ΜΕΜ104 Γλώσσα Προγραμματισμού Ι

Μιχάλης Πλεξουσάκης

6 Οκτωβρίου 2019

Μαθηματικά και Εφαρμοσμένα Μαθηματικά

Περιεχόμενα

1. Μεταβλητές

2. Εντολές συνθήκης

3. Συμβολοσειρές

4. Είσοδος δεδομένων από τον χρήστη

Μεταβλητές

Μεταβλητές

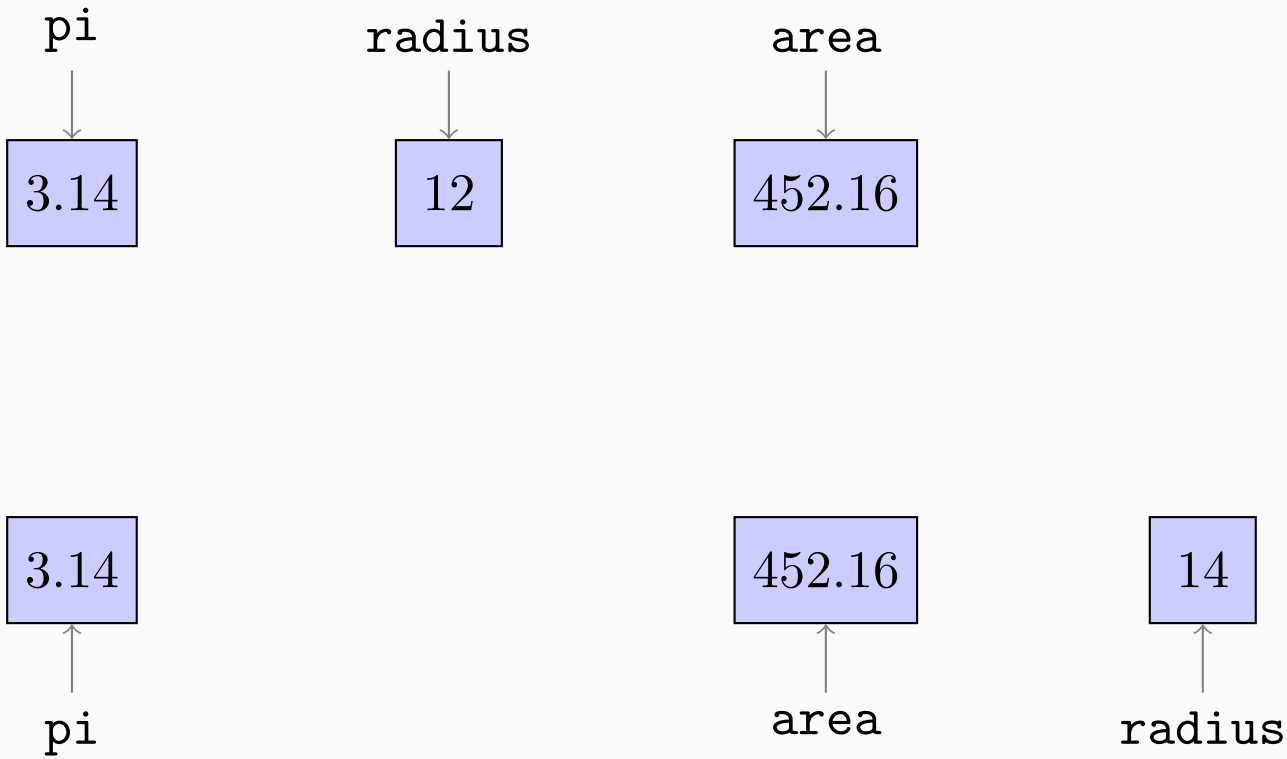
Οι *μεταβλητές* (*variables*) παρέχουν έναν τρόπο για να συσχετίζουμε ονόματα με αντικείμενα. Ο κώδικας

```
pi = 3.14
radius = 12
area = pi * radius**2
radius = 14
```

συνδέει τα ονόματα `pi` και `area` σε αντικείμενα τύπου `float` και το όνομα `radius` σε ένα αντικείμενο τύπου `int`.

Μετά την εκτέλεση της εντολής `radius = 14` το όνομα `radius` θα επανασυνδεθεί σε ένα διαφορετικό αντικείμενο τύπου `int`.

Μεταβλητές



Αυτή η ανάθεση δεν επιφέρει καμία αλλαγή στην τιμή η οποία συνδέεται η μεταβλητή `area`.

Μεταβλητές

- Στην Python, μια μεταβλητή είναι *απλώς ένα όνομα και τίποτα παραπάνω*.
- Μια εντολή *ανάθεσης τιμής* συσχετίζει το όνομα στα αριστερά του συμβόλου = με το αντικείμενο που υποδηλώνεται από την παράσταση στα δεξιά του συμβόλου =
- Ένα αντικείμενο μπορεί να έχει ένα, περισσότερα από ένα, ή κανένα όνομα που συσχετίζεται με αυτό
- Επειδή τα προγράμματα δεν διαβάζονται μόνο από υπολογιστές, είναι σημαντικό να επιλέγουμε *κατάλληλα* ονόματα μεταβλητών.

Μεταβλητές

Ποιό από τα παρακάτω δύο κομμάτια κώδικα θα επιλέγατε;

```
# Υπολογισμός εμβαδού
a = 3.14
b = 11
c = a*b**2
```

```
# Υπολογισμός εμβαδού
pi = 3.14
diam = 11
area = pi*diam**2
```

Και τα δύο, αν εκτελεστούν, θα κάνουν ακριβώς το ίδιο πράγμα.

Διαβάζοντας τον κώδικα στα αριστερά, δεν αναγνωρίζουμε κάποιο λάθος.

Κοιτάζοντας τον κώδικα στα δεξιά, γρήγορα καταλαβαίνουμε ότι η μεταβλητή `diam` θα έπρεπε να ονομαστεί `radius`, ή να αλλάξουμε τον τύπο υπολογισμού του εμβαδού.

Μεταβλητές

Τα ονόματα μεταβλητών στην Python

- επιτρέπεται να περιέχουν πεζά και κεφαλαία γράμματα, ψηφία και τον χαρακτήρα _, αλλά δεν μπορούν να ξεκινούν με ψηφίο
- γίνεται διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων

Ακόμα, υπάρχει ένας αριθμός *δεσμευμένων λέξεων*, οι λεγόμενες και *λέξεις-κλειδιά* με ειδική σημασία για την Python, που δεν μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών:

```
and, as, assert, break, class, continue, def, del,
   elif, else, except, exec, finally, for, from,
global, if, import, in, is, lambda, not, or, pass,
   print, raise, return, try, with, while, yield
```

Μεταβλητές

Για παράδειγμα, τα παρακάτω μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών

```
name, mylist, x1, number_of_players, computeArea,
                letter, digit, _special,
```

ενώ τα επόμενα δεν μπορούν

```
1name, 2y, class, exec, -num, #number
```

Μεταβλητές

Η ανάθεση τιμής γίνεται με τον τελεστή `=`. Για παράδειγμα,

```
pi = 3.14159
```

Η Python επιτρέπει τις πολλαπλές αναθέσεις τιμών, όπως

```
x = 2; y = 3 + 12.0/7
u = v = w = 0
```

οι οποίες συνδέουν το `x` με το 2 και το `y` με το 3, και τα `u`, `v`, `w` με το 0. Όλες οι παραστάσεις στη δεξιά πλευρά της ανάθεσης υπολογίζονται πριν από οποιαδήποτε αλλαγή τιμής. Αυτό επιτρέπει, π.χ.,

```
x, y = 2, 3
x, y = y, x  # Nice!
```

για την αντιμετάθεση των συνδέσεων των μεταβλητών `x` και `y`.

Εντολές συνθήκης

Εντολές συνθήκης

Τα *διακλαδιζόμενα προγράμματα* (branching programs) βασίζονται (συνήθως) σε μια *εντολή συνθήκης* με την παρακάτω δομή:

- έναν έλεγχο, δηλαδή μια παράσταση η οποία δίνει τιμή `True` ή `False`
- ένα τμήμα κώδικα που εκτελείται αν ο έλεγχος δίνει τιμή `True`, και
- ένα τμήμα κώδικα που εκτελείται αν ο έλεγχος δίνει τιμή `False`

Μετά την εκτέλεση μιας εντολής συνθήκης, η εκτέλεση του προγράμματος συνεχίζεται με τον κώδικα που ακολουθεί την εντολή.

Εντολές συνθήκης

Στην Python, μια εντολή συνθήκης έχει τη μορφή

`if` *λογική παράσταση*`:`
    τμήμα κώδικα
`else:`
    τμήμα κώδικα

Θα μπορούσαμε, για παράδειγμα, να γράψουμε

```
if x >= 0:
    absx = x
else:
    absx = -x
```

για να υπολογίσουμε την απόλυτη τιμή του αριθμού `x`.

Εντολές συνθήκης

Για να ελέγξουμε αν ο ακέραιος x είναι άρτιος ή περιττός μπορούμε να χρησιμοποιήσουμε την εντολή συνθήκης

```
if x%2 == 0:
    print('Άρτιος')
else:
    print('Περιττός')
```

Προσοχή: Η παράσταση `x%2 == 0` είναι `True` όταν το υπόλοιπο της διαίρεσης του `x` με το 2 είναι μηδέν, και `False` διαφορετικά.

Μην ξεχνάτε ότι το `==` είναι ο τελεστής σύγκρισης!

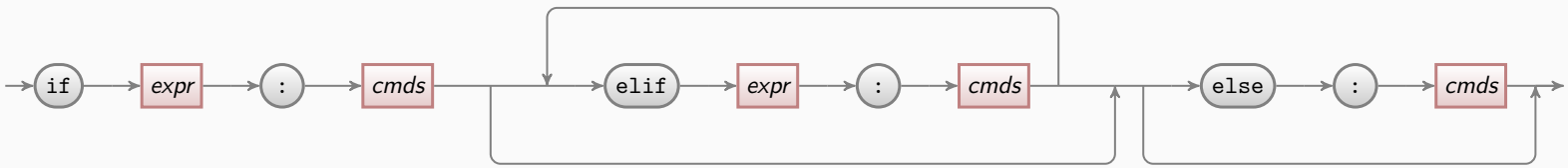
Εντολές συνθήκης

Σύνθετες λογικές παραστάσεις μπορούν να σχηματιστούν με τους λογικούς τελεστές `not`, `and` και `or`. Για παράδειγμα, το τμήμα κώδικα παρακάτω υπολογίζει τον μικρότερο των αριθμών `x, y` και `z` (βεβαιωθείτε ότι καταλαβαίνετε γιατί):

```
if x < y and x < z:
    print('x is smallest')
elif y < z:
    print('y is smallest')
else:
    print('z is smallest')
```

Η λέξη-κλειδί `elif` στον παραπάνω κώδικα σημαίνει “else if”, (διαφορετικά αν).

Εντολές συνθήκης



Εντολές συνθήκης

Ένας κομψότερος, ίσως, τρόπος υπολογισμού του μικρότερου των αριθμών `x, y` και `z` είναι ο ακόλουθος, ο οποίος μπορεί να γενικευθεί σε μεγαλύτερο πλήθος αριθμών:

```
smallest = x

if y < smallest:
    smallest = y
if z < smallest:
    smallest = z

print('smallest number is', smallest)
```

Εντολές συνθήκης

Ένα έτος είναι δίσεκτο αν διαιρείται με το 4 αλλά όχι με το 100,
ή αν διαιρείται με το 400:

```
year = 2002
leap = year%4 == 0 and year%100 !=0 or year%400 == 0

if leap:
    print(year, 'is a leap year')
else:
    print(year, 'is not a leap year')
```

Εντολές συνθήκης

Στην Python, οι *εσοχές* ερμηνεύονται σημασιολογικά. Αυτή είναι μια ιδιομορφία της γλώσσας και χρησιμοποιείται για να απομονώσει τμήματα κώδικα.

Για την εντολή συνθήκης οι εσοχές απομομώνουν το τμήμα κώδικα που εκτελείται όταν η συνθήκη έχει τιμή `True` και το τμήμα κώδικα που εκτελείται όταν η συνθήκη έχει τιμή `False`.

Ακόμα, οι λογικές παραστάσεις και η λέξη κλειδί `else` ακολουθούνται από άνω και κάτω τελεία.

Εντολές συνθήκης

Οι εντολές συνθήκης μπορεί να είναι φωλιασμένες (nested), όπως, για παράδειγμα, στο τμήμα κώδικα

```
if x % 2 == 0:
    if x % 3 == 0:
        print('Divisible by 2 and 3')
    else:
        print('Divisible only by 2')
elif x % 3 == 0:
    print('Divisible only by 3')
else:
    print('Not divisible by 2 or 3')
```

που ελέγχει τη διαιρετότητα του `x` με το 2 και το 3.

Εντολές συνθήκης

Η Python παρέχει μεθόδους για τον υπολογισμό του ελαχίστου/μεγίστου οποιουδήποτε πλήθους αριθμών:

```
>>> min(3, -2, 8)
-2
>>> x = 3; y = -8; z = 4; w = 1
>>> min(x, y, z, w)
-8
>>> max(3.1, 4.15)
4.15
```

αλλά και για την απόλυτη τιμή

```
>>> abs(-3.14)
3.14
>>> abs(4)
4
```

Συμβολοσειρές

Συμβολοσειρές

Για την αναπαράσταση χαρακτήρων η Python χρησιμοποιεί αντικείμενα τύπου `str` (string). Οι τιμές αντικειμένων τύπου `str` γράφονται είτε με απλά είτε με διπλά εισαγωγικά, όπως

```
'xyz',  "name",  'Peter Parker',  '123'
```

Η συνένωση δύο συμβολοσειρών επιτυγχάνεται με τον τελεστή `+`, για παράδειγμα,

```
>>> a = 'MEM'; type(a)
<class 'str'>
>>> b = '104'
>>> c = 'Programming I'
>>> a + b + ' ' + c
'MEM104 Programming I'
```

Συμβολοσειρές

ενώ με τον τελεστή * επιτυγχάνουμε παραγωγή αντιγράφων, με την έννοια

```
>>> 2*'ma'
'mama'
>>> 10*'-'
'----------'
>>> 'x'*3
'xxx'
>>> 2*'2'
'22'
```

```
>>> '2'*'2'
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'str'
```

Συμβολοσειρές

Οι συμβολοσειρές είναι ένας από τους πολλούς τύπους ακολουθιών (sequences) της Python για τις οποίες παρέχονται οι παρακάτω λειτουργίες:

Το *μήκος* μιας συμβολοσειράς μπορεί να βρεθεί με τη χρήση της συνάρτησης `len`

```
>>> len('abc')
3
>>> s = 'Maria'
>>> len(s)
5
```

Προσπέλαση με αριθμοδείκτη (indexing): για την εξαγωγή μεμονωμένων χαρακτήρων από μια συμβολοσειρά. Στην Python αρίθμηση των χαρακτήρων μιας συμβολοσειράς ξεκινάει από το μηδέν.

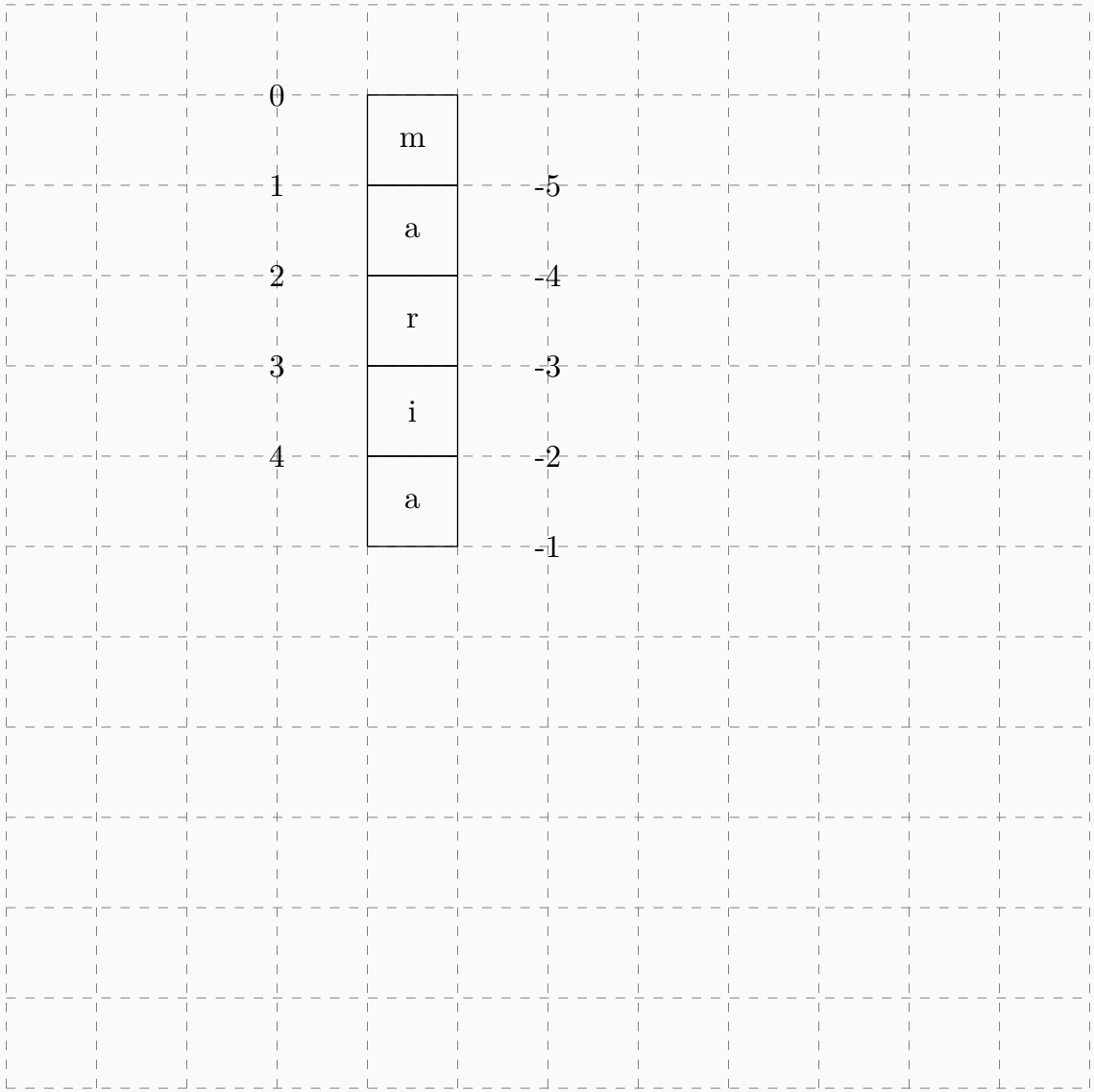
Συμβολοσειρές

```
>>> s = 'Maria'
>>> s[0]
'M'
>>> s[4]
'a'
>>> s[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> 'abc'[1]
'b'
```

Με χρήση αρνητικών αριθμών μπορούμε να δεικτοδοτήσουμε μια συμβολοσειρά από το τέλος της:

```
>>> 'abc'[-1]
'c'
```

Συμβολοσειρές



Συμβολοσειρές

Ο *τεμαχισμός* (slicing) χρησιμοποιείται για την εξαγωγή υπο-συμβολοσειρών (substrings) οποιουδήποτε μήκους:

Αν `s` είναι μια συμβολοσειρά, η παράσταση `s[i:j]` υποδηλώνει μια υπο-συμβολοσειρά του `s` που ξεκινά στον δείκτη θέσης `i` και τελειώνει στον δείκτη θέσης `j-1`.

```
>>> s = 'interdisciplinary'
>>> s[0:5]
'inter'
>>> s[5:9]
'disc'
>>> s[-6:-2]
'lina'
>>> s[2:-3]
'terdisciplin'
```

Συμβολοσειρές

Αν παραλειφθεί η τιμή πρίν την άνω και κάτω τελεία αυτή νοείται ως μηδέν. Αν παραλειφθεί η τιμή μετά την άνω και κάτω τελεία αυτή νοείται ως το μήκος της συμβολοσειράς. Έτσι,

```
>>> s = 'Maria'
>>> s[:2]
'Ma'
>>> s[1:]
'aria'
>>> s[:]
'Maria'
```

Συμβολοσειρές

Στην πραγματικότητα ο τεμαχισμός συμβολοσειρών είναι πολύ πιο ευέλικτος. Μπορούμε να γράψουμε

`s[start:end:step]`

για να αναφερθούμε στις θέσεις μεταξύ `start` και `end-1` με βήμα `step` της συμβολοσειράς `s`.

Για παράδειγμα,

```
>>> s = 'RedHotChiliPeppers'
>>> s[0:10:2]  # or s[:10:2]
'RdoCi'
```

Συμβολοσειρές

Επίσης,

```
>>> s = 'RedHotChiliPeppers'
>>> s[::2]
'RdoCiiepr'
>>> s[::-1]
'sreppePilihCtoHdeR'
>>> s[:]
'RedHotChiliPeppers'
>>> s[-1:-8:-1]
'sreppeP'
```

Συμβολοσειρές

Αν `s` είναι συμβολοσειρά και `i`, `j`, `step` είναι ακέραιοι, με `step` θετικό, τότε

```
>>> s[:]          # s[0], s[1], ..., s[-1]
>>> s[i:]         # s[i], s[i+1], ..., s[-1]
>>> s[:j]         # s[0], s[1], ..., s[j-1]
>>> s[i:j]        # s[i], s[i+1], ..., s[j-1]
>>> s[::step]     # s[0], s[step], ..., s[-1]
>>> s[i::step]    # s[i], s[i+step], ..., s[-1]
>>> s[:j:step]    # s[0], s[step], ..., s[j-1]
>>> s[i:j:step]   # s[i], s[i+step], ..., s[j-1]
```

Αν φυσικά `(j - i) % step != 0` τότε το τελευταίο στοιχείο θα είναι σε θέση πρίν την `j-1`.

Συμβολοσειρές

Στην περίπτωση που το βήμα είναι αρνητικό τότε η διάταξη των στοιχείων που επιστρέφονται είναι λίγο διαφορετική:

```
>>> s[::-step]     # s[-1], s[-1-step], ..., s[0]
>>> s[j::-step]    # s[j], s[j-step], ..., s[0]
>>> s[:i:-step]    # s[-1], s[-1-step], ..., s[i+1]
>>> s[i:j:-step]   # s[j], s[j-step], ..., s[i+1]
```

Συμβολοσειρές

Κάποιοι χαρακτήρες στην Python τυπώνονται χρησιμοποιώντας ως πρόθεμα την ανάποδη κάθετο. Για παράδειγμα:

```
>>> print('John said: \"I like Python!\"')
John said: "I like Python!"
>>> print('Tell John to:\n1. Come by at 10.\n2. Bring pizza.')
Tell John to:
1. Come by at 10.
2. Bring pizza.
```

Οι χαρακτήρες αυτοί ονομάζονται *χαρακτήρες διαφυγής*. Μερικοί ακόμα είναι οι '`\t`' (χαρακτήρας `tab` ή στηλοθέτης), ο χαρακτήρας '`\\`' για την εκτύπωση της ανάποδης καθέτου, και τα προθέματα '`\o`' και '`\x`' για την αναπαράσταση χαρακτήρων στο οκταδικό ή δεκαεξαδικό σύστημα.

Είσοδος δεδομένων από τον χρήστη

Είσοδος δεδομένων

Η λήψη δεδομένων στην Python επιτυγχάνεται με τη συνάρτηση `input`.

Η συνάρτηση `input` δέχεται ως όρισμα μια συμβολοσειρά την οποία εμφανίζει ως προτρεπτικό και περιμένει από τον χρήστη να πληκτρολογήσει κάτι και να πατήσει το πλήκτρο Enter. Ότι πληκτρολόγησει ο χρήστης επιστρέφεται ως συμβολοσειρά:

```
>>> name = input('What is your name? ')
What is your name?  John
>>> n = input('Type in an integer: ')
Type in an integer: 3
>>> type(n)
<class 'str'>
>>> 2*n
'33'
```

Είσοδος δεδομένων

Μπορούμε να μετατρέψουμε τη συμβολοσειρά `n` σε αντικείμενο τύπου `int` χρησιμοποιώντας τη σύνάρτηση `int()`:

```
>>> int(n)
3
>>> 4*int(n)
12
```

Όμοια, μπορούμε να μετατρέψουμε συμβολοσειρές σε αριθμούς κινητής υποδιαστολής `float` με χρήση της συνάρτησης `float()`:

```
>>> pi = '3.14'
>>> type('pi')
<class 'str'>
>>> 3*float(pi)
9.42
```