ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Γλώσσες Προγραμματισμού

Λίστες (Lists) - Python

Χαρά Πρασσά, chprassa@gmail.com

- Μία λίστα είναι ένα αντικείμενο το οποίο περιέχει πολλαπλά δεδομένα τα οποία δεν είναι αναγκαίο να έχουν το ίδιο data type (μπορεί δηλαδή να περιέχει ανομοιογενή στοιχεία).
- Μία λίστα είναι ένας δυναμικός τύπος δεδομένων. Αυτό σημαίνει ότι μπορούμε να προσθέσουμε ή να αφαιρέσουμε στοιχεία σε μία λίστα.
- Η λίστα είναι ένα διατεταγμένο σύνολο δεδομένων, η τάξη/αρίθμηση των στοιχείων της αρχίζει από το 0 (μηδέν). Κάθε στοιχείο μίας λίστας μπορεί να προσπελαστεί μέσω ενός δείκτη που καθορίζει τη θέση του στη λίστα.
- Για να ορίσουμε μια λίστα στην Python ακολουθούμε τα εξής εύκολα βήματα: μέσα σε square brackets ([ ]) δηλώνουμε τα στοιχεία (ή elements όπως επίσημα ονομάζονται) χωρισμένα με κόμμα.

Για παράδειγμα:

```
grades=[9,8,6]
names=['Μαρία','Αντώνης','Δήμητρα']
names_grades=['Μαρία',9,'Αντώνης',8,'Δήμητρα',6]
print(grades)
print(names)
print(names_grades)
```

Η εκτέλεση του προγράμματος θα δώσει ως αποτέλεσμα:

```
[9, 8, 6]
['Μαρία', 'Αντώνης', 'Δήμητρα']
['Μαρία', 9, 'Αντώνης', 8, 'Δήμητρα', 6]
```

Παράδειγμα λίστας που περιέχει ανομοιογενή στοιχεία:

```
lista=[1,2,3.14,'Python',True,False]
print(type(lista))
```

Η εκτέλεση του προγράμματος θα δώσει ως αποτέλεσμα:

<class 'list'>

- Δημιουργία κενής λίστας

```
empty_list=list()
print(empty_list)
```

Η εκτέλεση του προγράμματος θα δώσει ως αποτέλεσμα:
[]

- Η συνάρτηση list() μπορεί να δεχτεί κάποιο όρισμα το οποίο όμως θα πρέπει να είναι επαναληπτικό όπως μια συμβολοσειρά.

```
new_list=list('Python')
print(new_list)
```

Η εκτέλεση του προγράμματος θα δώσει ως αποτέλεσμα:

['P', 'y', 't', 'h', 'o', 'n']

- Για να αποκτήσουμε πρόσβαση σε οποιοδήποτε στοιχείο της λίστας δεν έχουμε παρά να γράψουμε το όνομα της λίστας και αμέσως μετά μέσα σε square brackets ([ ]) τον αριθμό του index.

Για παράδειγμα:

```
names_grades=['Μαρία',9,'Αντώνης',8,'Δήμητρα',6]
print(names_grades[2])
```

Output:

⇩

Αντώνης

- Επιτρέπεται επίσης να μπορούμε να μετράμε από το τέλος της λίστας προς την αρχή. Ο αριθμός -1 μας δίνει πρόσβαση στο τελευταίο στοιχείο της λίστας. Ο αριθμός -2 μας δίνει πρόσβαση στον προτελευταίο αριθμό της λίστας κτλ.

Για παράδειγμα:

```
names_grades=['Μαρία',9,'Αντώνης',8,'Δήμητρα',6]
print(names_grades[-2])
```

Output:

⇩

Δήμητρα

- Μια λίστα μπορεί επίσης να περιέχει μια ή και περισσότερες λίστες σαν στοιχεία. Για να αποκτήσουμε πρόσβαση στο στοιχείο που είναι λίστα απλά προσθέτουμε ένα ακόμα square bracket ([ ]) μετά το πρώτο. Το πρώτο bracket ορίζει το στοιχείο στη λίστα. Το δεύτερο bracket ορίζει στο συγκεκριμένο στοιχείο (που είναι λίστα) σε ποιο στοιχείο θέλουμε να αποκτήσουμε πρόσβαση.

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
print(names_grades[1][0])
```

Output:

9

- Χρησιμοποιώντας το δείκτη μπορούμε επίσης να αλλάξουμε την τιμή που υπάρχει στο συγκεκριμένο στοιχείο.

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
names_grades[0]='Γιώργος'
print(names_grades)
```

Output:

['Γιώργος', [9, 2], 'Αντώνης', 8, 'Δήμητρα', 6]

- Ο πιο απλός τρόπος για να διαβάσουμε μια ομάδα από στοιχεία μιας λίστας είναι: list[index1:index2]. Ο πρώτος αριθμός index1 δηλώνει τη θέση από την οποία θα αρχίζουμε να διαβάζουμε τα στοιχεία της λίστας ενώ το index2 δηλώνει σε ποια θέση θα σταματήσουμε χωρίς όμως να περιλαμβάνει αυτό τον αριθμό. Οι δύο δηλώσεις των index χωρίζονται από ένα colon (:).

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
print(names_grades[1:4])
```

Output

```
[[9, 2], 'Αντώνης', 8]
```

- Μπορούμε επίσης να χρησιμοποιήσουμε μόνο ένα από τα δύο indexes. Αφήνοντας το index1 και αφαιρώντας το index2 σημαίνει ότι θα διαβάσουμε τη λίστα ξεκινώντας από το index1 μέχρι το τέλος της λίστας. Αντίστοιχα, αν κρατήσουμε μόνο το index2 σημαίνει ότι επιθυμούμε να διαβάσουμε από την αρχή της λίστας μέχρι το σημείο που ορίζει το index2.

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
print(names_grades[:3])
print(names_grades[2:])
```

← αριστερά της : κενό, υπονοείται η αρχή της λίστας

← δεξιά της : κενό, υπονοείται το τέλος της λίστας

Output

```
['Μαρία', [9, 2], 'Αντώνης']
['Αντώνης', 8, 'Δήμητρα', 6]
```

- Υπάρχει μια τρίτη μεταβλητή που μπορούμε να προσθέσουμε όταν διαβάζουμε μια ομάδα από στοιχεία μιας λίστας, το βήμα (step). Με το βήμα δηλώνουμε τις τιμές που δεν θα συμπεριλάβουμε στο τελικό αποτέλεσμα. Αν για παράδειγμα ορίσουμε σαν βήμα τον αριθμό 2 τότε δηλώνουμε ότι θα διαβάσουμε κάθε δύο στοιχεία από το εύρος που έχουμε ορίσει.

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
print(names_grades[1:5:2])
```

Output:

⇩

```
[[9, 2], 8]
```

- *Για να διαβάσουμε όλη τη λίστα,*

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
print(names_grades[:])
```

Output:

⇩

```
['Μαρία', [9, 2], 'Αντώνης', 8, 'Δήμητρα', 6]
```

- *Για να διαβάσουμε όλες τις ζυγές θέσεις της λίστας,*

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
print(names_grades[1::2])
```

Output:

⇩

[[9, 2], 8, 6]

- *Για να διαβάσουμε τη λίστα με ανεστραμμένα τα στοιχεία της,*

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
print(names_grades[::-1])
```

Output:

⇩

[6, 'Δήμητρα', 8, 'Αντώνης', [9, 2], 'Μαρία']

- Μπορούμε να αντιγράψουμε μια λίστα; Ο λάθος τρόπος να αντιγράψετε μια λίστα είναι να γράψετε list1 = list2. Με αυτό τον τρόπο απλά έχετε δημιουργήσει δύο ονόματα μεταβλητών που αναφέρονται στο ίδιο ακριβώς σημείο της μνήμης στο οποίο βρίσκονται τα δεδομένα. Το αποτέλεσμα είναι ότι αν διαγράψετε ένα στοιχείο από το list1 τότε επηρεάζεται και το list2.

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
names_grades2=names_grades
names_grades2[0]='Γιάννης'
print(names_grades)
print(names_grades2)
```

Η λίστα είναι μια μεταλλάξιμη δομή, για αυτό εδώ δεν έχουμε 2 λίστες αλλά 1 λίστα η οποία είναι προσβάσιμη από 2 μεταβλητές

Output:

⇩

```
['Γιάννης', [9, 2], 'Αντώνης', 8, 'Δήμητρα', 6]
['Γιάννης', [9, 2], 'Αντώνης', 8, 'Δήμητρα', 6]
```

- Ο σωστός τρόπος αντιγραφής μιας λίστας είναι να γράψουμε δύο colon (::) χωρίς να δηλώσουμε κανέναν αριθμό. Το αποτέλεσμα είναι να αντιγράψουμε όλα τα στοιχεία από την αρχή μέχρι το τέλος σε μια νέα λίστα.

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
names_grades2=names_grades[::]
names_grades[0]='Γιάννης'
print(names_grades)
print(names_grades2)
```

Output:

```
['Γιάννης', [9, 2], 'Αντώνης', 8, 'Δήμητρα', 6]
['Μαρία', [9, 2], 'Αντώνης', 8, 'Δήμητρα', 6]
```

- Τομή ονομάζεται η δυνατότητα που έχουμε να πάρουμε κάποια τμήματα της λίστας, να πάρουμε δηλαδή μια υπολίστα αναφερόμενοι σε συγκεκριμένες θέσεις τις αρχικής λίστας.
- Υπάρχει η δυνατότητα να αναθέσω νέες τιμές σε στοιχεία μιας λίστας αλλάζοντας τη δομή της.

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
names_grades[0:2]=['Ελένη',[10,1]]
print(names_grades)
```

Output:

['Ελένη', [10, 1], 'Αντώνης', 8, 'Δήμητρα', 6]

- Υπάρχει η δυνατότητα να διαγράψω τιμές από τα στοιχεία μιας λίστας αλλάζοντας τη δομή της.

Για παράδειγμα:

```
names_grades=['Μαρία',[9,2],'Αντώνης',8,'Δήμητρα',6]
names_grades[0:3]=[]
print(names_grades)
```

Output:

[8, 'Δήμητρα', 6]

- Η προσπέλαση όλων των στοιχείων της λίστας επιτυγχάνεται με τον τελεστή **in** και με χρήση της **for**.

Παράδειγμα:

Έστω η λίστα students

students = ['Μαρία', 'Μανώλης', 'Δημήτρης', 'Αγγελική', 'Πέτρος']

Για να προσπελάσουμε όλα τα στοιχεία της λίστας:

```
students = ['Μαρία', 'Μανώλης', 'Δημήτρης', 'Αγγελική', 'Πέτρος']
for name in students:
   print(name)
```

Output:

Μαρία
Μανώλης
Δημήτρης
Αγγελική
Πέτρος

- Ο έλεγχος ύπαρξης ενός στοιχείου σε μία λίστα επιτυγχάνεται με τον τελεστή **in** και με χρήση της **if**.

Παράδειγμα:

Έστω η λίστα students

students = ['Μαρία', 'Μανώλης', 'Δημήτρης', 'Αγγελική', 'Πέτρος']

Ζητείται ο χρήστης να δίδει ένα όνομα και το πρόγραμμα να ελέγχει αν το όνομα αυτό υπάρχει στη λίστα εμφανίζοντας την τάξη του στη λίστα. Το πρόγραμμα θα έχει την παρακάτω μορφή:

```
students = ['Μαρία', 'Μανώλης', 'Δημήτρης', 'Αγγελική', 'Πέτρος']
onoma=input(' Δώσε το όνομα ')
if onoma in students:
   print('το όνομα ',onoma)
   print('βρίσκεται στη θέση ', students.index(onoma))
else:
   print('δεν υπάρχει αυτό το όνομα')
```

- Ο έλεγχος για το αν μια λίστα είναι άδεια γίνεται με τον παρακάτω κώδικα:

```
students = ['Μαρία', 'Μανώλης', 'Δημήτρης', 'Αγγελική', 'Πέτρος']
if not students:
   print('Η λίστα είναι άδεια')
else:
   print('Η λίστα δεν είναι άδεια')
```

- ***.append***

Με τη μέθοδο **append** επιτυγχάνεται εισαγωγή ενός στοιχείου σε μία υπάρχουσα λίστα. Το στοιχείο που θα εισαχθεί θα τοποθετηθεί στο τέλος της λίστας.

Σύνταξη:

lista.append(στοιχείο)

Παράδειγμα:

Έστω η λίστα students

students = ['Μαρία', 'Μανώλης', 'Δημήτρης', 'Αγγελική', 'Πέτρος']

Μετά την εκτέλεσης της: students.append('Ελένη') η λίστα θα είναι η:

['Μαρία', 'Μανώλης', 'Δημήτρης', 'Αγγελική', 'Πέτρος', 'Ελένη']

- ***.insert***

Η μέθοδος **insert** δέχεται δύο ορίσματα. Το πρώτο όρισμα ορίζει την τάξη (θέση) μέσα στη λίστα ενώ το δεύτερο όρισμα είναι το προς εισαγωγή στοιχείο.

Σύνταξη:

lista.insert(δεικτης, στοιχείο)

Παράδειγμα:

Η εντολή students.insert(3, 'Γιώργος') θα εισάγει το στοιχείο 'Γιώργος' στην 4η θέση (τάξη 3) και το περιεχόμενο της λίστας θα είναι:
['Μαρία', 'Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']

- ***.remove***

Με τη μέθοδο **remove** επιτυγχάνεται διαγραφή ενός στοιχείου σε μία υπάρχουσα λίστα. Εάν το στοιχείο υπάρχει περισσότερες από μία φορές διαγράφει το πρώτο που θα βρει. Εάν το στοιχείο δεν υπάρχει δημιουργείται λάθος εκτέλεσης του προγράμματος.

Σύνταξη:

lista.remove(στοιχείο)

Παράδειγμα:

Η εντολή students.remove('Μαρία') θα διαγράψει το στοιχείο με όνομα 'Μαρία' και το περιεχόμενο της λίστας θα είναι:

['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']

- *Αν το στοιχείο 'Μαρία' υπήρχε 2 φορές μέσα στη λίστα πως θα γινόταν η διαγραφή;*

```
students=['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Μαρία', 'Αγγελική', 'Πέτρος', 'Ελένη','Μαρία']
for student in students:
   if student=='Μαρία':
      students.remove('Μαρία')
print(students)
```

Output

['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']

- ***.pop***

Με τη μέθοδο **pop** επιτυγχάνεται διαγραφή ενός στοιχείου που βρίσκεται σε μια συγκεκριμένη θέση μέσα στη λίστα.

Σύνταξη:

lista.pop(index)

Παράδειγμα:

Η εντολή students.pop(0)θα διαγράψει το στοιχείο με όνομα 'Μαρία' και το περιεχόμενο της λίστας θα είναι:

['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']

- ***.index***

Η μέθοδος **index** δέχεται ως όρισμα ένα στοιχείο μιας λίστας και επιστρέφει την τάξη του στη λίστα.

Σύνταξη:

lista.index(στοιχείο)

Παράδειγμα:

Η εντολή print(students.index('Γιώργος')) θα εμφανίσει τον αριθμό 2. Δεδομένου ότι η τάξη του πρώτου στοιχείου είναι το 0 (μηδέν)

- ***.count***

Η μέθοδος count επιστρέφει τον αριθμό των εμφανίσεων ενός στοιχείου σε μια λίστα.

Σύνταξη:

lista.count(στοιχείο)

```
students=['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη','Δημήτρης']
print(students.count('Δημήτρης'))
```

Output

2

- ***.sort***

Η μέθοδος **sort** ταξινομεί μια λίστα.

Σύνταξη:

lista.sort() # ταξινόμηση αύξουσα
lista.sort(reverse=True) # ταξινόμηση φθίνουσα

```
students=['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']
students.sort()
print(students)
```

Output

['Αγγελική', 'Γιώργος', 'Δημήτρης', 'Ελένη', 'Μανώλης', 'Πέτρος']

```
students.sort(reverse=True)
print(students)
```

Output

['Πέτρος', 'Μανώλης', 'Ελένη', 'Δημήτρης', 'Γιώργος', 'Αγγελική']

- ***sorted( )***

Η sorted( ) συνάρτηση επιστρέφει ένα αντίγραφο της λίστας (δεν αλλάζει η αρχική λίστα) με τα ονόματα ή τους αριθμούς ταξινομημένα αλφαβητικά ή αριθμητικά. Αν δεν θέλετε να επηρεάσετε την αρχική λίστα, αυτή η συνάρτηση είναι η καλύτερη επιλογή.

Παράδειγμα:

```
students=['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']
sorted_students=sorted(students)
print(students)
print(sorted_students)
```

Output

```
['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']
['Αγγελική', 'Γιώργος', 'Δημήτρης', 'Ελένη', 'Μανώλης', 'Πέτρος']
```

```
sorted_students=sorted(students,reverse=True)
print(students)
print(sorted_students)
```

Output

```
['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']
['Πέτρος', 'Μανώλης', 'Ελένη', 'Δημήτρης', 'Γιώργος', 'Αγγελική']
```

- ***.reverse***

Η μέθοδος **reverse** αντιστρέφει τη σειρά των στοιχείων μιας λίστας.

Σύνταξη:

lista.reverse()

Παράδειγμα:

```
students=['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']
students.reverse()
print(students)
```

Output

['Ελένη', 'Πέτρος', 'Αγγελική', 'Γιώργος', 'Δημήτρης', 'Μανώλης']

- ***.clear***

Με την clear( ) μπορούμε να διαγράψουμε όλα τα στοιχεία μέσα σε μια λίστα.

Σύνταξη:

lista.clear()

Παράδειγμα:

```
students.clear()
print(students)
```

Output

[]

- ***del***

Η συνάρτηση **del** διαγράφει οποιοδήποτε στοιχείο είναι στο δείκτη που ορίζουμε.

Παράδειγμα:

```
students=['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']
del students[2]
print(students)
```

Output

['Μανώλης', 'Δημήτρης', 'Αγγελική', 'Πέτρος', 'Ελένη']

Προσοχή, η εντολή del students θα διαγράψει όλη τη λίστα!

- ***.join***

Μέσω της συνάρτησης **join** μπορούμε να μετατρέψουμε πολύ εύκολα μια λίστα σε συμβολοσειρά (string).

Παράδειγμα:

```
students=['Μανώλης', 'Δημήτρης', 'Γιώργος', 'Αγγελική', 'Πέτρος', 'Ελένη']
students_string = ",".join(students)
print(students_string)
```

Output

Μανώλης,Δημήτρης,Γιώργος,Αγγελική,Πέτρος,Ελένη

Όνομα συνάρτησης	Λειτουργία
len	Δέχεται ως όρισμα μία λίστα και επιστρέφει το πλήθος των στοιχείων της λίστας
min	Δέχεται ως όρισμα μία λίστα και επιστρέφει το μικρότερο των στοιχείων της λίστας
max	Δέχεται ως όρισμα μία λίστα και επιστρέφει το μεγαλύτερο των στοιχείων της λίστας
sum	Δέχεται ως όρισμα μία λίστα στην οποία τα στοιχεία της είναι αριθμοί και επιστρέφει το άθροισμα στοιχείων της λίστας

Παράδειγμα:

```
grades=[6, 7.5, 5, 8, 9.5, 6.5]
length=len(grades)
min_grade=min(grades)
max_grade=max(grades)
sum_of_grades=sum(grades)
print('Η λίστα έχει', length, 'στοιχεία')
print('Η μικρότερη βαθμολογία είναι', min_grade, 'και η μεγαλύτερη είναι', max_grade)
print('Το άθροισμα όλων των βαθμών είναι', sum_of_grades)
```

Output

```
Η λίστα έχει 6 στοιχεία
Η μικρότερη βαθμολογία είναι 5 και η μεγαλύτερη είναι 9.5
Το άθροισμα όλων των βαθμών είναι 42.5
```

- **Πρόσθεση λιστών:** Με την έννοια πρόσθεση λιστών εννοούμε τη **συνένωση** δύο η περισσοτέρων λιστών.

```
a=[1,2,3,4]
b=[5,6,7,8,9]
x=a+b
```

Η λίστα x θα έχει τη μορφή [1,2,3,4,5,6,7,8,9]

Εναλλακτικά, θα μπορούσαμε να γράψουμε x=a+[5,6,7,8,9]

- **Πολλαπλασιασμός λιστών:** Με την έννοια πολλαπλασιασμού μιας λίστας με έναν ακέραιο εννοούμε τη επανάληψη των στοιχείων όσες φορές προκύπτει από τον ακέραιο.

```
a=[1,2,3]
x=a*3
```

Η νέα λίστα x θα έχει τιμή [1,2,3,1,2,3,1,2,3]

- Μπορούμε επίσης να δημιουργήσουμε μια καινούργια λίστα μέσω της κατασκευής λίστας με περιγραφή (list comprehensions). Στο παρακάτω παράδειγμα βλέπουμε μια ήδη υπάρχουσα λίστα της οποίας τα στοιχεία διατρέχουμε ένα ένα, και κάθε ένα το υψώνουμε στο τετράγωνο. Το υψωμένο στο τετράγωνο στοιχείο πλέον ανήκει στην καινούργια λίστα.

1ος τρόπος

```
square_numbers = []
for x in range(1, 6):
   square_numbers.append(x**2)
print(square_numbers)
```

2ος τρόπος

```
square_numbers = [i**2 for i in range(1, 6)]
print(square_numbers)
```

μέσω της κατασκευής λίστας με περιγραφή χρειάστηκαν 2 γραμμές κώδικα για να δημιουργήσουμε τη λογική που θέλουμε να εκτελέσει το πρόγραμμα μας

Output

[1, 4, 9, 16, 25]

- Μια list comprehension δημιουργείται μέσω ενός ειδικού συντακτικού με αγκύλες. Πάντα ως αποτέλεσμα του είναι ένα καινούργιο αντικείμενο λίστας. Το γενικό μοντέλο είναι:

result = [transform iteration filter]

- Το κομμάτι του μετασχηματισμού (transform) γίνεται για κάθε φορά που προκύπτει από την διάτρεξη (iteration) και εφόσον ισχύει η συνθήκη φιλτραρίσματος (filter), αν αυτή υπάρχει. Η σειρά με την οποία γίνεται η διάτρεξη είναι συγκεκριμένη και καθορίζει την σειρά εμφάνισης των αποτελεσμάτων στη λίστα.

- Με βάση το γενικό ορισμό του list comprehension μπορούμε να προσθέσουμε και filters δηλαδή if-statements που φιλτράρουν τις τιμές πριν τις αναθέσουμε στο τελικό list. Ας δούμε πως υλοποιείται ένα τέτοιο απλό παράδειγμα.

```
square_numbers = [i**2 for i in range(1, 6) if i % 2==0]
print(square_numbers)
```

Output

[4, 16]

Για κάθε τιμή του i που ανήκει μέσα στο εύρος τιμών του range( ), κάνουμε πρώτα έναν έλεγχο αν είναι ζυγός αριθμός. Αν ναι, τότε δίνουμε την τιμή να εκτελεστεί από το transform σκέλος της εντολής που στο δικό μας παράδειγμα είναι η ύψωση στο τετράγωνο.

- Φυσικά μπορούμε να προσθέσουμε και το else μαζί με το if για να έχουμε έναν ολοκληρωμένο κώδικα συνθήκης. Όταν προσθέτουμε και το else, τότε μετακινούμε την συνθήκη πριν από το iteration.

```
square_numbers = [i**2 if i % 2==0 else i for i in range(1, 6)]
print(square_numbers)
```

Output

[1, 4, 3, 16, 5]

Παράδειγμα εξάσκησης: Ας δημιουργήσουμε ένα ακόμα παράδειγμα όπου έχουμε δύο λίστες που περιέχουν συμβολοσειρές. Θέλουμε να βρούμε ποιες συμβολοσειρές είναι κοινές και στις δύο λίστες.

```
Cities_a=['Athens','Paris','London','Tokyo']
Cities_b=['Los Angeles','Tokyo','New York','Dubai','Athens']
common_cities = [i for i in Cities_a if i in Cities_b]
print(common_cities)
```

Output

['Athens', 'Tokyo']

Παράδειγμα εξάσκησης: Θέλουμε να δημιουργήσουμε μια λίστα ώστε να αθροίσουμε όλους τους αριθμούς μέχρι το 100 που είναι πολλαπλάσια του 3 και του 7.

```
nums=[i for i in range(1,101) if i%3==0 or i%7==0]
print(sum(nums))
```

Output

2208

Παράδειγμα εξάσκησης: Να δημιουργήσετε μια νέα λίστα η οποία θα περιλαμβάνει τα στοιχεία της λίστας Cities_a με κεφαλαία γράμματα.

```
Cities_a=['Athens','Paris','London','Tokyo']
Cities_capital= [names.upper() for names in Cities_a]
print(Cities_capital)
```

Output

['ATHENS', 'PARIS', 'LONDON', 'TOKYO']

Παράδειγμα εξάσκησης: Να δημιουργήσετε μια λίστα η οποία να δέχεται τους βαθμούς ενός φοιτητή και να επιστρέφει το μήνυμα Pass ή Fail.

```
grades=[4,5,6,3,7,3,4,7,6]
text_grades=['Pass' if grade>=5 else 'Fail' for grade in grades]
print(text_grades)
```

Output

['Fail', 'Pass', 'Pass', 'Fail', 'Pass', 'Fail', 'Fail', 'Pass', 'Pass']

- Μπορείτε μέσω της κατασκευής λίστας με περιγραφή να εμφανίσετε τις θέσεις των μηνυμάτων Fail;

```
results=['Fail', 'Pass', 'Pass', 'Fail', 'Pass', 'Fail', 'Fail', 'Pass', 'Pass']
fail_results=[index for index, result in enumerate(results) if result=='Fail']
print(fail_results)
```

Output

[0, 3, 5, 6]

- Να δημιουργήσετε ένα πρόγραμμα που με την χρήση λίστας μέσα σε λίστα να διαβάζει τις βαθμολογίες τριών φοιτητών και να επιστρέφει το μέσο όρο.

```
grades=[['Giorgos',10,8,9],['Panos',9,5],['Stavros',10,6,8,6]]
sum=0
for student in grades:
    for grades in range(1,len(student)):
        sum+=student[grades]
    print(f"Student {student[0]} with average grade of {sum/(len(student)-1)}")
    sum=0
```

Output

```
Student Giorgos with average grade of 9.0
Student Panos with average grade of 7.0
Student Stavros with average grade of 7.5
```