Python 3 od podstaw

czyli jak poskromić węża

Michał Leszczyński

Na bazie szkolenia
Dominika Czarnoty i
Aleksandra Kawali

sli.do

#J413

Języki programowania – podział ze względu na paradygmat

Proceduralne
- C
- Fortran
- Basic

Obiektowe
- C++
- Objective-C
- C#
- Java
- Python
- Ruby

Funkcyjne
- Scala
- Haskell
- Erlang
- F#

Języki programowania – inny podział

Kompilowalne:
- C/C++
- Fortran
- Objective-C
- Java *
- C# *
- Haskell

Interpretowalne (skryptowe):
- Bash
- Python
- Ruby
- JavaScript
- PHP
- Perl

python™

- Prosty język wysokiego poziomu
- Nastawiony na czytelność kodu
- Posiada interpreter, w którym można testować różne rzeczy
- Preinstalowany na każdym(?) Linuxie
- Posiada (jeszcze) dwie wersje – Python 2 oraz Python 3

when will python 2 die

Wszystko | Wiadomości | Grafika | Zakupy | Filmy | Więcej | Ustawienia | Narzędzia

Około 143 000 000 wyników (0,77 s)

January 1, 2020

Python 2 support ends in 2020. Here's what you can do if you're stuck with Python 2 in what is fast becoming a Python 3 world. On **January 1, 2020**, the 2. 27 mar 2019

Python 2 EOL: How to survive the end of Python 2 | InfoWorld

https://www.infoworld.com › article › python-2-end-of-life-how-to-survive-t...

Soft korzystający z pythona

Przykłady:

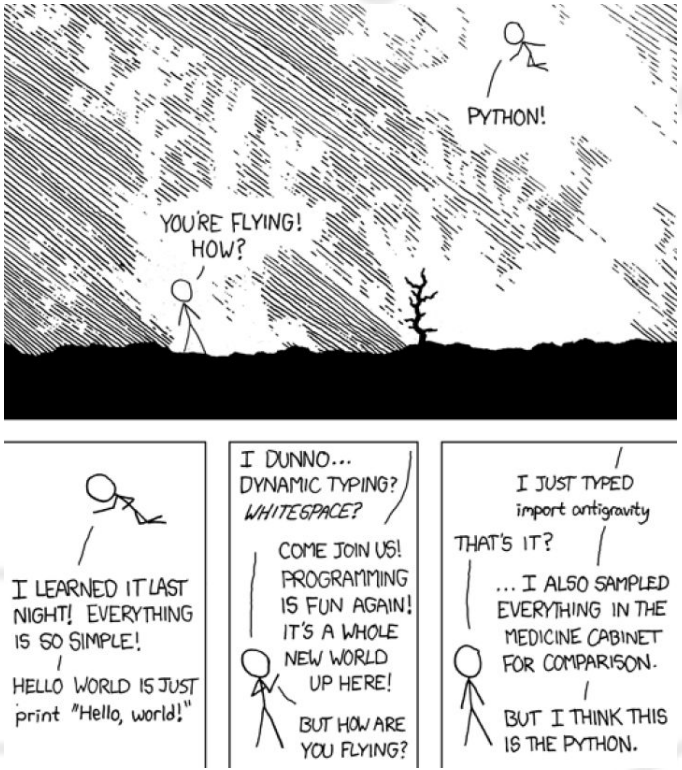
Szybki serwer HTTP:

python –m http.server

Szybki serwer FTP:

pip install pyftpdlib

python –m pyftpdlib


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Typy:

- `int` - liczba całkowita
- `float` - liczba zmiennoprzecinkowa, typ ten pozwala reprezentować duży zakres, ale z ograniczoną dokładnością
- `string` - łańcuch znaków, napis
- `bool` - typ logiczny może przyjmować wartości `True` lub `False`
- `None` - specjalna stała oznaczająca brak wartości (jak `null` w C/C++, Java, lub `nil` w niektórych językach)

```
calkowita = 7
rzeczywista = 7.5
rzeczywista = float(38)  # tworzenie liczby rzeczywistej z liczby calkowitej
rzeczywista = float('38')  # tworzenie liczby rzeczywistej z napisu
napis = 'witaj'
napis = "witaj"
napis = "Nie martw sie o 'pojedyncze' cudzyslowy."
logiczne = True
logiczne = False
```

Rzutowanie

W pythonie nie można zrobić `print("abc" + 5)` ponieważ obiekty nie są tego samego typu. Dlatego też operator dodawania nie ma sensu. Musimy powiedzieć jawnie, że chcemy traktować `5` jako `string`. W tym celu używamy rzutowania `print("abc" + str(5))`.

Inny przykład: `print(int("10") + 5)`, wypisze 15.

Komentarze

Interpreter ignoruje to co znajduje się w komentarzach. Komentarz liniowy rozpoczyna się od znaku `#`.

```
a = "To jest kod"
# A to komentarz
```

Komentarz blokowy rozpoczynamy i kończymy `"""`

```
"""
to
jest
komentarz
"""
```

Funkcje

Funkcje pozwalają zamknąć i nazwać blok kodu. Umożliwia to łatwe odwoływanie się do niego i oszczędza powtórzeń.

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

Wywołanie `square(5)` zwróci 25.

Dobre nawyki: Funkcje powinny mieć jedną odpowiedzialność, robić jedną rzecz, robić ją dobrze i obsługiwać wszystkie przypadki (w miarę możliwości). Funkcje pozwalają też nazwać kawałek kodu, nazwa powinna być odczasownikowa i opisywać to co funkcja robi. Złe nazwy: `a`, `x85`, `abc`, `qwerty`. Dobre nazwy: `cut_first_element`, `remove_everything`, `read_properties_from_file`.

Pętle

`for` służy do iterowania po kolekcji, jako kolekcję rozumiemy **Listy**, **Krotki**, **Słowniki**, **Generatory** (O tym później).

Wbudowana funkcja `range` zwraca kolekcję zawierającą kolejne liczby.

```
for i in range(5):
    print(i)
```

Wypisze 0, 1, 2, 3, 4

```
for i in range(10, 25, 5):
    print(i)
```

Wypisze 10, 15, 20. Możemy też podać listę z wartościami.

```
for i in [10, 25, 5]:
    print(i)
```

`while` to pętla, która wykonuje się dopóki warunek jest prawdziwy.

```
i=5
while i>0:
    print(i)
    i -= 1
```

Wypisze 5, 4, 3, 2, 1

Instrukcje warunkowe

Jeśli musimy zdecydować co ma się wykonać na podstawie jakiegoś warunku używamy instrukcji `if`.

```
if a > b:
    print(a)
elif a < b:
    print(b)
else:
    print("Są równe")
```

Jest też wersja inline, wartość zmiennej `a` będzie zależeć od warunku `condition`.

```
a = 5 if condition else 6
```

Zadania

BMI = masa[kg] / (wzrost[cm]^2)

1. Napisz funkcję która liczy BMI dla podanej masy ciała.
2. Funkcja która przymuje `n` i wypisze liczby od 1 do `n`, ale zamiast liczb podzielnych przez 3 pisze "Fizz" zamiast liczb podzielnych przez 5 pisze "Buzz" dla liczb podzielnych prez 3 i 5 pisze "FizzBuzz"(Jest to oklepane i typowe zadanie pojawiające się na rozmowach kwalifikacyjnych) np.

```
1
2
Fizz
4
Buzz
Fizz
7
8
```


I FIGURED OUT HOW TO
TURN ON MY MICROWAVE
USING PYTHON

Część dodatkowa

Pomoc w konsoli

W konsolę pythona wbudowane są dwie bardzo pomocne funkcje: `dir` oraz `help`:

```
a = [1,2,3]
dir(a)          # Wypisuje metody obiektu
help(a.append)  # Wyświetla tzw. 'docstring', czyli dokumentację do danej rzeczy
```

Struktury Danych

```
lista = [1, 2, 'c', 4, 5, 6]
krotka = (1, 2, 'e') # jest 'immutable' - nimodyfikowalna
slownik = { # tablica asocjacyjna, czyli indeksowana czymkolwiek (np. liczbami typu float)
#   klucz: wartość
    "a" : 123,
    "b" : "c",
    5 : "d"
}
```

Odwołania do elementów

```
lista[0]                # Pierwszy element
lista[4]                # Piąty element
lista[-1]               # Ostatni element
```

Slicing

```
a = [1, 2, 3, 4, 5]
a[0:2]                          # Pierwsze dwa elementy
a[-3:]                          # Od trzeciego od końca
a[1:4:2]                        # [start:stop:step]
a[::2]                          # Co drugi element z listy
```

Analogicznie dla słowników:

```
d = {} # pusty słownik

# Wstawianie elementów
d['a'] = 5
d[1] = 12
```

Operacje na listach

```
a.append('abc')          # Dodawanie elementów
a.pop()                          # Usunięcie ostatniego elementu
a = a + [3, 4, 5]        # Dodawanie list
b = ['a']*5                      # Oznacza to samo co ['a','a','a','a','a']
'a' in b                         # Sprawdza czy 'a' należy do b - zwraca wartość logiczną
```

Zadania

1. Na podstawie listy liczb całkowitych stwórz drugą, która zwiera tylko elementy parzyste
2. Znajdź metody listy, które pozwolą:
   - Usunąć element o danej wartości
   - Zwrócą indeks danego elementu
   - Usunąć element o danym indeksie
3.
```
a = list(range(10))
b = list(range(5,15))
c = []
```

   Dodaj do `c` część wspólną `a` i `b` (wynik funkcji `range` rzutujemy na listę, bo zwraca ona generator, o których będzie później)
4. Wyrzuć z listy `a` elementy znajdujące się też w `b` .
5.
```
s = "abcabcabaaasasda"
```

   Znajdź 2 sposoby na zamienienie wszystkich `'a'` na `'$'`