Odczyt skanera laserowego w jezyku Python

Janusz Jakubiak, Paweł Ludwików

31 marca 2015

1 Przykład programu do odczytu skanera

Przed uruchomieniem programu należy zmieniuć mu uprawnienia do wykonywania (`chmod`), w programie należy ustawić numer robota oraz wczytać driver `RosAriaDriver`

```
#!/usr/bin/env python
# -*- coding: latin2 -*-

# dokumentacja klasy RosAriaDriver na stronie
# http://panamint.ict.pwr.wroc.pl/~dbaransk/rosaria_drive/dox/html/index.html
from time import sleep
from drive import RosAriaDriver

# Tu podaj numer robota
robot=RosAriaDriver('/PIONIER3')

# Wczytanie danych ze skanera
skan=robot.ReadLaser()

# Wyswietlenie wszystkich odczytow
print skan

# liczba odczytow (danych w tabeli)
print len(skan)

# Wyswietlenie piatego elementu
print skan[5]

# wyswietlenie 6 kolejnych skanow
print skan[20:25]

# Z wykorzystaniem NumPy
# http://www.numpy.org/
from numpy import *
print skan>1
print skan[skan>2]
```

A Elementy składni języka Python

Python jest przejrzystym i wszechstronnym językiem programowania obiektowego. Używa eleganckiej składni, dzięki czemu programy są łatwe do zrozumienia. Zapewnia dużą różnorodność wbudowanych typów danych: liczbowych (zmiennoprzecinkowe, zespolone, liczby całkowite o nieskończonej precyzji), tekstowych, listy, słowniki, itp. Język zapewnia obsługę wyjątków pozwalając na efektywniej obsługiwać błędy.

Podstawowym złożonym typem danych w języku Python jest lista. Lista jest zapisywana jako wyszczególnienie wartości rozdzielone przecinkami ujęta w nawiasy kwadratowe. Elementy listy nie muszą byc tego samego typu. np. `a=['spam', 'eggs', 100, 1234]`.

Dostęp do elementu listy uzyskuje się stosując operator indeksowania, np. `a[2]`. Początkowy element posiada indeks 0. Użycie indeksów ujemnych powoduje zwrócenie odpowiedniej wartości elementu listy licząc od końca. Indywidualne elementy listy można dowolnie zmieniać, np: `a[2]=a[2]+3`.

Przykładowy program w języku Python (zapożyczony z wprowadzenia):

```
# liczby Fibonnaci'ego
a, b = 0, 1
while b < 10:
        print b
        a, b = b, a+b
```

Pierwsza linia zawiera przykład wielokrotnego przypisania: wartości `a` i `b` jednocześnie przyjmują wartości 0 i 1. Jak w każdym przypisaniu wartości prawostronne są obliczane najpierw, a potem podstawiane do lewej strony.

Pętla `while` wykonuje się tak długo jak długo warunek (tutaj `b < 10`) jest prawdziwy. W Pythonie, podobnie jak w C, każda niezerowa wartość całkowita ma wartość logicznej prawdy, każda niepusta lista lub ciąg znaków także. Standardowe operatory porównań są identyczne jak w C: `<` (mniejszy), `>` (większy), `==` (równy), `<=` (mniejszy lub równy), `>=` (większy lub równy) i `!=` (różny). W języku zdefiniowane są dwa wyrażenia logiczne: `True` oraz `False` posiadające logiczną wartość odpowiednio prawdy i fałszu. Łączenie warunków w złożone wyrażenia następuje przy pomocy opeatorów `and`, `or` i `not`.

Grupowanie wyrażeń odbywa się poprzez *wcięcia*. Każda linia tego samej grupy musi być wcięta o tą samą liczbę znaków. Koniec grupy oznacza się pustą linią. Zagnieżdżenie grup uzyskuje się przez zagnieżdżenie wcięć.

A.1 Kontrola przebiegu programu

Wyrażenie warunkowe uzyskuje się prawie identycznie jak w innych językach:

```
if x < 0:
        print 'ujemny'
elif x == 0:
        print 'zero'
else:
        print 'dodatni'
```

Pętla `for` różni się od odpowiedników w C i Pascalu. W Pythonie `for` iteruje po wartościach podanej sekwencji (listy lub ciągu) w kolejności ich pojawiania się.

Jeżeli zachodzi potrzeba iterowania po sekwencji liczb należy użyć funkcji `range()`, generującej listę zawierającą ciąg arytmetyczny.

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Punkt końcowy nie jest częścią wygenerowanej listy, `range(10)` generuje listę 10 wartości, dopuszczalnych indeksów listy o długości 10. Można także określić początek: `range(5,10)` oraz także krok: `range(0, 10, 3)`

Wyrażenie `break`, podobnie jak w C, powoduje opuszczenie najbardziej zagnieżdżonej pętli `for` lub `while`. Wyrażenie `continue` powoduje wykonanie pętli od początku.

A.2 Funkcje wbudowane

`len(a)` — podaje długość (liczbę elementów) `a`
`max(a)` — podaje wartość maksymalnego elementu listy `a`
`min(a)` — podaje wartość minimalnego elementu listy `a`
`print ...` — drukowanie wartości, podanie przecinka na końcu nie powoduje przejścia do następnej linii

Dla listy `l` dostępne są następujące funkcje:
`l.index(w)` — zwraca indeks pierwszego elementu o wartości `w`
`w in l` — test, czy lista `l` zawiera element o wartości `w`

A.3 Obsługa wyjątku Ctrl-C

```
def fun(k):       # funkcja uzytkownika
        ...
def safefun(k):   # nadzorca
        try:
                fun(k)
        except KeyboardInterrupt:
                print 'Ctrl-C'

go(safefun)       # uruchomienie
```