Politechnika Wrocławska

Python
wstęp do programowania
dla użytkowników WCSS

Dr inż. Krzysztof Berezowski
Instytut Informatyki, Automatyki i Robotyki
Politechniki Wrocławskiej

Wprowadzenie

CHARAKTERYSTYKA JĘZYKA

Filozofia języka Python

Język programowania:

- interpretowany,
- ogólnego przeznaczenia,
- wysokopoziomowy,

ukierunkowany na czytelność kodu źródłowego.

~~*„there's more than one way to do it”*~~

„there should be one

— and preferably only one —

obvious way to do it”

Kluczowe charakterystyki

- interpretowany,
- struktura przez wcięcia,
- dynamiczny system typów,
- wiązanie nazw w czasie wykonania
- automatyczne zarządzanie pamięcią,
- wsparcie różnych paradygmatów programow. (obiektowy, strukturalny, funkcjonalny),
- elastyczny i rozszerzalny,
- bogaty w biblioteki (NumPy SciPy Matplotlib).

Python - główne implementacje

- CPython (www.python.org)
  implementacja referencyjna w C
- Jython (www.jython.org)
  implementacja na maszynie wirtualnej Java
- IronPython (www.ironpython.org)
  implementacja na maszynie wirtualnej .NET

Python – środowiska zintegrowane

- Idle (natywne interaktywne IDE)
- Eclipse/PyDev (python, jython, IronPython)
- Netbeans for Python 6.5 EA (jython)
- Visual Studio (IronPython)
- Eric4/5
- Komodo (komercyjny)
- Wiele, wiele innych…

Interpreter Pythona

Tryb interaktywny

```
Python 2.7 (r27:82525, Jul 4 2010, …
Type "copyright", "credits" or "license()" …
>>> print "Hello, World!"
Hello, World!
>>> print 2*2
4
>>> 2*2
4
>>>
```

Praca wsadowa

Minimalny skrypt pythona (UN*X)

`script1.py`

```
#!/usr/bin/env python
print ”Hello, World!”
```

Uruchomienie:

```
kberezow@tesla:~/python$ python script1.py
```

albo:

```
kberezow@tesla:~/python$ chmod a+x script1.py
kberezow@tesla:~/python$ ./script1.py
Hello, World!
```

Dla przypomnienia

- Język programowania pozwala opisać program (sposób rozwiązania problemu obliczeniowego)

- poznać język programowania to poznać jego składnię oraz semantykę do opisu tych składowych

Obiektowy dynamiczny system typów

- wszystkie **wartości** są obiektami:
  - można dziedziczyć po typach wbudowanych (choć typy wbudowane w sumie nie są klasami)
- wszystkie **zmienne** (nazwy) są referencjami
  - nazwy wiązane są w czasie wykonania
- dynamiczny system typów
  - typy nie są hermetyzowane (klasa jest referencją)
  - zgodność typów jest kontrolowana (w pewnym umiarkowanym stopniu)
  - istnieją szerokie możliwości introspekcji

System typów języka Python

ZMIENNE I WARTOŚCI

System typów języka Python

- Dynamiczny
  - kontrola zgodności w czasie wykonania,
  - zgodność typów jest umiarkowanie kontrolowana,
  - typ mają **WARTOŚCI** a nie **ZMIENNE,**
  - istnieją szerokie możliwości introspekcji,
  - możliwość obliczeń na typach w czasie wykonania.
- Obiektowy
  - wszystkie wartości są obiektami,
  - wszystkie zmienne (nazwy) są referencjami,
  - wiązanie nazw w czasie wykonania (dynamic binding),

Typy proste

- Typ logiczny:

```
>>> print type(False)
<type 'bool'>
```

- Typ całkowity:

```
>>> print type(1)
<type 'int'>
```

- Typ zmiennoprzecinkowy:

```
>>> print type(1.0)
<type 'float'>
```

Typy numeryczne i operatory arytmetyczne

Typ	Reprezentacja
‘int’	Co najmniej 32-bitowa liczba całkowita (implementacja jako ‘long’ w języku C)
‘long’	Liczba całkowita nieograniczonej precyzji (uwaga na spadek wydajności przy przejściu przez granicę ‘int’ -> ‘long’)
‘float’	IEEE 754 podwójnej precyzji (64-bity reprezentacji, 80- obliczeniowo)
‘complex’	Para liczb typu ‘float’

Operacje wspierane przez wszystkie typy numeryczne

Operator	Interpretacja arytmetyczna
`x + y, x - y`	Dodawanie, odejmowanie
`x * y, x / y`	Mnożenie, dzielenie
`x // y, x % y`	Iloraz całkowity (9.0//4 == 2.0), reszta z dzielenia
`x ** y, pow(x, y)`	Potęgowanie
`+x, -x`	Wartość x, negacja x
`divmod(x, y)`	Para (x // y, x % y)
`int(x), long(x)`	Wartość całkowita x
`float(x),`	Wartość zmiennoprzecinkowa x
`complex(x, y)`	x + jy
`abs(x)`	Wartość bezwzględna liczby
`c.conjugate()`	Sprzężenie liczby zespolonej

Typy arytmetyczne ‘int’ i ‘float’

- **Operatory arytmetyczne (przykłady):**

```
10  + 9 = 19
10  - 9 = 1
10  * 9 = 90
10  / 9 = 1
10 // 9 = 1
10  % 9 = 1
10 ** 9 = 1000000000
```

```
10.0  + 9.0 = 19.0
10.0  - 9.0 = 1.0
10.0  * 9.0 = 90.0
10.0  / 9.0 = 1.11111111111
10.0 // 9.0 = 1.0
10.0  % 9.0 = 1.0
10.0 ** 9.0 = 1000000000.0
```

- **Konwersja ‘int’ do ‘long’ jest automatyczna**

```
def factorial(n):
    if n == 1:
        return 1;
    return n * factorial(n-1)
```

```
>>> res = factorial(300)
>>> print res.bit_length()
2042
>>> print type(res)
<type 'long'>
>>>
```

Typ ‘float’ - IEEE 754 double precision

```
def factorial(n):
    if n == 1:
        return 1.0;
    return n * factorial(n-1)
```

```
>>> res = factorial(170)
>>> print res
7.25741561531e+306
>>>
```

```
>>> res = factorial(171)
>>> print res
inf
>>>
```

- Ostrożnie z aproksymacją binarną liczb dziesiętnych

```
>>> 0.1 + 0.2
0.30000000000000004
>>> 0.1 + 0.2 == 0.3
False
```

- (później omówimy typ dziesiętny `Decimal`)

Manipulacja na bitach

- **Mają interpretację tylko na typach całkowitych:**

Operator	Interpretacja
A & B	Bitowy iloczyn logiczny
A | B	Bitowa suma logiczna
A ^ B	Bitowa różnica symetryczna
~A	Negacja bitowa
A << B	Przesunięcie A o B pozycji w lewo
A >> B	Przesunięcie A o B pozycji w lewo

- **Dynamiczna kontrola typów na przykładzie**

```
>>> 0.5 << 1
...
TypeError: unsupported operand type(s) for <<: 'float' and 'int'
>>>
```

Operatory podstawiania

- Operatory podstawiania:

Operator	Interpretacja
`A = B`	`Podstawia do zmiennej A referencję do B`
`A ◊= B`	`Podstawia do zmiennej A referencję do A◊B`

- ◊ - dowolny operator dwuoperandowy:

  +, -, *, /, //, %, **, &, |, ^, <<, >>, …

Operatory relacyjne

Operator	Interpretacja
`A == B`	Oblicza True, jeśli A jest równe B
`A != B` `A <> B`	Oblicza True, jeśli A jest różne B
`>`	Oblicza True, jeśli A jest większe od B
`<`	Oblicza True, jeśli A jest mniejsze od B
`>=`	Oblicza True, jeśli A jest większe bądź równe B
`<=`	Oblicza True, jeśli A jest mniejsze bądź równe B

Typy logiczny ‘bool’

- Operatory logiczne:

Priorytet	Operacja
!!!	`x or y`
!!	`x and y`
!	`not x`

- Operatory relacyjne mają wyższy priorytet niż logiczne, czyli:

```
not x == y
```

```
not (x == y)
```

Priorytety operatorów

Priorytet	Operatory
A == B	Potęgowanie **
A != B A <> B	Jednooperandowe: ~, +, -
>	Mnożenie i dzielenie: *, /, %, //
<	Dodawanie i odejmowanie +, -
>=	Przesunięcie bitowe >>, <<
<=	Iloczyn bitowy &
	Różnica symetryczna i suma bitowa ^, |
	Porównania (uporządkowanie) <=, <, >, >=
	Porównania (ekwiwalentność) <>, ==, !=
	Podstawienia =, ◊=

System typów języka Python

SEKWENCJE