Astronomiczny Kurs Studencki
cz. 3 - Pętle, warunki i funkcje

Paweł Biernacki

23 marca 2012 r.

Plan prezentacji

Struktura programu w pythonie

1. Programy są złożone z modułów.
2. Moduły zawierają stwierdzenia.
3. Stwierdzenia zawierają wyrażenia.
4. Wyrażenia tworzą i działają na obiektach.

Jak to wygląda w kodzie?

```
1 Linia naglowka:
2     Zagniezdzone wyrazanie
```

Struktura programu w pythonie

1. Programy są złożone z modułów.
2. Moduły zawierają stwierdzenia.
3. Stwierdzenia zawierają wyrażenia.
4. Wyrażenia tworzą i działają na obiektach.

Jak to wygląda w kodzie?

```
Linia naglowka:
    Zagniezdzone wyrazanie
```

Ale też:

```
if (A == 1 and
    B == 2 and
    C == 3):
    print('spam' * 3)
```

Podstawowe przykłady

```
1 if <test1>:           # Test if
2     <wyrazenia1>      # i zwiazany z nim blok
3 elif <test2>:         # Opcjonalny elif
4     <wyrazenia2>
5 else:                 # Opcjonalny else
6     <wyrazenia3>
```

Podstawowe przykłady

```
if <test1>:          # Test if
    <wyrazenia1>     # i zwiazany z nim blok
elif <test2>:        # Opcjonalny elif
    <wyrazenia2>
else:                # Opcjonalny else
    <wyrazenia3>
```

I coś 'na żywo':

```
x = 'killer rabbit'
if x == 'roger':
    print("how's jessica?")
elif x == 'bugs':
    print("what's up doc?")
else:
    print('Run away! Run away!')
```

Źle

```
1       x = 'SPAM'              # Err: pierwsza linia indent.
2   if 'rubbery' in 'shrubbery':
3       print(x * 8)
4           x += 'NI'           # Err: niespodziewana indent.
5           if x.endswith('NI'):
6                   x *= 2
7               print(x)        # Err: niespojna indentacja
```

Źle

```
1      x = 'SPAM'              # Err: pierwsza linia indent.
2  if 'rubbery' in 'shrubbery':
3      print(x * 8)
4          x += 'NI'           # Err: niespodziewana indent.
5          if x.endswith('NI'):
6                  x *= 2
7              print(x)        # Err: niespojna indentacja
```

Dobrze

```
1  x = 'SPAM'
2  if 'rubbery' in 'shrubbery':
3      print(x * 8)
4      x += 'NI'
5      if x.endswith('NI'):
6          x *= 2
7          print(x)    # Wypisuje "SPAMNISPAMNI"
```

Podstawowe przykłady

```
1 while <test>:          # Test petli
2     <wyrazenia1>       # Cialo petli
3 else:                  # Opcjonalny else
4     <wyrazenia2>       # Wykonaj jesli nie opusciles petli
```

Podstawowe przykłady

```
while <test>:          # Test petli
    <wyrazenia1>       # Cialo petli
else:                  # Opcjonalny else
    <wyrazenia2>       # Wykonaj jesli nie opusciles petli
```

I coś 'na żywo':

```
a=0; b=10
while a < b:
    print(a, end=' ')
    a += 1                    # lub a = a + 1
```

Czego wynikiem jest:

0 1 2 3 4 5 6 7 8 9

Dodatkowe elementy

Te elementy to:

- `break`
- `continue`
- `pass`

A tak to działa:

```
while <test1>:
    <wyrazenia1>
    # Opusc petle teraz, przeskocz else
    if <test2>: break
    # Idz na gore do test1
    if <test3>: continue
else:
    # Wykonaj jesli nie opusciles petli z break
    <wyrazenia2>
```

Podstawowe przykłady

```
1 for <cel> in <obiekt>:
2     <wyrazenia>
3 else:
4     <wyrazenia>
```

Podstawowe przykłady - c.d.

```
1  sum = 0
2  for x in [1, 2, 3, 4]:
3      sum = sum + x
4  print sum
5
6  # oraz np.
7  T = ("and", "I'm", "okay")
8  for x in T:
9      print(x, end=' ')
10
11 # jak tez
12 for y in range(len(T)):
13     print (y, end=' ')
```

Czego wynikiem jest:

Podstawowe przykłady - c.d.

```
1  sum = 0
2  for x in [1, 2, 3, 4]:
3      sum = sum + x
4  print sum
5
6  # oraz np.
7  T = ("and", "I'm", "okay")
8  for x in T:
9      print(x, end=' ')
10
11 # jak tez
12 for y in range(len(T)):
13     print (y, end=' ')
```

Czego wynikiem jest:

```
10
and I'm okay
0 1 2
```

Co się wiąże z funkcjami?

- `def` tworzy obiekt i przypisuje go do nazwy
- `lambda` tworzy obiekt ale zwraca go jako wynik
- `return` wysyła rezultat (obiekt) do wywołania
- `yield` wysyła rezultat (obiekt) do wywołania, ale pamięta gdzie się zatrzymało
- `global` deklaruje zmienne modułowe które będą przypisane

Podstawowe przykłady

```
def <nazwa>(arg1, arg2,... argN):
    <wyrazenia>
    return <wartosc>
```

```
if test:
    def func(): # Zdefiniuj func tak
...
else:
    def func(): # Albo w taki sposob
...

    func()      # Uruchom wybrana wersje
```

Banalny(?) przykład

```
1 def times(x, y):
2     return x * y
3
4 x = times(3.14, 4)  # zapisanie do obiektu
5 print times(2, 4)   # 8
6 print times('Ni', 4)# 'NiNiNiNi'
```

A reszta to już tylko Wasza kreatywność!

Funkcje przydatne przy pytaniu użytkownika

```
1 x = input('Podaj wartosc: ',)
2
3 y = raw_input('Twoja zmienna ',)
```

Zanim zadanie domowe...

...popatrzmy na przykład zbierający warunki i pętle:

```
1  while True:
2      reply = input('Enter text:')
3      if reply == 'stop':
4          break
5      elif not reply.isdigit():
6          print('Bad!' * 8)
7      else:
8          num = int(reply)
9          if num < 20:
10             print('low')
11         else:
12             print(num ** 2)
13 print('Bye')
```

Przykład działania

```
Enter text:19
low
Enter text:20
400
Enter text:spam
Bad!Bad!Bad!Bad!Bad!Bad!Bad!Bad!
Enter text:stop
Bye
```

Zadania domowe

Tj. coś do przećwiczenia (wreszcie kładziemy łapki na pythonie).

Napisać program, który:

1. Będzie wypisywał liczby pierwsze z zadanego przedziału i będzie potrafił rozpoznać nieprawidłowe dane wejściowe. Stowrzyć funkcję która będzie mogła być używana w innych programach.
2. Będzie tłumaczył liczby z systemu dziesiątkowego na szesnastkowy i na odwrót; powinien pytać użytkownika o dane wejściowe oraz potrafić rozpoznać w jakim systemie są zakodowane (podpowiedź: można/należy posłużyć się prefiksem przy liczbie).

W swojej prezentacji korzystałem z przykładów z książki M. Lutza “Learning Python”.