ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ:

1. Python Advanced 6 - Λάμδα

ΠΕΡΙΕΧΟΜΕΝΑ:

1. Εισαγωγή με διατήρηση ταξινόμησης
2. Επιπλέον λειτουργικότητα

ΜΑΘΗΜΑ 2.5: Το module bisect — 1. Εισαγωγή με διατήρηση ταξινόμησης

Το module bisect περιέχει:

- Μία γρήγορη (χρησιμοποιεί διχοτόμηση (bisection)) εισαγωγή στοιχείου σε ταξινομημένο πίνακα, ώστε να διατηρείται η διάταξη.
- Υλοποίηση της δυαδικής αναζήτησης (βλ. Python - Μάθημα 12)

- Εισάγουμε ένα στοιχείο σε μία ήδη ταξινομημένη ακολουθία με τη συνάρτηση:

Συνάρτηση	Επεξήγηση
insort(element, list, low=0, high=len(list))	Εισάγει το element στη λίστα list διατηρώντας την ταξινόμηση (προαιρετικά ορίζουμε εύρος στον πίνακα στο οποίο θα γίνει η εισαγωγή [low, high)

  - Αν το στοιχείο υπάρχει ήδη στον πίνακα, τότε η εισαγωγή θα γίνει δεξιότερα από όλες τις υπόλοιπες εμφανίσεις του.
- Αντίστοιχα ορίζονται και οι συναρτήσεις:

Συνάρτηση	Επεξήγηση
insort_left(element, list, low=0, high=len(list))	Ομοίως αλλά εισάγει το στοιχείο αριστερότερα από άλλες εμφανίσεις του
insort_right(element, list, low=0, high=len(list))	Ίδια συμπεριφορά με την insort(..)

- Η πολυπλοκότητα είναι O(logn) για τον εντοπισμό της θέσης και O(n) για την εισαγωγή του στοιχείου)

Παράδειγμα 1: insert_keep_sorted

```
from random import randrange
from bisect import insort

numbers = [0]
for i in range(1, 30):
    numbers.append(numbers[i-1] + randrange(2))
print(numbers)

insort(numbers, 5)
print(numbers)
```

Προσοχή:

- Η insort δεν μπορεί να λειτουργήσει με πίνακα που έχει ταξινομηθεί με τη sort και έχει χρησιμοποιηθεί κάποιο κλειδί
- Θα δούμε έναν τρόπο για να ξεπεραστεί αυτό το πρόβλημα στην επόμενη διαφάνεια.

Παράδειγμα 2: key_sorted_problem

```
numbers = [randrange(100) for _ in range(20)]
print(numbers)

numbers.sort(key=lambda x: x//10+x%10)
print(numbers)

insort(numbers, 11)
print(numbers)
```

Συμπληρωματικά ορίζονται και οι συναρτήσεις:

Συνάρτηση	Επεξήγηση
bisect(element, list, low=0, high=len(list))	Επιστρέφει τη θέση εισαγωγής του element στη λίστα list ώστε να διατηρηθεί η ταξινόμηση (προαιρετικά ορίζουμε εύρος στον πίνακα στο οποίο θα γίνει η εισαγωγή [low, high), χωρίς να εισάγει το στοιχείο

- Αν το στοιχείο υπάρχει ήδη στον πίνακα, τότε επιστρέφει τη θέση, δεξιότερα από όλες τις υπόλοιπες εμφανίσεις του.
- Αντίστοιχα ορίζονται και οι συναρτήσεις:

Συνάρτηση	Επεξήγηση
bisect_left(element, list, low=0, high=len(list))	Ομοίως αλλά επιστρ. τη θέση αριστερότερα από άλλες εμφανίσεις του
bisect_right(element, list, low=0, high=len(list))	Ίδια συμπεριφορά με τη bisect(..)

Παράδειγμα 3: bisect_example.py

```
numbers = [0]
for i in range(1, 30):
    numbers.append(numbers[i-1] + randrange(2))
print(numbers)
print(bisect(numbers, 10))
print(bisect(numbers, 10, 5, 10))
```

Επίλυση του προβλήματος ταξινόμησης με βάση κλειδιά:

- Προτείνεται (official docs) να χρησιμοποιούμε tuples στον πίνακα με τα κλειδιά να έχουν υπολογιστεί εκ των προτέρων.

Παράδειγμα 4: key_sorted_solution.py

```
new_value = 25
key = new_value//10 + new_value%10
pos = bisect(keys, key)
insort(keys, key)
numbers.insert(pos, new_value)
print(numbers)
print(keys)
```

- Ενώ μέσω των bisect* μπορούμε να υλοποιήσουμε παραλλαγές της δυαδικής αναζήτησης

Παράδειγμα 4: typical_search_examples(docs).py

```
def find_lt(a, x):
    'Find rightmost value less than x'
    i = bisect_left(a, x)
    if i:
        return a[i-1]
    raise ValueError

print(index(numbers, 5))
```