ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ:

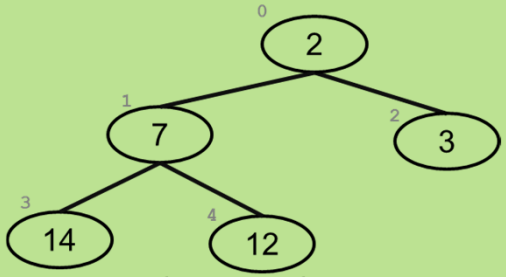
1. Python Advanced: Μάθημα 2 - Iterators
2. Python Advanced: Μάθημα 6 - Λάμδα

ΠΕΡΙΕΧΟΜΕΝΑ:

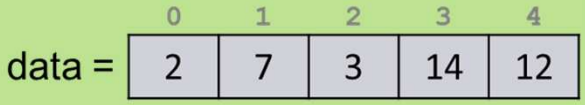
1. Βασικές Πράξεις
2. Δευτερεύουσες Πράξεις

Το module heapq:

- Μοντελοποιεί έναν σωρό ελαχίστων.
- Ο σωρός ελαχίστων είναι μία δομή δεδομένων που διατηρεί τα δεδομένα σε ένα δένδρο με την ιδιότητα ότι κάθε κόμβος είναι μικρότερος από τα παιδιά του και είναι σχεδόν πλήρες.



- Η απεικόνιση του δένδρου γίνεται σε μορφή πίνακα.



Ο σωρός ελαχίστων:

- Προσφέρει γρήγορη μετατροπή ενός πίνακα σε σωρό (**heapify**), σε χρόνο O(n)
- Εντοπίζουμε το **ελάχιστο στοιχείο** σε σταθερό χρόνο O(1)
- Γίνεται **προσθήκη ενός νέου στοιχείου** σε χρόνο O(logn)
- Γίνεται **απομάκρυνση του πρώτου στοιχείου** σε χρόνο O(logn)
- και γενικά είναι μία δομή που προτιμάται, όταν γνωρίζουμε ότι είτε δεν έχουμε πρόσβαση σε όλα τα δεδομένα, είτε θα έχουμε σταδιακά πρόσβαση και θέλουμε να χτίζουμε τη δομή που θα διατηρεί τα στοιχεία, ταξινομήμένα

Συναρτήσεις:

Συνάρτηση	Επεξήγηση
heapify(list)	Μετατρέπει τη λίστα σε σωρό ελαχίστων
heappush(heap, item)	Προσθέτει το στοιχείο item στο σωρό heap
heappop(heap)	Επιστρέφει το ελάχιστο στοιχείο του heap

Παράδειγμα 1: creation

```
from heapq import heapify, heappush, heappop
from random import randrange

array = [randrange(20) for i in range(10)]
print(array)
heapify(array)
print(array)

heappush(array, randrange(20))
heappush(array, randrange(20))
print("="*40)
while len(array)>0:
    item = heappop(array)
    print(f"{item}, {array}")
```

Προσφέρονται και οι εξής (Δευτερεύουσες) Συναρτήσεις

Συνάρτηση	Επεξήγηση
heappushpop (heap, item)	1) Προσθέτει το στοιχείο item στο σωρό heap 2) Αφαιρεί και επιστρέφει το ελάχιστο στοιχείο
heapreplace (heap, item)	1) Αφαιρεί το ελάχιστο στοιχείο του heap 2) Προσθέτει το στοιχείο item στο σωρό heap 3) Επιστρέφει το ελάχιστο στοιχείο
nsmallest (n, iterable, key=None)	Επιστρέφει τα n μικρότερα στοιχεία του iterable χρησιμοποιώντας εσωτερικά ένα heap. key: Συγκριτής που θα χρησιμοποιηθεί (key: item->value)

Ορίζεται και η nlargest() με ίδια ορίσματα και αντίστοιχη λειτουργία με τη nsmallest()

Παράδειγμα 2: smallest.py

```
array = [Person("name", randrange(20)) for i in range(10)]
print(*array)
heapify(array)
print(*array)

ml = nsmallest(4, array)
print(*ml)

ml = nsmallest(4, array, lambda person: 20-person.age)
print(*ml)
```

και ακόμη μία συνάρτηση χρήσιμη σε iterators

Συνάρτηση	Επεξήγηση
merge (*iterables, key=None, reverse=False)	Θεωρεί ότι τα iterables είναι ταξινομημένα. Τα συνενώνει σε έναν νέο iterator με τη σειρά που εμφανίζονται, ο οποίος και επιστρέφεται. key: Συγκριτής που θα χρησιμοποιηθεί (key: item->value) reverse: Αν τεθεί True, τότε ο iterator επιστρέφει τα στοιχεία με αντίστροφη σειρά.

Παράδειγμα 3: merging.py

```
data1 = [randrange(20) for _ in range(10)]
data2 = [randrange(20) for _ in range(10)]
data3 = [randrange(20) for _ in range(10)]

data1.sort(key=lambda x:x%3)
data2.sort(key=lambda x:x%3)
data3.sort(key=lambda x:x%3)

print(data1)
print(data2)
print(data3)
print(list(merge(data1, data2, data3, key=lambda x:x%3)))
```