ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Δομή Δεδομένων

- Ο όρος δομή δεδομένων αναφέρεται σε ένα σύνολο δεδομένων τα οποία έχουν με κάποιο τρόπο ομαδοποιηθεί κάτω από ένα κοινό όνομα.
- Η δομή δεδομένων αναφέρεται στο σχήμα (τον τρόπο) οργάνωσης των δεδομένων που χρησιμοποιεί το πρόγραμμά μας. Οι βασικές δομές δεδομένων της Python που θα εξετάσουμε στο κεφάλαιο αυτό είναι **οι συμβολοσειρές, οι λίστες, οι πλειάδες, τα σύνολα και τα λεξικά.**

Στατικές & Δυναμικές Δομές Δεδομένων

- Ο διαχωρισμός των δομών σε **στατικές** και **δυναμικές** έχει να κάνει με το κατά πόσο ο χώρος που καταλαμβάνουν τα δεδομένα στη μνήμη είναι σταθερός (στατική δομή) ή μπορεί να μεταβάλλεται (δυναμική δομή).
- Στην περίπτωση της **στατικής δομής** **το μέγεθος που αυτή θα καταλαμβάνει στη μνήμη καθορίζεται κατά τη συγγραφή του προγράμματος από τον ίδιο τον προγραμματιστή.** Όταν το πρόγραμμα ξεκινήσει να εκτελείται δεσμεύεται στη μνήμη ο χώρος που έχει καθοριστεί. Το μέγεθος των στατικών δομών παραμένει σταθερό κατά την εκτέλεση του προγράμματος, αφού δεν μπορούμε να αφαιρέσουμε ούτε να προσθέσουμε αντικείμενα στις δομές αυτές. Στην Python στατικές δομές δεδομένων είναι οι **πλειάδες** (tuples), όπως θα δούμε παρακάτω.
- Στην περίπτωση της **δυναμικής δομή**ς το πρόγραμμα μπορεί να ζητήσει από το λειτουργικό σύστημα όση μνήμη απαιτείται για τη δημιουργία της δομής δεδομένων, κατά την εκτέλεση του προγράμματος. **Οι δυναμικές δομές δεδομένων μπορούν να μεταβάλουν το μέγεθός τους, προσθέτοντας ή αφαιρώντας αντικείμενα**. Η πιο γνωστή δυναμική δομή δεδομένων είναι η **λίστα** η οποία είναι και η βασική δομή δεδομένων της Python. Με βασικό δομικό λίθο τη λίστα μπορούμε να υλοποιήσουμε όποια σύνθετη δομή δεδομένων θέλουμε, όπως η στοίβα, η ουρά, το δέντρο κ.λπ.

Στατικές & Δυναμικές Δομές Δεδομένων

- Έναν πίνακα, μπορούμε, να τον υλοποιούμε σε κάποιες γλώσσες προγραμματισμού και ως στατική δομή δεδομένων.

- Ουσιαστικά, η **λίστα** της Python δεν είναι τίποτα παραπάνω από ένα **δυναμικό πίνακα**, δηλαδή έναν πίνακα του οποίου το μέγεθος μπορεί να αυξομειώνεται κατά την εκτέλεση του προγράμματος.

- Ενδιαφέρον παρουσιάζει ο τύπος της συμβολοσειράς (str) ο οποίος επιτρέπει τη δυναμική δέσμευση μνήμης αλλά όχι τη μεταβολή του μεγέθους της.

ΣΥΜΒΟΛΟΣΕΙΡΑ

Συμβολοσειρά (str)

- Η συμβολοσειρά είναι μια ακολουθία από χαρακτήρες και μπορεί να αποτελείται από περισσότερες από μία λέξεις, με τις λέξεις να μπορούν να είναι στην Ελληνική γλώσσα, στην Αγγλική ή σε κάθε γλώσσα που υποστηρίζεται από το πρότυπο Unicode. Μια συμβολοσειρά την ορίζουμε με εισαγωγικά αμφίπλευρα, μονά ή διπλά.

- a="Καλημέρα" *// Εκχώρηση στη μεταβλητή a της συμβολοσειράς "Καλημέρα"*
- b="ηλιόλουστη μέρα"
- z=a+b
- print z *// Καλημέραηλιόλουστη μέρα*
- d= a*2
- print d *// ΚαλημέραΚαλημέρα*

Συμβολοσειρά (str)

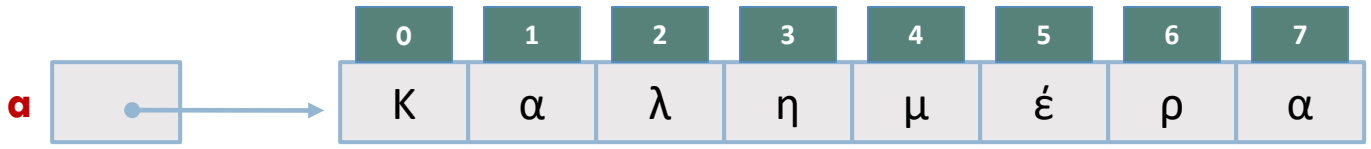
- Η μορφή της μεταβλητής a που φυλά τη συμβολοσειρά "Καλημέρα", στην μνήμη

a							
0	1	2	3	4	5	6	7
Κ	α	λ	η	μ	έ	ρ	α

- Η συνάρτηση **len(a)** επιστρέφει το μήκος της συμβολοσειράς (εδώ **8**)

Συμβολοσειρά (str)

- Η μορφή της μεταβλητής a που φυλά τη συμβολοσειρά "Καλημέρα", στην μνήμη



- Η συνάρτηση **len(a)** επιστρέφει το μήκος της συμβολοσειράς (εδώ **8**)

Συμβολοσειρά (str)

- Η μορφή της μεταβλητής b που φυλά τη συμβολοσειρά "ηλιόλουστη μέρα", στην μνήμη

b

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
η	λ	ι	ό	λ	ο	υ	σ	τ	η		μ	έ	ρ	α

- Η συνάρτηση **len(b)** επιστρέφει το μήκος της συμβολοσειράς (εδώ **15**)

Συμβολοσειρά (str)

- Η έκφραση **b[1] επιστρέφει τον χαρακτήρα που βρίσκεται στην θέση 1** δηλ. το **λ**

b

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
η	λ	ι	ό	λ	ο	υ	σ	τ	η		μ	έ	ρ	α

Συμβολοσειρά (str)

- Η έκφραση **b[11:15] επιστρέφει όλους τους χαρακτήρες ξεκινώντας από την θέση 11 και σταματώντας στην 14** δηλ. **μέρα**

b

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
η	λ	ι	ό	λ	ο	υ	σ	τ	η		μ	έ	ρ	α

Συμβολοσειρά (str)

- Η έκφραση **b[0:10] επιστρέφει όλους τους χαρακτήρες ξεκινώντας από την θέση 0 και σταματώντας στην 9** δηλ. **μέρα**

b														
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
η	λ	ι	ό	λ	ο	υ	σ	τ	η		μ	έ	ρ	α

Συμβολοσειρά (str)

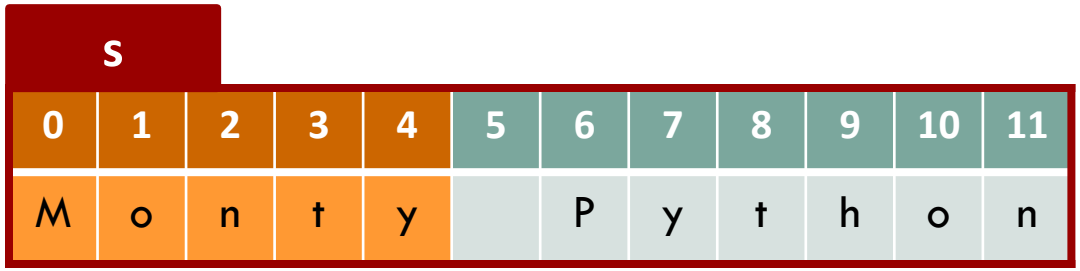
- Μπορούμε να πάρουμε ένα τμήμα της συμβολοσειράς με χρήση του τελεστή **":"**. **Όταν γράφουμε a[*αρχή:τέλος]* *επιστρέφεται το μέρος της συμβολοσειράς που ξεκινάει από τον χαρακτήρα στη θέση αρχή μέχρι τη θέση τέλος, χωρίς όμως να συμπεριλαμβάνεται ο χαρακτήρας της θέσης τέλος.*** Στη βιβλιογραφία το τμήμα αυτό της συμβολοσειράς αναφέρεται και ως φέτα (slice).

b

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
η	λ	ι	ό	λ	ο	υ	σ	τ	η		μ	έ	ρ	α

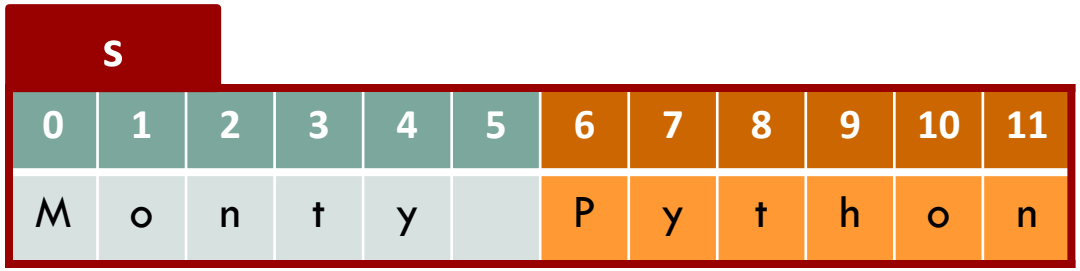
Συμβολοσειρά (str)


```
>>> s[0:5]
'Monty'
```



Συμβολοσειρά (str)

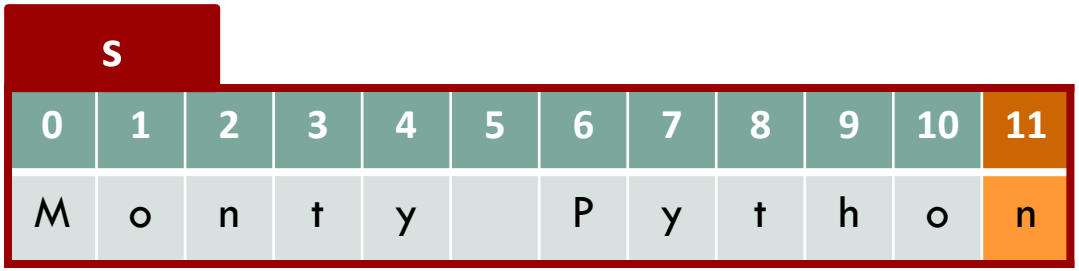

```
>>> s[6:len(s)]
'Python'
```



- len(s) ισούται με 12

Συμβολοσειρά (str)


```
>>> s[len(s)-1]
'n'
```



- len(s) – 1 ισούται με 12-1 άρα 11

Συμβολοσειρά (str)


```
>>> s[-1]
'n'
```

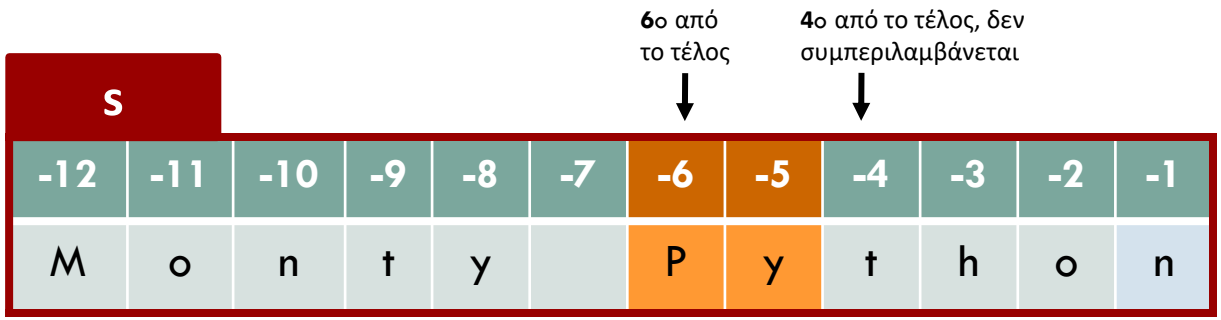
s

-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
M	o	n	t	y		P	y	t	h	o	n

- Μπορούμε να κάνουμε χρήση αρνητικής αρίθμησης για να δηλώσουμε ότι ξεκινάμε από το τέλος της συμβολοσειράς . s[-1] σημαίνει το πρώτο σύμβολο από το τέλος.

Συμβολοσειρά (str)

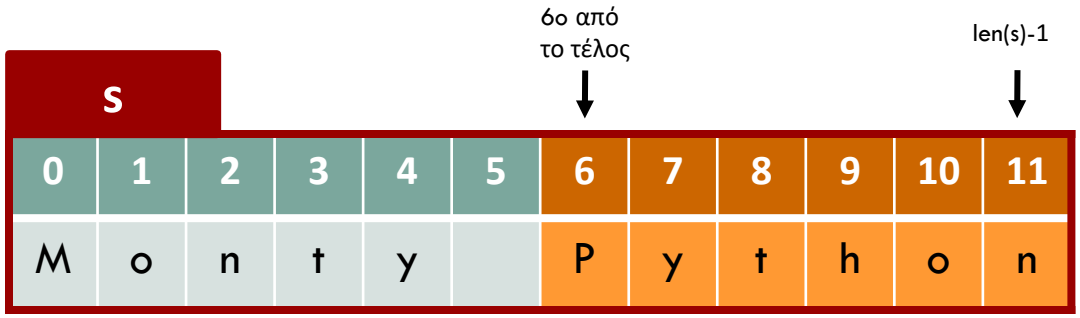

```
>>> s[-6:-4]
'Py'
```



Συμβολοσειρά (str)

>>> s[-6:len(s)] → s[-6 : 12]

'Python'

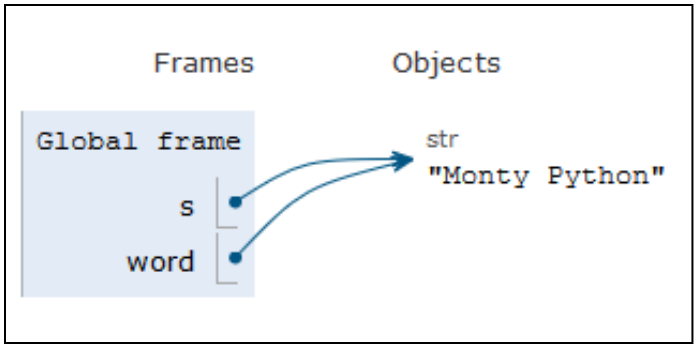


Συμβολοσειρά (str)

- Ένα θέμα που πρέπει να προσέξουμε, είναι η χρήση του τελεστή εκχώρησης στις συμβολοσειρές.

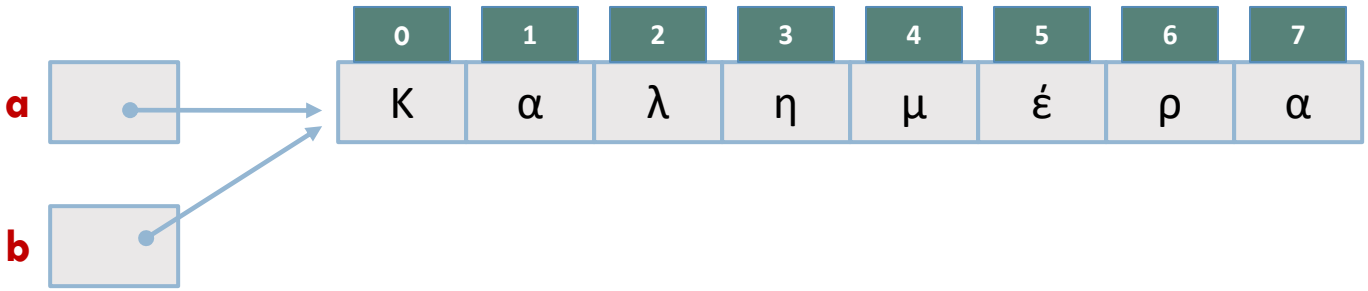
word=s
και οι δύο μεταβλητές αναφέρονται στο ίδιο αντικείμενο

- Παραπάνω οι δύο μεταβλητές word, s αναφέρονται στην ίδια περιοχή μνήμης.



Συμβολοσειρά (str)

b=a



Τελεστής in

- Ελέγχει αν ένας χαρακτήρας ή σύνολο χαρακτήρων περιέχονται σε μια συμβολοσειρά

- **>>> "y" in s**
- **True**
- **>>> "Monty" in s[0:5]**
- **True**

Συγκριτικοί Τελεστές

- Οι γνωστοί **συγκριτικοί τελεστές**, οι οποίοι **συγκρίνουν με βάση τη λεξικογραφική διάταξη των χαρακτήρων**, ισχύουν και στις συμβολοσειρές.

- **>>> s[1]>s[2]**
- **True**
- **>>> s[0:5]<s[6:12]** *// συγκρίνεται το πρώτο γράμμα κάθε λέξης*
- **True**

s											
0	1	2	3	4	5	6	7	8	9	10	11
M	o	n	t	y		P	y	t	h	o	n

Συγκριτικοί Τελεστές

- s="Monty Mython"
- >>> s[0:5]<s[6:12] // *συγκρίνονται τα πρώτα γράμματα κάθε λέξης και αν ίδια περνά να συγκρίνει τα δεύτερα κοκ*
- True

s

0	1	2	3	4	5	6	7	8	9	10	11
M	o	n	t	y		M	y	t	h	o	n

Μέθοδοι συμβολοσειρών

- Κάθε συμβολοσειρά είναι ένα **αντικείμενο** το οποίο διαθέτει και κάποιες ενσωματωμένες συναρτήσεις (**μεθόδους**). Οι μέθοδοι είναι ίδιοι για κάθε αντικείμενο που ανήκει στον τύπο str.

- Για να δω όλες τις μεθόδους που υποστηρίζει ο τύπος str χρησιμοποιώ την επόμενη έκφραση στον Interpreter.

- **>>> dir(str)**

Μέθοδος upper()

- **Μετατρέπει τα μικρά σε ΚΕΦΑΛΑΙΑ**

- >>> s="python"
- >>> **s.upper()**
- 'PYTHON'
- >>> print s
- 'python'

- **ΠΡΟΣΟΧΗ! Το πραγματικό περιεχόμενο της s δεν αλλάζει**

Μέθοδος capitalize()

- Μετατρέπει τον **πρώτο χαρακτήρα** της συμβολοσειράς **σε ΚΕΦΑΛΑΙΟ** εφόσον αυτός είναι γράμμα

- >>> s="monty python"
- >>> **s.capitalize()**
- 'Monty python'
- >>> print s
- ' monty python'

- **ΠΡΟΣΟΧΗ! Το πραγματικό περιεχόμενο της s δεν αλλάζει**

Μέθοδος lower()

- Μετατρέπει τα ΚΕΦΑΛΑΙΑ σε μικρά

- >>> s=“MONTY PYTHON"
- >>> **s.lower()**
- ‘monty python'
- >>> print s
- ‘MONTY PYTHON '

- **ΠΡΟΣΟΧΗ! Το πραγματικό περιεχόμενο της s δεν αλλάζει**

Άσκηση

- >>> word = “monty”
- >>> “python”.upper()
- ----
- >>> ‘a’ **in ‘python’**
- ----
- >>> ‘y’ **in ‘python’**
- ----
- >>> word == ‘monty’
- ----
- >>> ‘antonis’ > ‘antonia’
- ----
- >>> 1000 < 2
- ----
- >>> ‘1000’ < ‘2’
- ----

Άσκηση-Λύση

- >>> word = “monty”
- >>> “python”.upper()
- ‘PYTHON’
- >>> ‘a’ **in ‘python’**
- False
- >>> ‘y’ **in ‘python’**
- True
- >>> word == ‘monty’
- True
- >>> ‘antonis’ > ‘antonia’
- True
- >>> 1000 < 2
- False
- >>> ‘1000’ < ‘2’
- True

ΠΡΟΣΟΧΗ!

- **Οι συμβολοσειρές δεν είναι τροποποιήσιμες. Δηλαδή δεν μπορούμε να αλλάξουμε μέρος της συμβολοσειράς.**
- Για παράδειγμα, η εντολή **word[0] = 'p'** θα μας επιστρέψει **λάθος**, αφού δεν μπορεί να εκτελεστεί.
- Οποιαδήποτε επεξεργασία θέλουμε να κάνουμε σε συμβολοσειρές γίνεται με χρήση, του τελεστή **: για αποκοπή του μέρους που θέλουμε και του τελεστή + για συνένωση.**

Άσκηση

- Στην πιο κάτω γραμματοσειρά
- s = “Brad Pett”
- Να αλλαχθεί ο 6ος χαρακτήρας από “e” se “i”

Λύση

- Στην πιο κάτω γραμματοσειρά
- s = “Brad Pett”
- Να αλλαχθεί ο 6ος χαρακτήρας από “e” se “i”
- s = s[0:6] + “i” + s[7:len(s)]

Άσκηση

- Παρακάτω δίνονται μερικά παραδείγματα εντολών με αλφαριθμητικά στον διερμηνευτή της Python:

```
>>> word = “PYTHON”
>>> len(word)
---
>>> print word[5]+word[0]
---
>>> str(28) == ‘28’
---
>>> print  int(‘496’) + 4
---
```

Άσκηση

- Παρακάτω δίνονται μερικά παραδείγματα εντολών με αλφαριθμητικά στον διερμηνευτή της Python:

```
>>> word = “PYTHON”
>>> len(word)
6
>>> print word[5]+word[0]
NP
>>> str(28) == ‘28’
True
>>> print  int(‘496’) + 4
500
```

Συμβολοσειρές

- Η **συνάρτηση** **len** επιστρέφει το μήκος, δηλαδή το πλήθος των χαρακτήρων του αλφαριθμητικού
- Ο **τελεστής** **+** όταν εφαρμόζεται σε αντικείμενα τύπου string, έχει σαν αποτέλεσμα τη συνένωσή τους σε μια συμβολοσειρά
- Η **συνάρτηση** **str** μετατρέπει μια τιμή σε συμβολοσειρά
- Με τη **συνάρτηση** **int** μπορούμε να μετατρέψουμε ένα αλφαριθμητικό στον ακέραιο αριθμό που αναπαριστά.

Τελεστής in – Έλεγχος ύπαρξης

- Τι θα εμφανίσουν οι πιο κάτω εντολές;

```
>>> "Py" in "Python"
---
>>> "a" in "Python"
---
>>> "a" not in "Python"
---
```

Τελεστής in – Έλεγχος ύπαρξης


```
>>> "Py" in "Python"
True
>>> "a" in "Python"
False
>>> "a" not in "Python"
True
```

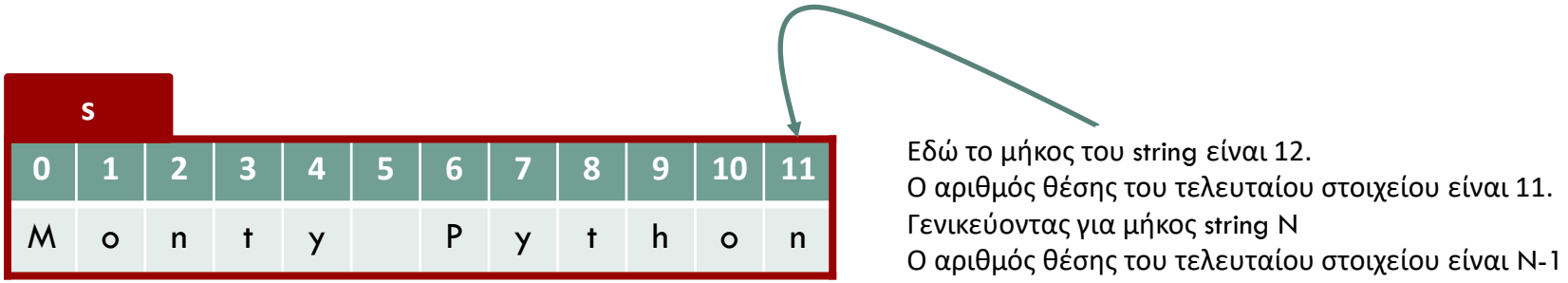
Εφαρμογή 1- Συμβολοσειρές

- Να γραφτεί πρόγραμμα που διαβάζει μια συμβολοσειρά που δίνεται από το πληκτρολόγιο και στη συνέχεια εντοπίζει πόσες φορές περιέχεται το γράμμα "t" μέσα σ' αυτή.

Εφαρμογή 1- Συμβολοσειρές

Έστω δίνεται ένα τυχαίο string π.χ. “Monty Python”. Ο υπολογιστής δεν μπορεί να γνωρίζει το μήκος του σε χαρακτήρες άρα πρέπει να καλέσω τη συνάρτηση len(s) για να το βρω.

N=len(s)



Εφαρμογή 1- Συμβολοσειρές

Στη συνέχεια ο υπολογιστής πρέπει να διατρέξει τους χαρακτήρες του string ξεκινώντας από την αρχή θέση 0 έως το τέλος θέση N-1 και να κάνει τις πιο κάτω ενέργειες. Να διαβάσει τον πρώτο χαρακτήρα και αν αυτός είναι το t να αυξήσει έναν μετρητή k.

```
if s[0]=="t" :
    k=k+1
```

Μετά θα πρέπει να διαβάσει τον δεύτερο χαρακτήρα και αν αυτός είναι το t να αυξήσει τον μετρητή k.

```
if s[1]=="t" :
    k=k+1
```

Μετά θα πρέπει να διαβάσει τον τρίτο χαρακτήρα και αν αυτός είναι το t να αυξήσει τον μετρητή k.

```
if s[2]=="t" :
    k=k+1
```

κ.ο.κ.

```
if s[N-1]=="t" :
    k=k+1
```

Εφαρμογή 1- Συμβολοσειρές


```
 i
 ↓
 0    if s[0]=="t" :
          k=k+1

 1    if s[1]=="t" :
          k=k+1

 2    if s[2]=="t" :
          k=k+1

       .........

N-1   if s[N-1]=="t" :
          k=k+1
```

Η εντολή ελέγχου πρέπει να εκτελεστεί τόσες φορές όσοι οι χαρακτήρες του string, δηλ. **Ν φορές**.

Θα βάλω τον μετρητή i να μετρήσει από το 0 έως το Ν-1. Άρα να διατρέξει τη λίστα τιμών [0,1,2,…,N-1]

Τη λίστα θα παράγω καλώντας την συνάρτηση range(N)

Εφαρμογή 1

Εφαρμογή 1

```
k=0
s=raw_input("Δώσε συμβολοσειρά:")
N=len(s)
for  i  in range(N) :
   if  s[i]== "t" :
       k=k+1
print k
```

Εφαρμογή 2- Συμβολοσειρές

- Να γραφεί συνάρτηση η οποία παραλαμβάνει από το κυρίως πρόγραμμα μια συμβολοσειρά και επιστρέφει πίσω το πλήθος των "t" που αυτή περιέχει.

Εφαρμογή 2

Εφαρμογή 2

```
def counter(w):
    k=0
    N=len(w)
    for i in range(N) :
        if w[i]=="t" :
            k=k+1
    return k



#Main Program
s=raw_input("Δώσε συμβολοσειρά:")
print "Η συμβολοσειρά έχει",counter(s),"χαρακτήρες t"
```

Πολυμορφική συμπεριφορά

- Ένα εξαιρετικά ενδιαφέρον χαρακτηριστικό της Python είναι η πολυμορφική συμπεριφορά της, η οποία φαίνεται στο παρακάτω παράδειγμα:

```
>>> def times( a, b ):
        return a * b
```

```
>>> times( 2, 3 )
6
>>> times( "python#", 3 )
python#python#python#
```