Έλεγχος ροής

Μιχάλης Δρακόπουλος

Μάρτιος 2021

Από το MATLAB στην Python

Εντολές ροής προγράμματος

Πολύ σημαντική παρατήρηση: Οι παρακάτω εντολές (`if`, `while`, `for`) εκτελούν κατάλληλα μια ομάδα (block) εντολών, γνωστό και ως **σώμα** της εντολής. Στο MATLAB η οριοθέτηση του block γίνεται με τη λέξη `end`. Στην Python η οριοθέτηση επιτυγχάνεται με **στοίχιση σε εσοχή (indentation)** των εντολών του block.

Εντολές ελέγχου `if`

```
if condition_1:
    commands_1
elif condition_2:
    commands_2
...
elif condition_N:
    commands_N
else:
    commands
```

Οι διακλαδώσεις `elif` είναι προαιρετικές (≥ 0), όπως και η `else` που αν υπάρχει πρέπει να είναι η τελευταία διακλάδωση στη δομή `if`.

[1]:
```
# Simple if
length = float(input('Length? '))
scale = float(input('Scale? '))
if scale != 1:
    print('Scaled by', scale)
    length = length*scale      # length *= scale
print('Final length =', length)
```

```
Length?  100
Scale?  1.2
```

```
Scaled by 1.2
Final length = 120.0
```

[2]:
```
# Simple if
length = float(input('Length? '))
```

```
scale = float(input('Scale? '))
if scale != 1:
    print('Scaled by', scale)
    length = length*scale      # length *= scale
print('Final length =', length)
```

```
Length?  100
Scale?  1

Final length = 100.0
```

[3]:
```
# if-else
x = int(input('? '))
if x%2 == 0:
    print('Even')
else:
    print('Odd')
print('--- The End ---')
```

```
?  12

Even
--- The End ---
```

[4]:
```
# if-elif-else
x = float(input('x?'))
y = float(input('y?'))
z = float(input('z?'))
if x < y and x < z:
    m = x
elif y < z:
    m = y
else:
    m = z
print('The minimum of', x, y, z, 'is', m)
```

```
x? 18
y? 12
z? 19

The minimum of 18.0 12.0 19.0 is 12.0
```

[5]:
```
# Nested ifs
x = int(input('x? '))
print('x = ', x)
if x%2 == 0:
    if x%3 == 0:
        print(x, 'is divisible by 2 and 3')
    else:
        print(x, 'is divisible by 2 but not by 3')
```

```
elif x%3 == 0:
    print(x, 'is divisible by 3 but not by 2')
else:
    print(x, 'is not a multiple of 2 or 3')
print('--- The End ---')
```

```
x?  15
```

```
x =  15
15 is divisible by 3 but not by 2
--- The End ---
```

Εντολές επανάληψης `while`

```
while condition:
    commands
```

[6]:
```
# Powers of 2
N = int(input('N? '))
v = 1
i = 0
while i <= N:
    print('2 ^', format(i, '2d'), '=', format(v, '4d'))
    v = 2*v
    i = i + 1
print('--- The End ---')
```

```
N?  10
```

```
2 ^  0 =    1
2 ^  1 =    2
2 ^  2 =    4
2 ^  3 =    8
2 ^  4 =   16
2 ^  5 =   32
2 ^  6 =   64
2 ^  7 =  128
2 ^  8 =  256
2 ^  9 =  512
2 ^ 10 = 1024
--- The End ---
```

[7]:
```
# Sum of digits
N = int(input('N?'))
dsum = 0
while N > 0:
    dsum = dsum + N%10
    N = N // 10
print('Digit sum is', dsum)
```

```
N? 25234

Digit sum is 16
```

Εντολές επανάληψης `for`

Στην απλούστερη μορφή της (και σε αντιστοιχία με παρόμοιες εντολές επανάληψης που συναντάμε σε άλλες γλώσσες προγραμματισμού) η εντολή `for` διατρέχει μια αλληλουχία *ακεραίων* αριθμών ξεκινώντας από μια **αρχική τιμή** εως μια **τελική τιμή** με κάποιο (προαιρετικό) **βήμα**.

Στην Python αυτό γίνεται με τη βοήθεια της συνάρτησης `range()`

```
for variable in range(...):
    commands
```

Η συνάρτηση `range()` Δημιουργεί διαδοχικά μια αλληλουχία ακεραίων. Έχει τις εξής παραλλαγές: 1. Τιμές από 0 έως `K-1` με βήμα 1 > `python range(K)` 2. Τιμές από `M` έως `N-1` με βήμα 1 > `python range(M, N)` 3. Τιμές από `M` έως `N-1` με βήμα `S` (για `S < 0`, πρέπει `M > N`) > `python range(M, N, S)`

Η της `for` παίρνει διαδοχικά τις τιμές που δίνει η `range()` και για κάθεμια από αυτές εκτελεί τις στο σώμα της.

[8]:
```
# Sum of i, i**2, i**3, i = 0, 1, 2, ..., 10
s = 0
ssq = 0
scb = 0
for i in range(11):
    print(i, i**2, i**3, sep = '\t')
    s = s + i
    ssq = ssq + i**2
    scb = scb + i**3
print(20*'-')
print(s, ssq, scb, sep = '\t')
```

```
0       0       0
1       1       1
2       4       8
3       9       27
4       16      64
5       25      125
6       36      216
7       49      343
8       64      512
9       81      729
10      100     1000
--------------------
55      385     3025
```

[9]:
```
for i in range(-2, 5):
    print(i, end=' ')
```

```
print('\n--- The End ---')
```

```
-2 -1 0 1 2 3 4
--- The End ---
```

[10]:
```
print('Countdown!')
for i in range(10, -1, -1):
    print(i)
print('Lift-off!!')
```

```
Countdown!
10
9
8
7
6
5
4
3
2
1
0
Lift-off!!
```

Οι εντολές `break` και `continue`

- Η `break` διακόπτει τις επαναλήψεις σε `for` και `while`
- Η `continue` παραλείπει την τρέχουσα επανάληψη στην εντολή `for`

[11]:
```
# Find first integer divisible by 11 and 13
x = 1
while True:
    if x%11 == 0 and x%13 == 0:
        break
    x = x + 1
print(x, 'is divisible by 11 and 13')
```

```
143 is divisible by 11 and 13
```

[12]:
```
# Sum of i, i**2, i**3, i = 0, 2, 4, ..., 10
s = 0
ssq = 0
scb = 0
for i in range(11):
    if i%2 != 0:
        continue
    print(i, i**2, i**3, sep = '\t')
    s = s + i
    ssq = ssq + i**2
```

```
    scb = scb + i**3
print(20*'-')
print(s, ssq, scb, sep = '\t')
```

```
0       0       0
2       4       8
4       16      64
6       36      216
8       64      512
10      100     1000
--------------------
30      220     1800
```

Ο κλάδος `else` των εντολών `for` και `while`

Μια ιδιομορφία της Python είναι ότι οι εντολές `for` και `while` μπορούν να έχουν *προαιρετικά* έναν κλάδο `else`

```
for ... in range(...):
    commands_1
else:
    commands_2
```

```
while condition:
    commands_1
else:
    commands_2
```

Και στις 2 περιπτώσεις οι εντολές `commands_1` αποτελούν το *σώμα* της εντολής που εκτελείται *επαναληπτικά*. Οι εντολές `commands_2` εκτελούνται στο **τέλος** των επαναλήψεων και **μόνο όταν** οι επαναλήψεις ολοκληρώθηκαν χωρίς διακοπή. Εκτελούνται δηλαδή **μόνο** στην περίπτωση που οι επαναλήψεις **δεν διακόπηκαν** απότομα από κάποιο `break` στις `commands_1`.

[13]:
```
# Check if y is prime - without while-else
y = int(input('? '))
x = y // 2
is_prime = True
while x > 1 and is_prime:
    if y%x == 0:
        print(x, 'divides', y)
        is_prime = False
    else:
        x = x - 1
if is_prime:
    print(y, 'is prime')
```

```
?  143
```

```
13 divides 143
```

[14]:
```
# Check if y is prime - with while-else
y = int(input('? '))
x = y // 2
while x > 1:
    if y%x == 0:
        print(x, 'divides', y)
        break
    x = x - 1
else:
    print(y, 'is prime')
```

```
?  157
```

```
157 is prime
```