Akademia Pythona

III Instrukcje i skladnia

KN Pythona - Kurs Pythona

KN Pythona wita na kursie Pythona.

Plan:

- Wprowadzenie do instrukcji Pythona
- Przypisania, wyrazenia
- Testy if i reguly skladni

Programy w Pythonie

Hierarchia programow w Pythonie:

- Programy skladaja sie z modulow
- Moduly zawieraja instrukcje
- Instrukcje zawieraja wyrazenia
- Wyrazenia tworza i przetwarzaja obiekty

Instrukcje w Pythonie

Instrukcja	Rola	Przyklad
Przypisanie	Tworzenie referencji	a, *b = ‘dobry’, ‘zly’, ‘sredni’
Wywolania	Wykonywanie funkcji	log.write(‘test’)
if/elif/else	Wybor dzialania	if ‘python’ in text: print(text)
for/else	Iteracja po sekwencjach	for x in mylist: print(x)
while/else	Ogolne petle	while x > y: print(‘x > y’)
pass	Oznaczenie braku instrukcji	def x(): pass

Instrukcje w Pythonie

Instrukcja	Rola	Przyklad
break	Wyjscie z petli	while True: if input() == ‘exit’: break
continue	Kontynuacja petli	while True: if skiptest(): continue
def	Funkcje i metody	def add(a, b): return a + b
yield	Funkcje generatora	def gen(n): for i in n: yield i * 2
global	Przestrzenie nazw	x = ‘old’; def f(): global x, y; x = ‘new

Instrukcje w Pythonie

Instrukcja	Rola	Przyklad
nonlocal	Przestrzenie nazw	def outer(): x = ‘old’;def function(): nonlocal x; x = ‘n’
import	Dostep do modulow	import sys
from	Dostep do atrybutow	from sys import stdin
class	Budowanie klas	class SubClass(Sup): ...

Instrukcje w Pythonie

Instrukcja	Rola	Przyklad
try/except/finally	Przechwytywanie wyjatkow	try:action() except: print(‘e’)
raise	Wywolywanie wyjatkow	raise endSearch(location)
assert	Sprawdzanie w debugowaniu	assert x > y, ‘y too small’

Instrukcje w Pythonie

Instrukcja	Rola	Przyklad
with/as	Menedzery kontekstu	with open('text.txt') as text: ...
del	Usuwanie referencji	del dane[k]

Historia dwoch if-ow

C:

```
if (x > y) {
    x = 1;
    y = 2;
}
```

Historia dwoch if-ow

Python:

```
if x > y:
    x, y = 1, 2
```

Historia dwoch petli

C:

```c
#include<stdio.h>
int main()
{
    int r,c,num;
    r=1;
    while ( r <= 10)
    {
        c=1;
        while(c <= 10)
        {
            num=r*c;
            printf(" %3d",num);
            c=c+1;
        }
        r=r+1;
        printf("\n");
```

Historia dwoch petli

Python:

```
r = 1
while r <= 10:
    c = 1
    while c <= 10:
        num = r * c
        print(num, end=' ')
        c += 1
    r += 1
    print('\n')
```

Historia dwoch petli

Python:

```
for c in range(10):
    for r in range(10):
        print(r * c, end=' ')
    print('\n')
```

Co dodaje Python?

```
wiersz naglowka:
    zagniezdzony blok instrukcji
```

Co usuwa Python?

Python:

- Nawiasy sa opcjonalne
- Koniec wiersza jest koncem instrukcji
- Koniec wiersza to koniec bloku

Historia dwoch if-ow

C:

```
if (x)
    if (y)
        instrukcja1;
else
    instrukcja2;
```

Historia dwoch if-ow

Python:

```
if x:
    if y:
        instrukcja1
else:
    instrukcja2
```

Przypadki specjalne

Wiele instrukcji w jednym wierszu

```
a = 1; b = 2; print(a + b);
```

Przypadki specjalne

Instrukcja w wielu wierszach

```
m_list = [111,
          222,
          333]
```

Przypadki specjalne

Instrukcja w wielu wierszach

```
x = (a + b +
     c + d)
```

Przypadki specjalne

Instrukcja w wielu wierszach (niezalecane)

```
x = a + b + \
    c + d
```

Przypadki specjalne

Jednowierszowe cialo instrukcji

```
if x > y: print(x)
```

Instrukcje przypisania

Intstrukcje przypisania:

- Przypisania tworza referencje do obiektow
- Zmienne tworzone sa przy pierwszym przypisaniu
- Przed odniesieniem sie do zmiennych trzeba je najpierw przypisac
- Przypisania niejawne

Formy instrukcji przypisania

Forma podstawowa

```
x = 'python'
```

Formy instrukcji przypisania

Przypisania rozpakowujace krotki i listy

```
low, upp = 'python', 'PYTHON' # krotka
[low, upp] = ['python', 'PYTHON'] # lista
```

Formy instrukcji przypisania

Swap w Pythonie

```
a, b = b, a
a, b, c = c, a, b # etc...
```

Formy instrukcji przypisania

Przypisania sekwencji

```
[a, b, c] = (1, 2, 3)
(a, b, c) = 'abc'
```

Formy instrukcji przypisania

Zaawansowane wzorce przypisywania sekwencji

```
((a, b), c) = ('ab', 'c')
red, green, blue = range(3)
```

Formy instrukcji przypisania

Rozszerzona skladnia rozpakowania sekwencji

```
seq = [1, 2, 3, 4]
a, *b = seq # a = 1, b = [2, 3, 4]
*a, b = seq # a = [1, 2, 3], b = 4
a, *b, c = seq # a = 1, b = [2, 3], c = 4
a, b, *c = seq # a = 1, b = 2, c = [3, 4]
```

Zaawansowane wzorce przypisywania sekwencji

Zastosowanie w petli for

```
for (a, *b, c) in [(1, 2, 3, 4), (5, 6, 7, 8)]:
    print(a, b, c)
```

Formy instrukcji przypisania

Przypisania z wieloma celami

```
a = b = c = 1 # wspoldzielona referencja do obiektu
a = b = []
b.append(2) # a, b = ([2], [2])
```

Formy instrukcji przypisania

Rozszerzone instrukcje przypisania

```
x = x + y
x += y # forma rozszerzona
```

Formy instrukcji przypisania

Rozszerzone instrukcje przypisania

```
x += y
x *= y
x %= y
x &= y
x ^= y
x <<= y
x -= y
x |= y
x /= y
x **= y
x // y
x >>=y
```

Reguly dotyczace nazw zmiennych

Reguly:

- Skladnia: (znak _ lub litera) + (dowolna liczba liter, cyfr i znakow _)
- Wielkosc liter ma znaczenie (x to co innego niz X)
- Slowa zarezerwowane nie moga byc stosowane

Slowa zarezerwowane

Slowa zarezerwowane:

- False
- None
- True
- and
- as
- assert
- break
- class
- continue

Slowa zarezerwowane

Slowa zarezerwowane:

- def
- del
- elif
- else
- except
- finally
- for
- from

Slowa zarezerwowane

Slowa zarezerwowane:

- global
- if
- import
- in
- is
- lambda
- nonlocal
- not
- or

Slowa zarezerwowane

Slowa zarezerwowane:

- pass
- raise
- return
- try
- while
- with
- yield

Konwencje dotyczace nazewnictwa

Konwencje:

- nazwy _* nie sa importowane przy uzyciu 'from modul import *'
- nazwy __*__ sa zdefiniowane przez system i maja specjalne znaczenie dla interpretera
- nazwy __* sa lokalne dla zawierajacych je klas
- nazwa _ zachowuje w sesji interaktywnej wynik ostatniego wyrazenia

Wiecej » PEP8

Instrukcje wyrazen

Instrukcje wyrazen

```
function(a, b)
obj.method(a, b, c)
yield x ** 2
```

Instrukcje wyrazen

Modyfikacje w miejscu

```
l = [1, 2]
l.append(23)
# vs
l = l.append(4)
# l = None
```

Testy if i reguly skladni

Ogolny format

```
if <test1>:
    <instrukcje1>
elif <test2>:
    <instrukcje2>
else:
    <instrukcje3>
```

Reguly skladni Pythona

Reguly skladni:

- Instrukcje wykonywane sa jedna po drugiej, o ile nie wskazemy innego sposobu
- Granice blokow i instrukcji wykrywane sa w sposob automatyczny
- Instrukcje zlozone skladaja sie z wiersza naglowka, znaku dwukropka i wcietych instrukcji
- Puste wiersze, spacje i komentarze sa zazwyczaj ignorowane
- Lancuchy znakow dokumentacji sa ignorowane, ale zapisywane i wyswietlane przez narzedzia

Wyrazenie trojargumentowe if/else

Wyrazenie trojargumentowe if/else

```
x = 1 if test() else 2
```