Kurs rozszerzony języka Python

Wykład 1.

Marcin Młotkowski

2 października 2019

Plan wykładu

Plan wykładu

Wykładowca: Marcin Młotkowski

Termin wykładu: środa, 12:15–14:00, sala 25

Strona wykładu http://www.ii.uni.wroc.pl/~marcinm/dyd/python

Materiały pomocnicze

- www.python.org
- Dive into Python, Mark Pilgrim
- Expert Python Programming, Tarek Ziade
- The Hitchhiker Guide to Python, Kenneth Reitz, Tanya Schlusser
- Python 3 Object Oriented Programming, Dusty Phillips
- Python3 Patterns & Idioms Book, Bruce Eckel
- Programming Python, Mark Lutz

Pracownia

- Pierwsza część semestru (ok. 10 tyg.) — listy z krótkimi zadaniami programistycznymi
- Druga część semestru — większy projekt

Zaliczenie

Zdobycie przynajmniej połowy punktów.

Plan kursu

1. Język Python
   składnia, typy podstawowe, wbudowane struktury danych, obiekty, programowanie funkcjonalne
2. Standardowe biblioteki
   przetwarzanie tekstu, bazy danych, interfejsy graficzne, I/O, protokoły sieciowe, wątki, SciPy, NumPy
3. Zaawansowane zagadnienia
   testowanie i dokumentowanie, refleksje, współpraca z innymi językami

Plan wykładu

Początki języka Python

Lata 90 — CWI Amsterdam, Guido van Rossum

Stan obecny

Python Software Foundation (PSF)

Aktualna wersja (1.10.2019)

- 2.7.14
- 3.7.16

Aktualna wersja (1.10.2019)

- 2.7.14
- 3.7.16

Ranking popularności języków programownia TIOBE: 3 pozycja
`https://www.tiobe.com/tiobe-index/python/`

Dlaczego Python jest fajny

Realizacja różnych paradygmatów

- Paradygmat programowania strukturalnego
- Paradygmat programowania obiektowego
- Paradygmat programowania funkcjonalnego

Dlaczego Python jest fajny

Wbudowane typy:

Listy

```
vec = [1, 2, 3]
doubled_vec = [ 2*e for e in vec]
```

Słowniki

```
tel = { 'krzysiek' : 235711, 'ewa' : 246810 }
print tel['ewa']
```

Dlaczego Python jest fajny

Batteries included

- Biblioteki operacji we/wy
- Obsługa wyrażeń regularnych
- HTTP, HTML, XML
- Interfejsy okienkowe (pyGTK, wxPython, Tkinter)
- SciPy, NumPy
- ...

Inne cechy Pythona

Dynamiczny system typów

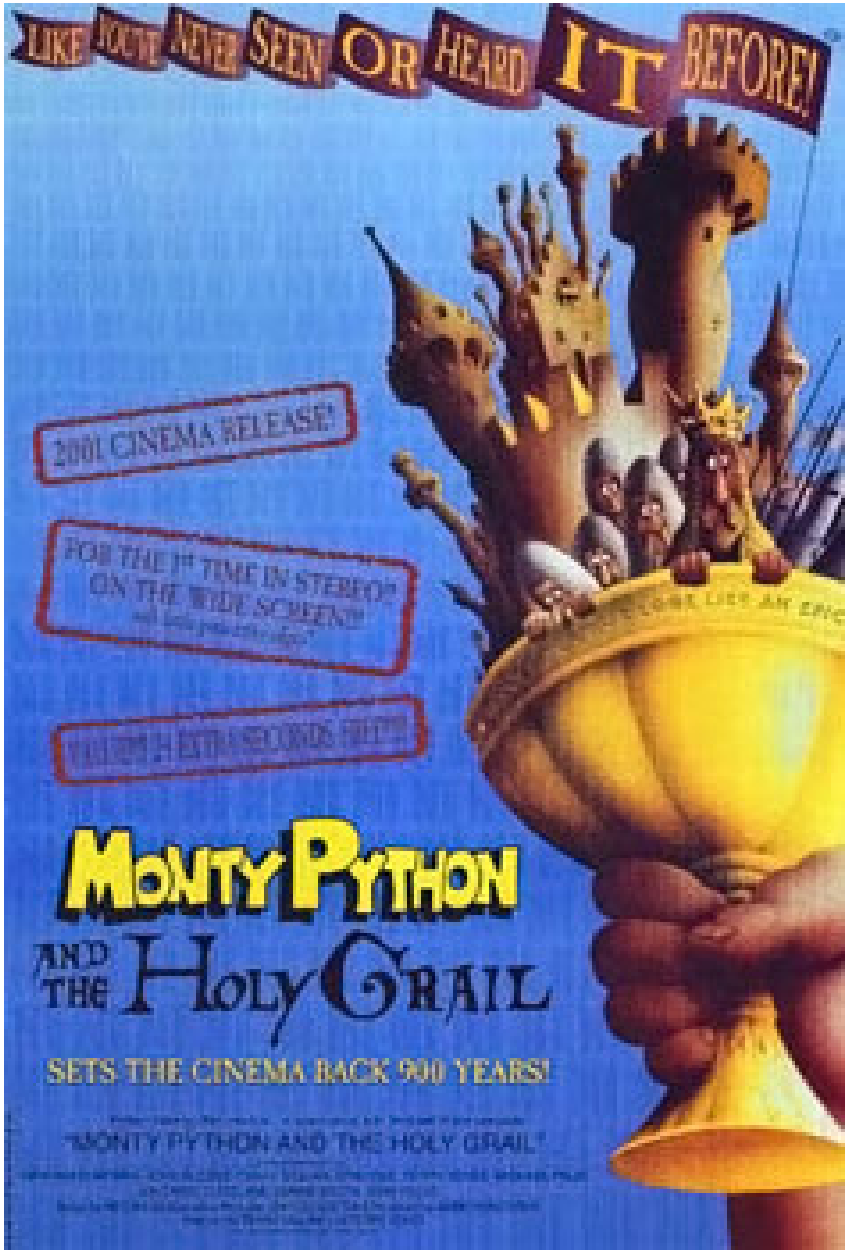
```
>>> 2 + "dwa"
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: unsupported operand type(s) for +: 'int'
and 'str'
>>>
```

Zastosowania

Projekty

- Narzędzia systemowe (RedHat), Google
- Django
- eksploracja danych

Skąd pochodzi nazwa?



Plan wykładu

Tryb interaktywny

```
$ python3
>>> 2+2
4
>>> [1,2,3][-1:]
[3]
Ctrl-d
$
```

Pierwsza pomoc w nagłej potrzebie

Tryb interaktywny

```
>>> type(3.1415)
<type 'float'>
>>> dir(float)
....
>>> dir(3.1415)
....
>>> float.__doc__
```

Tryb wsadowy

```
$ python3 plik.py
```

Co się dzieje

1. Kompilacja programu

Tryb wsadowy

```
$ python3 plik.py
```

Co się dzieje

1. Kompilacja programu
2. Czasem tworzy się plik *.pyc

Tryb wsadowy

```
$ python3 plik.py
```

Co się dzieje

1. Kompilacja programu
2. Czasem tworzy się plik *.pyc
3. Program jest wykonywany

Wskazówki

- Edytory z podświetleniem składni: vim, gedit, geany, emacs

Wskazówki

- Edytory z podświetleniem składni: vim, gedit, geany, emacs
- Narzędzia
  - idle
  - PythonCard/codeEditor
  - PyCharm
  - Visual Studio Code

Wskazówki

- Edytory z podświetleniem składni: vim, gedit, geany, emacs
- Narzędzia
  - idle
  - PythonCard/codeEditor
  - PyCharm
  - Visual Studio Code
- Jupyter

Wskazówki

- Edytory z podświetleniem składni: vim, gedit, geany, emacs
- Narzędzia
  - idle
  - PythonCard/codeEditor
  - PyCharm
  - Visual Studio Code
- Jupyter
- Pliki "wykonywalne" i polskie litery:

plik.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

Plan wykładu

Standardowe typy liczbowe

Typ **int**

- Stałe: 0x123, 0x123456789L, 0o123, 0o6789L
- Zakres $[-sys.maxint - 1, sys.maxint]$

Standardowe typy liczbowe

Typ **int**

- Stałe: 0x123, 0x123456789L, 0o123, 0o6789L
- Zakres $[-sys.maxint - 1, sys.maxint]$

Typ **float**

Stałe: 3.14, 10., .002, .271e1

Dodatkowe typy liczbowe

Typ **long**

10000000000000000000000000000000L

Typ **complex**

Stałe: $1 + 3j$, $x + 12j$, $complex(x, 0)$

Konwersja między typami

Użycie nazwy typu jako operatora rzutowania

float(1), long(1.0), int(3.14)

Konwersja podczas obliczeń

- $(1.0 * 4)//3 = 1.0$
- $(1.0 * 4)/3 = 1.33333333333333$
- $float(4)/3 = 4/float(3)$

Wyrażenia logiczne

Prawda

True, wartości niepuste

Fałsz

0, False, None, [], { }

Wyrażenia logiczne

Prawda

True, wartości niepuste

Fałsz

0, False, None, [], { }

Operatory

- and, or, not
- $==$, $!=$, $1 < y < 3$

Przykłady

- Listy: [12,3]
- Napisy: "abc", 'def', 'Zażółć żółtą jaźń'
- Krotki: (1, "jeden", (1, 2+3j, 0x4))
- Słowniki:

```
htmlColor = { 'turquoise' : (64,224,208),
              'NavyBlue' : NavyBlue }
```

- Zbiory

Przypisania

```
x = 123
```

Przypisania

```
x = 123
```

```
x = x if x > 0 else -x
```

Instrukcja warunkowa

```
if x > 0:
    print('dodatnia')
elif x < 0:
    print('ujemna')
else : print('zero')
```

Instrukcje pętli

Instrukcja while

```
a, b = 0, 1
while b < 10:
    print (b)
    a, b = b, a + b
```

Instrukcje pętli

Instrukcja for

```
a = [1,2,3,4]
for e in a:
    print (e)
print ("koniec")
```

Instrukcje pętli

"Prawdziwa" instrukcja for

```
suma = 0
for i in range(100):
    suma = suma + i
print ("suma=", suma)
```

Inne instrukcje

- Instrukcje break i continue
- Instrukcja pusta pass

```
while (True): pass
```

Procedury i funkcje

```
def funkcja(arg1, arg2=1, arg3=[3]):
    print (arg1, arg2, arg3)
    return 4

funkcja("jeden", 2)
print (funkcja(1, 2, 3))
```

Procedury i funkcje

```
def kwadrat(x): return x*x

print (kwadrat(10))

def funkcja(arg1, arg2=1, arg3=[3]):
    print (arg1, arg2, arg3)
    return 4

funkcja("jeden", 2)
print (funkcja(1, 2, 3))
```

Komentarze

```
def fun (arg):
    """To jest bardzo wazna funkcja
    uzywac z wielka ostoznoscia"""

    # koniec gdy argument pusty
    if arg == None: return
    return arg
```

Wejście/wyjście

Python 2.*

```
print ("Hello world")
x = input("Podaj x: ")
y = input("Podaj y: ")
print "x =", x, " y =", y
```

Wejście/wyjście

Python 2.*

```
print ("Hello world")
x = input("Podaj x: ")
y = input("Podaj y: ")
print "x =", x, " y =", y
```

Python 3.0

```
print("Hello world")
x = input("Podaj x: ")
y = input("Podaj y: ")
print("x =", x, " y =", y)
```

Instrukcja print

```
print ("To jest tekst po wypisaniu")
```

Instrukcja print

```
print ("To jest tekst po wypisaniu")

print (2, 'dodać', 2, 'daje', 4)
```

Instrukcja print

```
print ("To jest tekst po wypisaniu")

print (2, 'dodać', 2, 'daje', 4)

print (2, 'dodać', 2, 'daje', 4, sep='*')
```

Instrukcja print

```
print ("To jest tekst po wypisaniu")

print (2, 'dodać', 2, 'daje', 4)

print (2, 'dodać', 2, 'daje', 4, sep='*')

print ('2 + 2', end='')
print ('daje', 4)
```