia")
4 else:
5     print("Liczna ujemna lub zero")
```

Liczba dodatnia

```
1 x = 0
2 y = 5
3 if x < y:
4     print('yes1')
5
6 if y < x:
7     print('yes2')
8
9 if x:
10    print('yes3')
11
12 if y:
13    print('yes4')
14
15 if x or y:
16    print('yes5')
17
18 if x and y:
19    print('yes6')
```

yes1
yes4
yes5

elif

```
1 if <expr>:  
2     <statement(s)>  
3 elif <expr>:  
4     <statement(s)>  
5 elif <expr>:  
6     <statement(s)>  
7     ...  
8 else:  
9     <statement(s)>
```

```
1 a = 5
2 if a > 0:
3     print("Liczba dodatnia")
4 elif a == 0:
5     print("Zero")
6 else:
7     print("Liczna ujemna")
```

Liczba dodatnia

Zagnieżdżone instrukcje warunkowe:

```
1 if <expr>:  
2     <statement(s)>  
3     if <expr>:  
4         <statement(s)>
```

```
1 i = 21
2 if i > 0:
3     print("liczba jest dodatnia")
4     if i % 2 == 0:
5         print("Liczba jest dodatkowo parzysta")
```

liczba jest dodatnia

Bibliografia

- <https://pl.wikipedia.org/wiki/Python>, dostęp online 20.02.2023.
- <https://bulldogjob.pl/news/264-java-php-ruby-jak-wlasciwie-wymawiac-nazwy-technologii>.
dostęp online 12.02.2019.
- https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html, dostęp online 14.02.2019.
- K. Ropiak, Wprowadzenie do języka Python,
<http://wmii.uwm.edu.pl/~kropiak/wd/Wprowadzenie%20do%20j%C4%99zyka%20Python.p>
dostęp online 14.02.2019.
- B. Slatkin, Efektywny Python. 59 sposobów na lepszy kod, Helion 2015.
- <https://www.python.org/dev/peps/pep-0008/>, dostęp online 20.02.2023.

Bibliografia - cd2

- <https://www.flynerd.pl/2017/05/python-4-typy-i-zmienne.html>, dostęp online 14.02.2019.
- <http://pytlearn.csd.auth.gr/p0-py/01/print.html>, dostęp online 15.02.2019.
- https://www.tutorialspoint.com/python3/python_lists.htm, dostęp online 17.02.2019.
- <https://realpython.com/python-data-types/>, dostęp online 5.01.2022
- https://www.w3schools.com/python/python_variables.asp, dostęp online 5.01.2022
- https://www.w3schools.com/python/python_variables_multiple.asp, dostęp online 5.01.2022

Bibliografia - cd3

- <https://realpython.com/python-print/>, dostęp online 5.01.2022
- <https://www.programiz.com/python-programming/operators>, dostęp online 5.01.2022
- <https://realpython.com/python-conditional-statements/>, dostęp online 5.01.2022
- <https://realpython.com/python-for-loop/>, dostęp online 5.01.2022
- <https://realpython.com/python-while-loop/>, dostęp online 5.01.2022

