ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ Η/Υ
σε Python

Διάλεξη 9: Διαχείριση αρχείων

Μάρκος Ζάμπογλου, mzampoglou@aegean.gr

Αρχεία δεδομένων

- Ο κώδικας που είδαμε ως τώρα, λειτουργούσε με δεδομένα από το χρήστη ή περασμένα απευθείας μέσα στον κώδικα
- Προφανώς αυτός δεν είναι ένας πρακτικός τρόπος διακίνησης πληροφορίας
- Σε πρακτικές εφαρμογές η πληροφορία συνήθως αντλείται είτε από αρχεία, είτε από βάσεις δεδομένων, είτε από το δίκτυο

Άνοιγμα αρχείου

- Για οποιαδήποτε αλληλεπίδραση με αρχείο στη βασική Python (δηλαδή χωρίς επιπλέον βιβλιοθήκες), χρησιμοποιούμε τη συνάρτηση **`open()`**

```
open(file, mode='r', buffering=-1, encoding=None,
errors=None, newline=None, closefd=True, opener=None)
```

  - Από όλες τις παραμέτρους που μπορεί να δεχτεί, συνήθως μας ενδιαφέρουν οι **`file`** και **`mode`**
  - Επιστρέφει ένα αντικείμενο που περιέχει το αρχείο

Modes της open()

```
f = open('data.txt') #default mode='r'
```

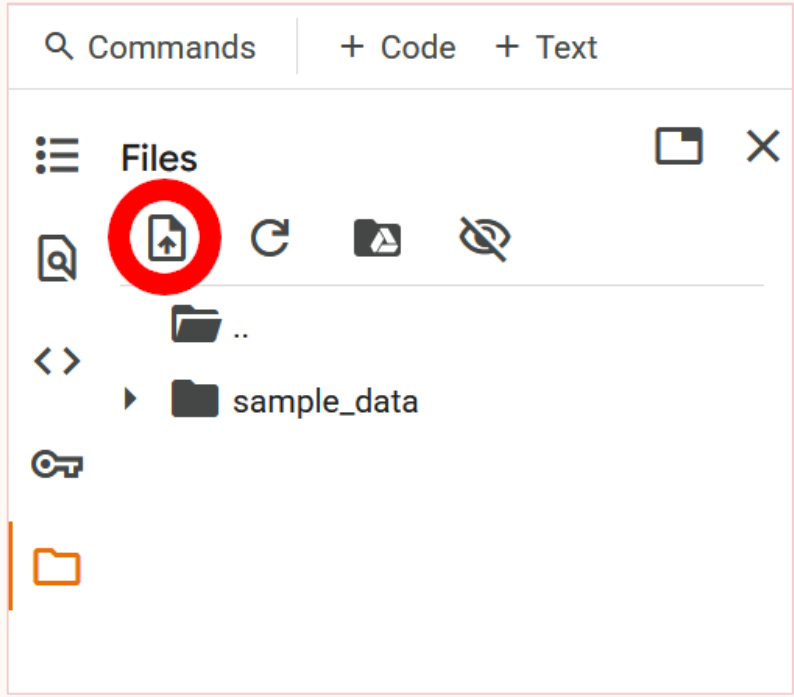
Άνοιξε το data.txt για ανάγνωση σε binary mode

- Πιθανές τιμές της παραμέτρου ***mode***:

`'r'`	Άνοιγμα για **ανάγνωση** (*default*, συνώνυμο της `rt`)
`'w'`	Άνοιγμα για **γράψιμο** (αντικαθιστά υπάρχον αρχείο)
`'x'`	Άνοιγμα για **γράψιμο** (αποτυγχάνει αν υπάρχει ήδη το αρχείο)
`'a'`	Άνοιγμα για **προσθήκη στο τέλος** υπάρχοντος αρχείου
`'+'`	Άνοιγμα για **updating** (γράψιμο και ανάγνωση)
`'b'`	Άνοιγμα σε ***binary* mode**
`'t'`	Άνοιγμα σε ***text* mode** (*default* αν δεν θέσουμε `'b'`)

Τοποθεσία αρχείου

- Κάθε αρχείο βρίσκεται εξορισμού σε έναν φάκελο που χαρακτηρίζεται από μια διαδρομή (path)
- Στο Jupyter Notebook και στα αρχεία script, όταν γράφω ένα όνομα αρχείου χωρίς path, ο κώδικας το αναζητά δίπλα στο αρχείο του κώδικα (.ipynb ή .py)
- Στο Google Colab, πρέπει να το ανεβάσουμε χρησιμοποιώντας το File tab στα αριστερά.



Ανάγνωση περιεχομένων αρχείου με τη `.read()`

```
f = open('Διάλεξη8_data.txt')
full_text = f.read()
print(type(f))
f.close()
print('---------')
print(full_text)
print('---------')
print(type(full_text))
<class '_io.TextIOWrapper'>
---------
Αρνάκι άσπρο και παχύ
[…]
---------
<class 'str'>
```

Η μέθοδος **`.read()`** σε ένα αρχείο τύπου text, αν δεν της δώσουμε καμία παράμετρο, επιστρέφει όλα τα περιεχόμενά του αρχείου σε ένα string

Κλείσιμο αρχείου

- Είναι απαραίτητο να κλείνουμε το αντικείμενο **`file`** όταν τελειώσουμε.
- Αλλιώς δημιουργούνται ενδεχόμενα προβλήματα στο αρχείο ή και στη μνήμη.

```
print(f.closed)
f.close()
print(f.closed)
```

- Εναλλακτικά, μπορούμε να το ανοίγουμε με την **`with`**, που το κλείνει αυτόματα όταν ολοκληρωθεί ("***context management***"):

```
with open('Διάλεξη8_data.txt ') as f:
        print("Πρώτη γραμμή:")
        full_text = f.read()
print(full_text)
```

Με το τέλος του μπλοκ, το αρχείο που δείχνει η **`f`** κλείνει

Η **`f`** και οι άλλες μεταβλητές που δημιουργήσαμε όμως παραμένουν

Διαχείριση σφαλμάτων

- Αν το όνομα που δώσω δεν αντιστοιχεί σε αρχείο, προκαλείται σφάλμα
- Λύση: δομή **try...except**

```
try:
  with open('Διάλεξη8_data.txt') as f:
    full_text = f.read()
except FileNotFoundError:
  print("Υπάρχει πρόβλημα με το αρχείο")
else:
  print(full_text)
```

Η έννοια του δείκτη (pointer)

- Κατά την ανάγνωση και την εγγραφή ενός αρχείου υπάρχει ένας **δείκτης** που κρατάει τον τελευταίο χαρακτήρα στον οποίο σταμάτησε η διαδικασία
- Π.χ. η `.read(5)` διαβάζει πέντε χαρακτήρες και μετακινεί το δείκτη μετά τον πέμπτο από αυτούς
  - Η επόμενη `.read()` θα ξεκινήσει από αυτό το σημείο
- Αντίστοιχα ο δείκτης κατά το γράψιμο, μένει στη θέση της τελευταίας εγγραφής
- Έχουμε επίσης στη διάθεσή μας τη μέθοδο **`seek()`**

  `file.seek(0)`

  Στέλνει τον pointer στην αρχή του αρχείου

  Επιστρέφει: **`None`**

🤓 Στην πραγματικότητα η **`seek`** έχει πολύ περισσότερες δυνατότητες, αλλά δε θα μπούμε σε αυτές

file.read() και file.seek(0)

```
with open('Διάλεξη8_data.txt') as f:
  partial_text = f.read(5)
  print(partial_text)
```
Αρνάκ
```
  f.read(4)
  partial_text2 = f.read(5)
  print(partial_text2)
```
προ κ
```
  f.seek(0)
  print(f.read(5))
```
Αρνάκ

`readline()` και `readlines()`

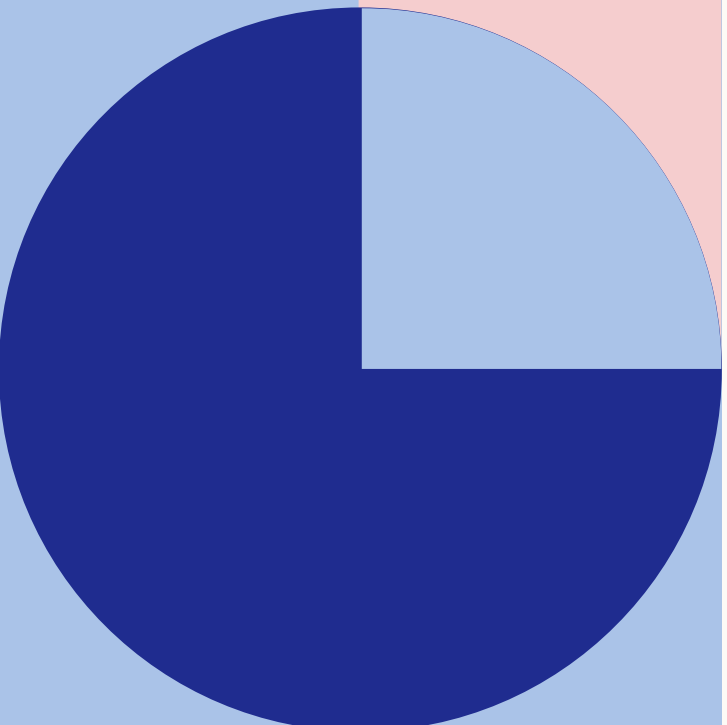
- Η **`readline()`** επιστρέφει μια ολόκληρη γραμμή (δηλαδή το string μέχρι και τον επόμενο χαρακτήρα αλλαγής γραμμής)
  - Στην επαναληπτική σάρωση γραμμών, είναι χρήσιμο ότι για την Python το κενό string θεωρείται False

```
line = f.readline()
print(line)
while line: #μέχρι να βρούμε κενή γραμμή
    line = f.readline()
    print(line)
```

- Η **`readlines()`** επιστρέφει όλες τις γραμμές σε μια λίστα από string

Παράδειγμα 9.1

Γιατί βγαίνουν αραιά οι γραμμές; Να τροποποιήσετε τον παραπάνω κώδικα ώστε να εμφανίζονται χωρίς επιπλέον κενά.



Παράδειγμα 9.1

Γιατί βγαίνουν αραιά οι γραμμές; Να τροποποιήσετε τον παραπάνω κώδικα ώστε να εμφανίζονται χωρίς επιπλέον κενά.

```
with open('Διάλεξη8_data.txt') as f:
  line = f.readline()
  while line:
    print(line.replace("\n",""))
    line = f.readline()
```

`write()` και `writelines()`

- Για να γράψουμε σε ένα αρχείο, χρησιμοποιούμε τη **`write()`** και τη **`writelines()`**
- Για να μπορούμε να γράψουμε σε ένα αρχείο, πρέπει να το ανοίξουμε στο κατάλληλο mode (**`w`**, **`a`**, ή **`x`**)
- Η **`write()`** ξεκινάει να γράφει από το σημείο που βρίσκεται ο pointer

Γράφοντας με τη `write()`

```
with open('data_written.txt','w') as f:
    f.write('Γράφω χαρακτήρες')
    f.write(' και αλλάζω γραμμή.\n')
    f.write('Και ξαναγράφω')
```

Ας δούμε τι γράψαμε:

```
with open('data_written.txt') as f:
    print(f.read())
```

```
Γράφω χαρακτήρες και αλλάζω γραμμή.
Και ξαναγράφω
```

Γράφοντας με τη `writelines()`

- Η μέθοδος **`writelines()`** ενός αντικειμένου αρχείου δέχεται ως παράμετρο μια λίστα από strings και την γράφει στο αρχείο
  - Ξεκινάει και αυτή από το σημείο που βρίσκεται ο pointer
- Παρά με τη συνωνυμία με τη **`readlines()`**, δεν προσθέτει αλλαγή γραμμής στο τέλος του κάθε string

```
with open('data_written.txt','a') as f:
  f.writelines(['Μια λίστα ', ' από στοιχεία\n', '
                                που γράφεται ολόκληρη'])
```

Παράδειγμα 9.2

Δίνεται ένα αρχείο με όνομα `server.log` που περιέχει συμβάντα συστήματος, ένα ανά γραμμή με τη μορφή **[TIMESTAMP] [STATUS] [MESSAGE]**.

Π.χ. `2024-05-20 10:31:45 ERROR Timeout`

Να γραφεί πρόγραμμα που να το διαβάζει και να δημιουργεί δύο αρχεία: ένα `errors.log` με όλες τις γραμμές που έχουν status ERROR, και ένα `summary.txt` που θα περιέχει το πλήθος των γραμμών του `server.log`, το πλήθος των γραμμών ERROR και τον πιο συνηθισμένο τύπο ERROR (με βάση το MESSAGE).

Παράδειγμα 9.2

Δίνεται ένα αρχείο με όνομα `server.log` που περιέχει συμβάντα συστήματος, ένα ανά γραμμή με τη μορφή **[TIMESTAMP] [STATUS] [MESSAGE]**.

Π.χ. `2024-05-20 10:31:45 ERROR Timeout`

Να γραφεί πρόγραμμα που να το διαβάζει και να δημιουργεί δύο αρχεία: ένα `errors.log` με όλες τις γραμμές που έχουν status ERROR, και ένα `summary.txt` που θα περιέχει το πλήθος των γραμμών του `server.log`, το πλήθος των γραμμών ERROR και τον πιο συνηθισμένο τύπο ERROR (με βάση το MESSAGE).

```
lines=0
errors=0
with open('server.log','r') as s:
  with open('errors.log','w') as e:
    l = s.readlines()
    for line in l:
      lines+=1
      if line.split(' ')[2]=='ERROR':
        e.write(line)
        errors+=1
with open('summary.txt','w') as t:
  t.write(f"Total lines: {lines}.
Total errors: {errors}.")
```

Συντήρηση – Pickling (1/2)

- Για να αποθηκεύσω δεδομένα στην πηγαία τους μορφή (όχι ως strings δηλαδή), η Python παρέχει τη βιβλιοθήκη **`pickle`**
  - Και τις μεθόδους **`picle.dump(object,file)`** που γράφει το αντικείμενο **`object`** στο αρχείο **`file`** και επιστρέφει **`None`**
  - και **`picle.load(file)`** που διαβάζει το αντικείμενο που περιέχεται στο αρχείο **`file`** και το επιστρέφει

```
import pickle
with open('pickle.bin','wb') as f:
    pickle.dump([1,2,3],f)
    pickle.dump([4,5,6],f)
with open('pickle.bin','rb') as f:
    a=pickle.load(f) #a=[1,2,3]
    b=pickle.load(f) #b=[4,5,6]
```

Συντήρηση – Pickling (2/2)

- Η **`pickle`** συνδυάζεται με δυαδικό (*binary*) mode ανάγνωσης και εγγραφής (**`'rb'`** και **`'wb'`**)
- Όταν αποθηκεύω πολλά αντικείμενα σε ένα αρχείο, ο μόνος τρόπος να τα ξεχωρίσω είναι η σειρά με την οποία γράφτηκαν
- Χάνουν τα ονόματά τους
- Μια εύκολη λύση είναι να βάλω όλα μαζί σε ένα λεξικό και να τα αποθηκεύσω ως ένα αντικείμενο