ΠΕΡΙΕΧΟΜΕΝΑ:

1. Αρχεία
    1. Αρχεία Κειμένου
        1. modes
    2. Δυαδικά Αρχεία
    3. JSON αρχεία
2. Algorithm: Merge Sort (Ταξινόμηση με Συγχώνευση)
3. Game Project: Tic Tac Toe (με “AI”)
4. Data Project: CRUD – Αποθήκευση σε Αρχείο

Κωνσταντίνος Ζ.

Χρυσός Χορηγός Μαθήματος

Αγγελική Γ. - Σταυριανή Γ.

Ασημένιος Χορηγός Μαθήματος

Τα **αρχεία κειμένου (text files)** περιέχουν μόνο χαρακτήρες

- Η διαχείριση ενός αρχείου στην Python γίνεται ως εξής
  - Ανοίγουμε το αρχείο με την open:

```
f = open(filename, mode)
```

    - filename: το όνομα του αρχείου (string)
    - mode: τρόπος ανοίγματος (string):
      - r: reading(default) - διάβασμα
      - w: writing - γράψιμο
      - a: appending - προσάρτηση
      - r+: reading and writing – διάβασμα και γράψιμο
    - f: αντικείμενο που έχει μεθόδους για την επεξεργασία του αρχείου.
  - Κάνουμε την επεξεργασία του αρχείου (διάβασμα ή γράψιμο ανάλογα με τον τρόπο που ανοίξαμε το αρχείο)
    - π.χ. για γράψιμο χρησιμοποιούμε τη write:

```
f.write("String to be written")
```

  - Κλείσιμο αρχείου

```
f.close()
```

Παράδειγμα 1: file.open.close.py

```
f = open("test.txt", "w")
f.write("My text goes here")
f.close()
```

Διαχείριση του αρχείου με τη with:

- Γράφουμε το άνοιγμα του αρχείου με την εντολή:

```
with open(filename, mode) as f:
```

  - και οι εντολές διαχείρισης του αρχείου στοιχίζονται δεξιά κατά ένα επίπεδο.

- Με τη with δεν απαιτείται κλείσιμο του αρχείου (γίνεται αυτόματα).
- Είναι καλύτερος τρόπος για το άνοιγμα, γιατί αν το πρόγραμμα τερματίσει απροσδόκητα, περνάει ο έλεγχος στη Python για το κλείσιμο του αρχείου.

Παράδειγμα 2: with.py

```
with open("test2.txt", "w") as f:
    f.write("My text goes here\n")
    f.write("and another line")
```

Άσκηση 1:

Κατασκευάστε πρόγραμμα το οποίο:

- Ορίζει μία λίστα με 1500 τυχαίους ακέραιους στο εύρος [0,100]
- Αποθηκεύει τη λίστα σε ένα αρχείο κειμένου με όνομα numbers.txt, έτσι ώστε κάθε αριθμός να είναι σε διαφορετική γραμμή του αρχείου.

Όταν κάνουμε **άνοιγμα για διάβασμα (r)** μπορούμε:

- Να διαβάσουμε όλο το αρχείο με τη μέθοδο **read**:

```
# read.py
with open("test2.txt", "r") as f:
    contents = f.read()
    print(contents)
```

  - (Δέχεται προαιρετικό όρισμα το πλήθος των bytes που θα διαβάσει)
- Να διαβάσουμε κατά γραμμές το αρχείο με ένα **for** στο αντικ/νο:

```
# for.readlines.py
with open("test2.txt", "r") as f:
    for line in f:
        print(line)
```

  - (Προσοχή ότι διαβάζει και το χαρακτήρα αλλαγής γραμμής)
- Να κατασκευάσουμε μία λίστα με τις γραμμές με τη **readlines**:

```
# readlines.py
with open("test2.txt", "r") as f:
    lines = f.readlines()
print(lines)
```

  - Υπάρχει και η readline() που διαβάζει μία γραμμή

Άσκηση 2: Τροποποιήστε το πρόγραμμα readlines ώστε να μην περιλαμβάνει στα μέλη της χαρακτήρες αλλαγής γραμμής

Όταν κάνουμε **άνοιγμα για γράψιμο (w)** μπορούμε:

- Να γράψουμε μία συμβολοσειρά με τη **write**
  - (Προσοχή ότι δεν προσθέτει χαρακτήρες αλλαγής γραμμής)

Άσκηση 3: Γράψτε μία συνάρτηση copy_file(filename1, filename2) που αντιγράφει τα περιεχόμενα του filename2 στο filename1

Άσκηση 4: Κατασκευάστε πρόγραμμα το οποίο:

- Ανοίγει για γράψιμο το αρχείο overwrite.txt και γράφει μία γραμμή κειμένου σε αυτού.

Τρέξτε το πρόγραμμα δύο φορές. Τι παρατηρείτε;

Όταν κάνουμε **άνοιγμα για προσάρτηση (a)** μπορούμε:

- Να γράψουμε στο τέλος του αρχείου
- (χωρίς να διαγράψουμε τα περιεχόμενά του)

Άσκηση 5: Κατασκευάστε πρόγραμμα το οποίο ανοίγει το αρχείο overwrite.txt για προσάρτηση, γράφει μία γραμμή και τερματίζει

Παρατήρηση:

- Το άνοιγμα για διάβασμα είναι default, συνεπώς αν ανοίξουμε το αρχείο ως:

```
f = open(filename)
```

  - Το αρχείο θα ανοίξει για διάβασμα

Άσκηση 6: Ο Βασιλιάς Ληρ

Κατεβάστε το έργο "King Lear" από το shakespeare.mit.edu και σώστε το σε ένα αρχείο.

- Διαβάστε κατά γραμμές τα περιεχόμενα, αποθηκεύοντας τα σε μία λίστα γραμμών.
- Τυπώστε όλο το έργο έτσι ώστε:
  - Να εμφανίζεται μία κενή γραμμή πριν και μετά από κάθε όνομα (τα ονόματα είναι με κεφαλαία γράμματα)
  - Να εμφανίζονται οι υπόλοιπες γραμμές ένα tab δεξιά.

Άσκηση 7: Συγχώνευση αρχείων

Κατασκευάστε μία συνάρτηση με όνομα merge η οποία:

- Δέχεται ως παραμέτρους τρία ονόματα αρχείων.
- Συγχωνεύει τα περιεχόμενα των δύο πρώτων αρχείων στο τρίτο (πρώτα τα περιεχόμενα του πρώτου αρχείου και μετα τα περιεχόμενα του δεύτερου αρχείου)

(Προχωρημένο θέμα που μπορεί να παραληφθεί. βλ. βίντεο)

Ο τύπος δεδομενων byte:

- Αποθηκεύει bytes
- Μία τιμή, κωδικοποιείται με b ακολουθούμενο από ένα string από bytes (τα οποία κωδικοποιούνται με ένα \x ακολουθούμενο από δύο δεκαεξαδικά ψηφία), π.χ.:

```
b'\x00\x00\x00\x0f'
```

- Μετατρέπουμε έναν ακέραιο x σε bytes με τη μέθοδο to_bytes:

```
x.to_bytes(4, byteorder='big')
```

- Και μετατρέπουμε bytes σε ακέραιο ως εξής:

```
x = int.from_bytes(b, byteorder='big')
```

Παράδειγμα 3: bytes.py

```
x = 15
b = x.to_bytes(4, byteorder='big')
print(b)
n = int.from_bytes(b, byteorder='big')
print(n)
```

Παρατήρηση:

- Γενικά οι τύποι δεδομένων της Python δεν έχουν σταθερό μήκος, οπότε απαιτείται προσεκτική μετατροπή σε bytes.

Για να ανοίξουμε ένα δυαδικό αρχείο:

- Προσθέτουμε το **b στο mode**.

Παράδειγμα 4: binary.file.py

```
numbers = [1,2,3,4]
with open("binary.dat", "wb") as f:
    for number in numbers:
        f.write(number.to_bytes(4, byteorder='big'))

with open("binary.dat", "rb") as f:
    for i in range(len(numbers)):
        b = f.read(4)
        print(int.from_bytes(b, byteorder='big'))
```

Για να κάνουμε τυχαία προσπέλαση σε ένα δυαδικό αρχείο:

- **tell():** Δίνει τη θέση γραψίματος/εγγραφής (cursor)
- **seek(offset, from):** offset=bytes από το from (0=αρχή, 1=τρέχουσα θέση, 2=τέλος)

Ενώ το άνοιγμα του αρχείου γίνεται σε **mode rb+,** δηλαδή:

- Άνοιγμα σε δυαδική μορφή με δικαίωμα στο οποίο επιτρέπεται γράψιμο και έγγραφή.

Παράδειγμα 5: random.access.py

(Στο παράδειγμα γίνεται τυχαία προσπέλαση σε ένα δυαδικό αρχείο).

- Το **JSON (JavaScript Object Notation):**
  - **Είναι ένας τρόπος σύνταξης δεδομένων ώστε να αποθηκεύονται και να ανταλλάσσονται με εύκολο τρόπο.**
  - Το module json μας προσφέρει τρόπους για να μετατρέπουμε:
    - Δεδομένα που είναι τύπων dictionary, list, tuple, string, int, float, Boolean στο πρότυπο JSON
    - Δεδομένα JSON στις αντίστοιχα Python αντικείμενα.

Συνεπώς:

- Χειριζόμαστε τα δεδομένα μας, τα μετατρέπουμε σε JSON και τα αποθηκεύουμε σε αρχείο κειμένου.
- Ανοίγουμε το αρχείο κειμένου, διαβάζουμε τα JSON αντικείμενα και τα μετατρέπουμε σε αντικείμενα της Python.

Χρήση

- Κάνουμε import το module json
- Γράφουμε στο αρχείο με τη **μέθοδο dump():**

```
json.dump(obj, f)
```

  - obj: είναι το αντικείμενο που θέλουμε να γράψουμε
  - f: το αντικείμενο διαχείρισης του αρχείου
- Διαβάζουμε από το αρχείο με τη **μέθοδο load()**

```
obj = json.load(f)
```

  - Παίρνει όρισμα το αρχείο και επιστρέφει το αντικείμενο που διάβασε

Παράδειγμα 6: json.dump

```
import json
numbers = [1,2,3,4,5,6,7]
with open("numbers.json", "w") as f:
   json.dump(numbers, f)
```

Παράδειγμα 7: json.load

```
import json

with open("numbers.json", "r") as f:
   numbers = json.load(f)

print(numbers)
```

Αποθήκευση σε **κοινό JSON αρχείο πολλών αντικειμένων**

- Αποθηκεύουμε κάθε αντικείμενο σε μία ξεχωριστή γραμμή.

Διάβασμα από κοινό JSON αρχείο πολλών αντικειμένων:

- Διαβάζουμε μία μία τις γραμμές ως συμβολοσειρές
- Κάθε συμβολοσειρά τη μετατρέπουμε σε αντικείμενο με την:

```
obj = json.loads(string)
```

Παράδειγμα 8: json.multiple.data

Στο παράδειγμα αυτό, αποθηκεύουμε 4 αντικείμενα διαφορετικών τύπων δεδομένων και έπειτα τα διαβάζουμε από το αρχείο.

Η ενέργεια:

- "Άνοιξε το αρχείο και διάβασε το και αν δεν υπάρχει, αρχικοποίησε απλά τα δεδομένα" απαιτεί **χειρισμό της εξαίρεσης** FileNotFoundError και γίνεται με τον εξής κώδικα:

```
try:
    with open("reminders.txt") as f:
        reminders = json.load(f)
except FileNotFoundError:
    reminders = []
```

Θα μάθουμε περισσότερα για τις εξαιρέσεις στο μάθημα 19.

Άσκηση 8: Reminders

Κατασκευάστε ένα απλό πρόγραμμα υπενθυμίσεων:

- Μία υπενθύμιση είναι μία συμβολοσειρά
- Οι υπενθυμίσεις θα αποθηκεύονται σε μία λίστα από συμβολοσειρές.
- Το πρόγραμμα θα εμφανίζει ένα μενού με τέσσερις επιλογές: Προσθήκη υπενθύμισης, διαγραφή υπενθύμισης, εκτύπωση υπενθυμίσεων και έξοδος.

Οι υπενθυμίσεις να φορτώνονται στην αρχή του προγράμματος από ένα JSON αρχείο και οποτεδήποτε γίνονται αλλαγές στη λίστα υπενθυμίσεων να γίνεται και η αποθήκευση στο αρχείο.

Άσκηση 9: Administration

Κατασκευάζουμε τον έλεγχο της εισόδου με username και password. Συγκεκριμένα:

- Κάθε χρήστης θα είναι ένα λεξικό με στοιχεία full_name, username, password, role (admin ή user)
- Όλοι ο χρήστες θα αποθηκεύονται σε μία λίστα.
- Αποθηκεύουμε τη λίστα σε ένα αρχείο JSON (users.json)

Κατασκευάστε ένα πρόγραμμα που να δημιουργεί και να προσθέτει χρήστες στο αρχείο (admin.py)

Κατασκευάστε ένα δεύτερο πρόγραμμα (user.py):

- Θα διαβάζει τη λίστα χρηστών από το αρχείο
- Θα προτρέπει το χρήστη να εισάγει το username και το password
- Θα βγάζει μορφοποιημένο μήνυμα:
  - Μήνυμα λάθους, αν το username ή το password είναι λάθος
  - "Welcome Admin" στο διαχειριστή
  - "Welcome" ακολουθούμενο από το πλήρες όνομα στο χρήστη.

Επεκτάσεις - Παρατηρήσεις

- Τα αρχεία έχουν αρκετές επεκτάσεις, όπως διαχείριση αρχείων σε φάκελους, κωδικοποιήσεις χαρακτήρων, ειδικές συναρτήσεις κ.α. που θα δούμε σε προχωρημένα μαθήματα

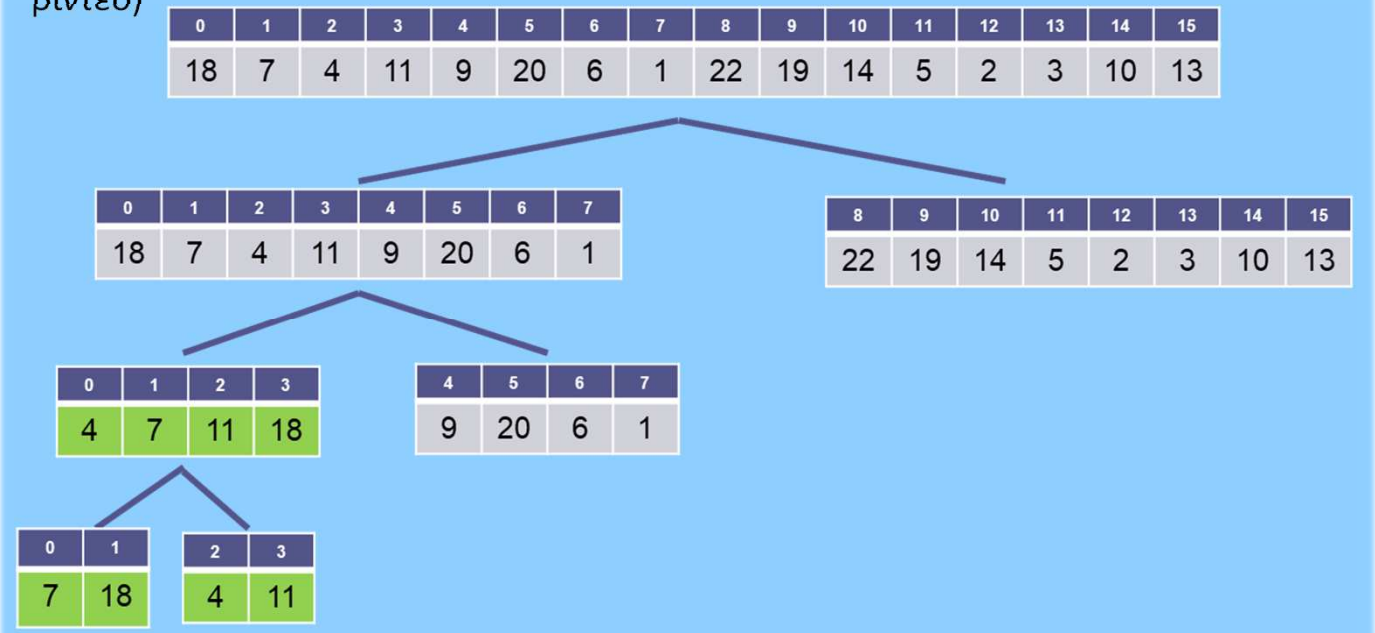
Συνεχίζουμε με το πρόβλημα της ταξινόμησης, με έναν ακόμη αλγόριθμο που το επιλύει.

Το **σκεπτικό της ταξινόμησης με συγχώνευση (merge sort)** είναι:

- (Αναδρομικά) Ταξινόμησε το αριστερό μισό του πίνακα
- (Αναδρομικά) Ταξινόμησε το δεξί μισό του πίνακα
- Συγχώνευσε τους δύο υποπίνακες, σε έναν ταξινομημένο πίνακα.

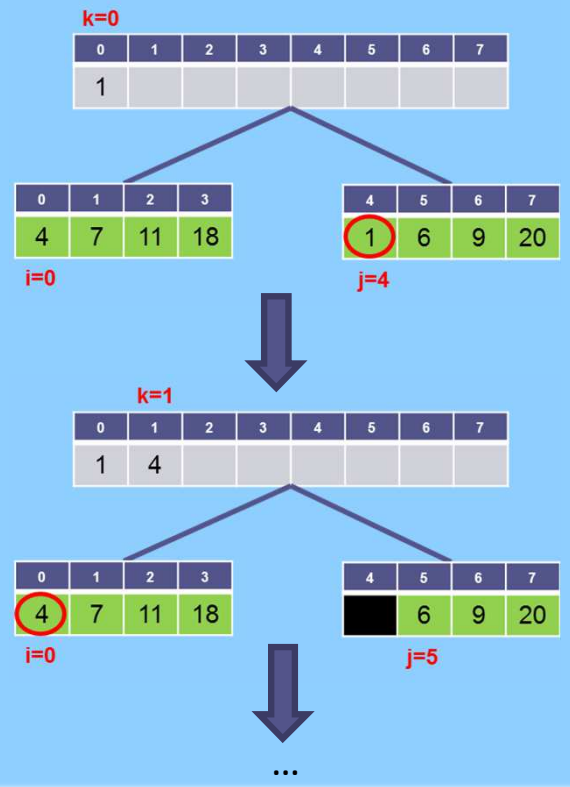
Παράδειγμα Εκτέλεσης:

Βλέπουμε ένα ενδιάμεσο βήμα της αναδρομικής διαδικασίας. Η αναδρομική κλήση για τις θέσεις 0-3 έχει ολοκληρωθεί και ο υποπίνακας είναι ταξινομημένος (βλ. βίντεο)



Η **διαδικασία της συγχώνευσης(merge)**:

- Σαρώνοντας από αριστερά προς τα δεξιά τους δύο υποπίνακες,
- Επιλέγει το μικρότερο στοιχείο και το θέτει στον ταξινομημένο πίνακα:



Άσκηση 10:

Υλοποιήστε τον αλγόριθμο merge sort

Άσκηση 11.1: Παίκτης εναντίον Υπολογιστή

Θα τροποποιήσουμε το πρόγραμμα ώστε να μπορούμε να παίξουμε εναντίον του υπολογιστή :

- Το loop της επιλογής τετραγώνου θα αφορά μόνο τον παίκτη X
- Κατασκευάστε συνάρτηση με όνομα computer_moves η οποία ο παίκτης O (υπολογιστής) θα επιλέγει στην τύχη ένα από τα άδεια τετράγωνα.

Κάντε και όποιες άλλες τροποποιήσεις κρίνετε απαραίτητες, στην τελευταία έκδοση του πργράμματος της Τρίλιζας (Μάθημα 11, άσκηση 9), έχοντας τα παραπάνω ως κατευθυντήριες γραμμές.

Παρατήρηση για τη 2η άσκηση:

- Ο αλγόριθμος αυτός προσομοιώνει τη δική μας σκέψη
- Υπάρχουν και αλγόριθμοι πιο συστηματικής εφαρμογής της τεχνητής νοημοσύνης (π.χ. αλγόριθμος minimax)
- ή ακόμη και εφαρμογές μηχανικής μάθησης!

Άσκηση 11.2: Επίπεδα δυσκολίας παιχνιδιού

Η προηγούμενη έκδοση είναι κάπως άστοχη μιας και ο υπολογιστής ουσιαστικά δεν σκέφτεται την κίνησή του. Θα κάνουμε τροποποιήσεις ώστε να είναι πιο "έξυπνος".

- Ορίστε μια μεταβλητή με όνομα level στη main().
- Αρχικά ο χρήστης να επιλέγει το επίπεδο δυσκολίας (1-εύκολο, 2-δύσκολο)
- Η συνάρτηση computer_moves να παίρνει όρισμα το level. Αν το level = 1, τότε ο υπολογιστής να παίζει με τυχαίο τρόπο (όπως πριν). Αν το level = 2, τότε ο υπολογιστής επιλέγει το τετράγωνό του με τον εξής αλγόριθμο:
  - Να γίνει έλεγχος σε κάθε γραμμή, στήλη ή διαγώνιο για το αν είναι σε κίνδυνο να νικήσει ο παίκτης (έχει συμπληρώσει 2 από τα 3 τετράγωνα). Αν ναι, να αποτρέπεται αμέσως με την επιλογή του κατάλληλου τετραγώνου.
  - Αν το κεντρικό τετράγωνο είναι άδειο, τότε να επιλεχθεί αυτό.
  - Αλλιώς να επιλέγεται τυχαία γωνιακό τετράγωνο.
  - Αλλιώς να επιλέγεται τυχαία τετράγωνο που είναι μεσαίο σε γραμμή ή στήλη.

Η παραπάνω εκδοχή είναι "αμυντική". Ο υπολογιστής παίζει για να μην χάσει! Σκεφθείτε και επιθετικές εκδοχές παιχνιδιού.

Άσκηση 12: Αποθήκευση σε Αρχεία

Τροποποιήστε την τελευταία έκδοση του project (Μάθημα 14, άσκηση 5) ώστε να αποθηκεύει σε μορφή JSON τις δομές των μαθητών και των καθηγητών. Συγκεκριμένα σε κάθε module:

- Κατασκευάστε συναρτήσεις με όνομα init_pupils_data και init_teachers_data αντίστοιχα οι οποίες φορτώνουν από το σκληρό δίσκο τα αρχεία και θα αρχικοποιεί τις δύο δομές.
- Κατασκευάστε συναρτήσεις με όνομα save_pupils_data και save_teachers_data αντίστοιχα οι οποίες αποθηκεύουν στο σκληρό δίσκο τις δύο δομές.
- Καλέστε σε κατάλληλα σημεία του προγράμματος τις δύο συναρτήσεις.