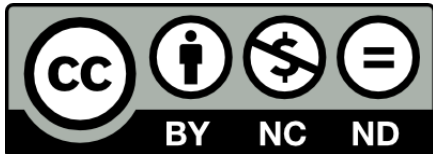


Εισαγωγή στην Python

9





Copyright

Το παρόν εκπαιδευτικό υλικό προσφέρεται ελεύθερα υπό τους όρους της άδειας Creative Commons:

- Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Όχι Παράγωγα Έργα 3.0.

Για να δείτε ένα αντίγραφο της άδειας αυτής επισκεφτείτε τον ιστότοπο

<https://creativecommons.org/licenses/by-nc-nd/3.0/gr/>

Στ. Δημητριάδης, 2015

Περιεχόμενα

- Open: Αρχεία
- Αρχεία κειμένου (Text files)
- Δυαδικά αρχεία (Binary files)
- Διατήρηση (pickling)
- Διαχείριση εξαιρέσεων

open

Αρχεία



Αρχεία: μικρή εισαγωγή

--1

- **Αρχείο:** ένα ονοματισμένος αποθηκευτικός χώρος στον υπολογιστή που τον διαχειρίζεται το λειτουργικό σύστημα
- **Open:** στην Python η συνάρτηση open δημιουργεί ένα αντικείμενο αρχείου (file object) το οποίο λειτουργεί ως σύνδεσμος προς το εξωτερικό φυσικό αρχείο που βρίσκεται κάπου στον υπολογιστή

```
Python 3.4.0 Pygame2.1.0-Dev2 - C:\Python34\Python.exe [1]
# (3-1) ΒΡΟΧΟΣ ΠΑΙΧΝΙΔΙΟΥ
while True:
    # (3-1) ΟΡΑ ΤΕΤΟΝΟΤΗΤΕΣ: ΕΛΕΓΧΟΣ
    for ev in pygame.event.get():
        if ev.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    # (3-2) ΚΑΘΟΡΙΣΜΟΣ ΚΑΤΑΣΤΑΣΗΣ
    # ΕΛΕΓΧΟΣ ΚΕΙΛΟΝ
    if ev.type == pygame.KEYDOWN:
        print("Τύπος = ", ev.type, "Κωδικός")
        if ev.key == K_LEFT:
            color = BLACK
        elif ev.key == K_RIGHT:
            color = WHITE
        elif ev.key == K_UP:
            color = RED
        elif ev.key == K_DOWN:
            color = GREEN
        elif ev.key == K_ESCAPE:
            pygame.quit()
            sys.exit()

    # (3-3) ΛΟΓΙΣΤΙΚΗ ΠΑΙΧΝΙΔΙΟΥ
    #
    my_screen.fill(color)
    pygame.display.update() # (3-4) Ενημέρωση V1
    [x] Ctrl C
```



- Αφού καλέσετε την **‘open’** μπορείτε να μεταφέρετε αλφαριθμητικά (‘γράμμες’) δεδομένων από και προς το εξωτερικό αρχείο καλώντας τις μεθόδους του αντικειμένου

Αρχεία: μικρή εισαγωγή

--2

- **core type:** τα αρχεία θεωρούνται αντικείμενα βασικού τύπου γιατί δημιουργούνται από μια ενσωματωμένη συνάρτηση (built-in function)
- **Όμως:** δεν είναι ούτε αριθμοί, ούτε ακολουθίες (πχ. λίστες), ούτε αντιστοιχίσεις (πχ. λεξικά) – απλά και μόνον προσφέρουν **μεθόδους** για όλες τις συνηθισμένες εργασίες επεξεργασίας αρχείου.
- **Μέθοδοι** αρχείου (file methods): οι περισσότερες αφορούν **διεργασίες εισόδου/εξόδου δεδομένων** από και προς το εξωτερικό αρχείο



Open: Άνοιγμα αρχείου

--1

- Για το άνοιγμα αρχείου καλούμε τη συνάρτηση '**open**' με ορίσματα:
 - **Όνομα αρχείου** (external filename)
 - **Τύπος επεξεργασίας** (processing mode)
- Η κλήση επιστρέφει ένα αντικείμενο τύπου αρχείου εφοδιασμένου με τις αντίστοιχες μεθόδους

Πχ.:

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

```
my_file.method()
```



Open: Άνοιγμα αρχείου

--2

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

- **filename**: μπορεί να περιλαμβάνει απόλυτο ή σχετικό μονοπάτι (path) προσδιορισμού της θέσης του φυσικού αρχείου
 - Αν δεν προσδιορίσουμε κάποιο μονοπάτι το φυσικό αρχείο θα πρέπει να βρίσκεται στον ίδιο φάκελο με τον πηγαίο κώδικα
- **mode**: υπάρχουν 3 βασικοί τρόποι ανοίγματος αρχείου:
 - 'r' για ανάγνωση δεδομένων από το αρχείο
 - 'w' για εγγραφή δεδομένων στο αρχείο
 - 'a' για προσθήκη (append) δεδομένων στο αρχείο
- **Τύποι αρχείων**: υπάρχουν 2 βασικοί τύποι αρχείων:
 - Κειμένου (text)
 - Δυαδικό (binary)

Operation	Interpretation
<code>output = open(r'C:\spam', 'w')</code>	Create output file ('w' means write)
<code>input = open('data', 'r')</code>	Create input file ('r' means read)
<code>input = open('data')</code>	Same as prior line ('r' is the default)
<code>aString = input.read()</code>	Read entire file into a single string
<code>aString = input.read(N)</code>	Read up to next N characters (or bytes) into a string
<code>aString = input.readline()</code>	Read next line (including \n newline) into a string
<code>aList = input.readlines()</code>	Read entire file into list of line strings (with \n)
<code>output.write(aString)</code>	Write a string of characters (or bytes) into file
<code>output.writelines(aList)</code>	Write all line strings in a list into file
<code>output.close()</code>	Manual close (done for you when file is collected)
<code>output.flush()</code>	Flush output buffer to disk without closing
<code>anyFile.seek(N)</code>	Change file position to offset N for next operation
<code>for line in open('data'): use line</code>	File iterators read line by line
<code>open('f.txt', encoding='latin-1')</code>	Python 3.X Unicode text files (str strings)
<code>open('f.bin', 'rb')</code>	Python 3.X bytes files (bytes strings)
<code>codecs.open('f.txt', encoding='utf8')</code>	Python 2.X Unicode text files (unicode strings)
<code>open('f.bin', 'rb')</code>	Python 2.X bytes files (str strings)

Αρχεία Κειμένου

Text files

Παράδειγμα: Γράψιμο σε Αρχείο κειμένου 'w'

```
# Open file1.txt to write
my_file = open('file1.txt', 'w')
for i in range(10):
    line = 'Line-' + str(i) + '\n'
    my_file.write(line)
my_file.close()
```

- Η μέθοδος **close()** :
- **Μεταφέρει** τυχόν τελευταία δεδομένα και **κλείνει** τη σύνδεση με το φυσικό αρχείο
- **Απελευθερώνει** δεσμευμένους πόρους
- **ΔΕΝ** καταργεί όμως το αντικείμενο `my_file`

- Το φυσικό αρχείο `file1.txt` **ανοίγει για γράψιμο ('w')**
 - Αν δεν υπάρχει δημιουργείται
 - Αν υπάρχει δημιουργείται εκ νέου (χάνονται δεδομένα!)
- Η `open` επιστρέφει το αντικείμενο αρχείου `my_file`
- Καλώντας τη μέθοδο **write** γράφουμε 10 γραμμές (αλφαριθμητικά) στο αρχείο
- Το `'\n'` χρειάζεται γιατί η `write` δεν προσθέτει αυτόματα χαρακτήρα τέλους γραμμής

Παράδειγμα: Διάβασμα από Αρχείο κειμένου 'r'

```
# Open file1.txt to read
my_file = open('file1.txt', 'r')
line = my_file.readline()
print(line)
my_file.close()
```

```
>>>
Line-0
>>>
```

- Το φυσικό αρχείο file1.txt **ανοίγει για διάβασμα ('r')**
- Η open επιστρέφει το αντικείμενο αρχείου my_file
- Η readline() επιστρέφει μια γραμμή κάθε φορά από το αρχείο
- Το '\n' ερμηνεύεται και εκτελείται από την print(line) αλλάζοντας γραμμή

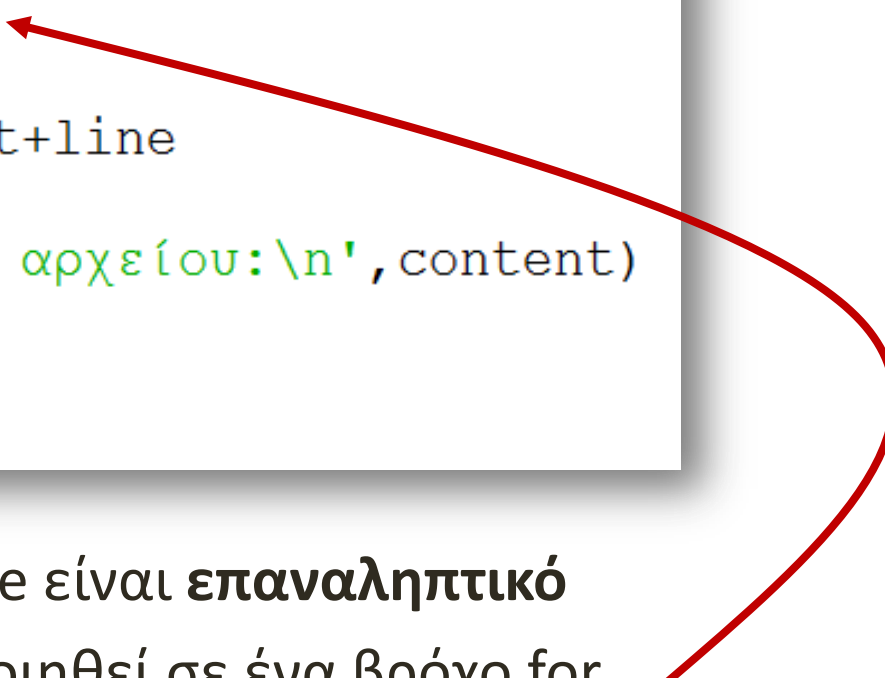
Διάβασμα του περιεχομένου με **for loop**

```
my_file = open('file1.txt', 'r')
content = ''

for line in my_file:
    print(line)
    content = content + line

print('\nΠεριεχόμενο αρχείου:\n', content)

my_file.close()
```



- Το αντικείμενο `my_file` είναι **επαναληπτικό**
- Μπορεί να χρησιμοποιηθεί σε ένα βρόχο `for` ...για την ανάγνωση ολόκληρου του περιεχομένου του αρχείου

Ανάγνωση από Αρχείο κειμένου

Εναλλακτικοί τρόποι

```
aString = my_file.read()
aString = my_file.read(N)
aString = my_file.readline()
aList = my_file.readlines()
```

• **read():** Διαβάζει ολόκληρο το αρχείο σε ένα αλφαριθμητικό

• **read(N):** Διαβάζει N χαρακτήρες σε ένα αλφαριθμητικό

• **readline():** Διαβάζει μία γραμμή σε ένα αλφαριθμητικό

• **readlines():** Διαβάζει ολόκληρο το αρχείο σε μία λίστα (με τα \n)

Διαχειριστής 'with' (context manager)

```
with open('file1.txt', 'w') as f:
    for i in range(10):
        line = 'Line-' + str(i) + '\n'
        f.write(line)
```

- Η δομή **with** 'διαχειρίζεται' όλες τις σχετικές λειτουργίες κατά το άνοιγμα και κλείσιμο αρχείου
- Εξασφαλίζει το κλείσιμο χωρίς να γράψουμε την close()

```
with open('file1.txt', 'r') as f:
    content = f.read()

print(content)
```

Ανάγνωση με read & readlines

```
with open('file1.txt', 'r') as f:  
    content = f.read()  
    print(content)
```

```
with open('file1.txt', 'r') as f:  
    content = f.readlines()  
    print(content)
```

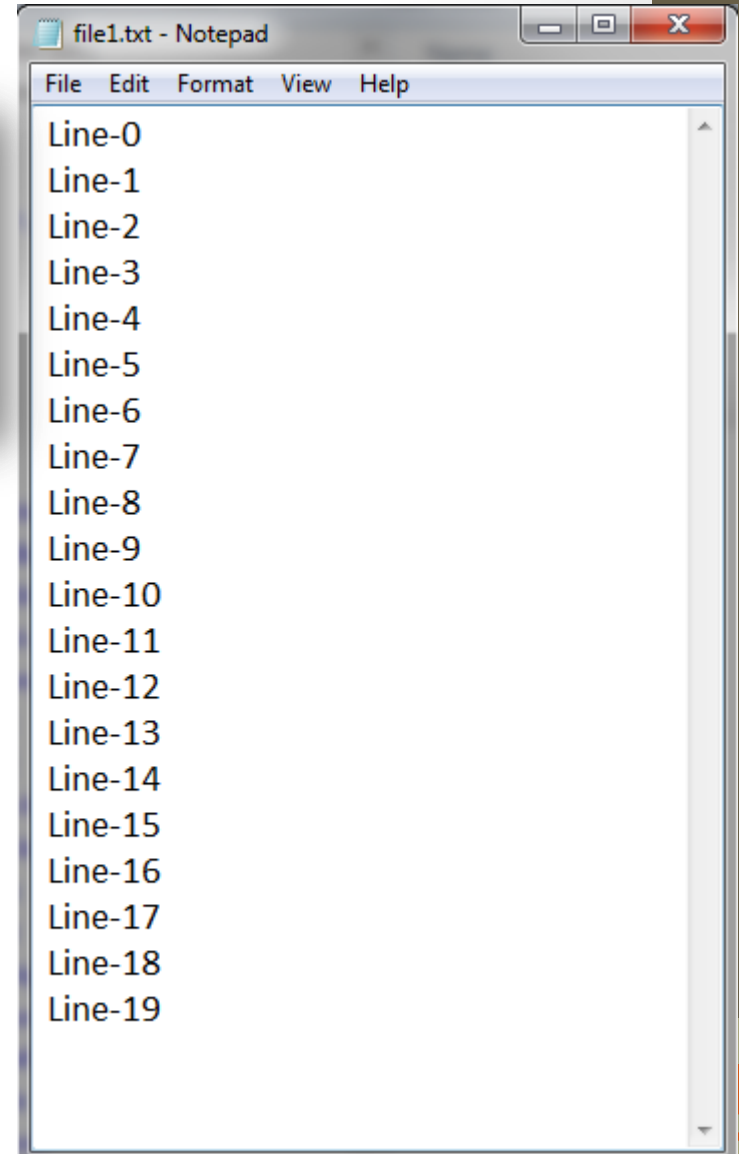
- Δοκιμάστε να τρέξετε τους δύο αυτούς τρόπους ανάγνωσης δεδομένων από το αρχείο κειμένου
- Είναι ισοδύναμοι; Γιατί;
- Η print(lineList) εμφανίζει το παρακάτω – Γιατί;

```
>>>  
['Line-0\n', 'Line-1\n', 'Line-2\n', 'Line-3\n', 'Line-4\n',  
'Line-5\n', 'Line-6\n', 'Line-7\n', 'Line-8\n', 'Line-9\n']
```


Προσθήκη (append) σε αρχείο κειμένου 'a'

```
with open('file1.txt', 'a') as f:  
    for i in range(10,20):  
        line = 'Line-'+str(i)+'\n'  
        f.write(line)
```

- Προσθήκη 10 νέων γραμμών στο αρχείο κειμένου (Line-10 μέχρι και Line-19)
- Ανοίγοντας το αρχείο σε mode 'a' ο δείκτης επόμενης εγγραφής αυτόματα τοποθετείται στο τέλος (στο παράδειγμα στο τέλος της 10^{ης} γραμμής Line-9)



Τυχαία πρόσβαση (random access) σε αρχείο κειμένου

tell() & **seek()** -1/2

```
with open('file1.txt') as f:
    print(f.tell())
    line = f.readline()
    print(line)
    print(f.tell())
```

```
>>>
0
Line-0

8
```

- Η **tell()** επιστρέφει τη θέση του δείκτη ανάγνωσης/εγγραφής στο αρχείο
 - (μετρώντας σε χαρακτήρες)
- Εξηγήστε την έξοδο που παράγει ο κώδικας

Αν δεν δηλώσουμε *mode* εννοείται 'r'

Τυχαία πρόσβαση (random access) σε αρχείο κειμένου

tell() & **seek()**

-2/2

```
with open('file1.txt') as f:
    print(f.tell())
    print(f.readline())
    f.seek(0)
    print(f.readline())
    print(f.tell())
```

- Η **seek()** μετακινεί τον δείκτη ανάγνωσης/εγγραφής στη θέση που καθορίζει το όρισμά της
 - (μετρώντας σε χαρακτήρες)

```
>>>
0
Line-0

Line-0

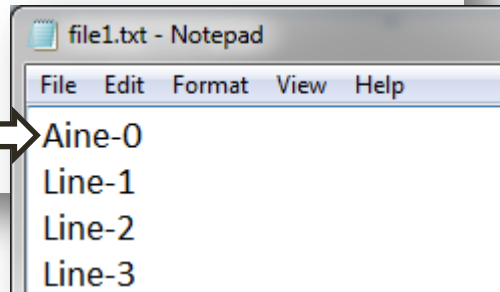
8
```

Παράδειγμα: Πολλαπλοί χειρισμοί

```
with open('file1.txt', 'r+') as f:
    f.seek(8)
    print(f.read(6))
    print(f.tell())

    f.seek(16)
    print(f.read(6))
    print(f.tell())

    f.seek(0)
    f.write('A')
```



• **'r+'** Ανοίγει για ανάγνωση και εγγραφή

- Αν το αρχείο δεν υπάρχει δεν θα δημιουργηθεί
- Ο δείκτης τοποθετείται στην αρχή του αρχείου

• **'a+'** Ανοίγει για ανάγνωση και εγγραφή, όμως:

- Αν το αρχείο δεν υπάρχει τότε δημιουργείται
- Ο δείκτης τοποθετείται στο τέλος του αρχείου
- Η seek() είναι ανενεργός στην περίπτωση αυτή

- Ανοίγοντας ένα αρχείο κειμένου με **'r+'** μπορούμε να γράφουμε και να διαβάζουμε σε όποιο σημείο θέλουμε μετακινώντας τον δείκτη με τη **seek()**

- **Προσοχή:** το γράψιμο νέων δεδομένων σημαίνει ότι διαγράφονται τα παλιά στη θέση εκείνη

Δυαδικά Αρχεία

Binary files

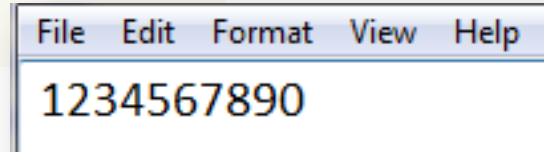
Κειμένου (Text) vs. Δυαδικά (Binary)

- *Κειμένου (Text files):*
 - Ερμηνεύουν τα δεδομένα ως αλφαριθμητικά (strings)
 - Εκτελούν Unicode (απο-) κωδικοποίηση αυτόματα
 - Αναγνωρίζουν και μεταφράζουν το end-of-line εξ ορισμού
- *Δυαδικά (Binary files):*
 - Αναπαριστούν τα δεδομένα ως τύπου **bytes**
 - Δηλ. ακολουθία ακεραίων που αναπαριστούν απόλυτες τιμές bytes
 - Επιτρέπουν στον κώδικα να έχει πρόσβαση στα δεδομένα χωρίς αλλαγές/ερμηνείες

Ανάγνωση δεδομένων από δυαδικό αρχείο

--1/2

Περιεχόμενα του αρχείου
bfile.bin



```
with open('bfile.bin', 'rb') as bf:
    data = bf.read()

print(type(data))
print(data)
print(data[0:2])
print(data[0:2][0])
print(bin(data[0]), bin(data[1]))

for d in data[0:10]:
    print(bin(d), ' ', end='')

print('\n', bin(data[0]) and bin(data[1]))
```

‘read
binary’



Ανάγνωση δεδομένων από δυαδικό αρχείο

--2/2

```
<class 'bytes'>
```

```
b'1234567890'
```

```
b'12'
```

```
49
```

```
0b110001 0b110010
```

```
0b110001 0b110010 0b110011 0b110100 0b110101
```

```
0b110110 0b110111 0b111000 0b111001 0b110000
```

```
0b110010
```

```
with open('bfile.bin', 'rb') as bf:  
    data = bf.read()
```

```
print(type(data))
```

```
print(data)
```

```
print(data[0:2])
```

```
print(data[0:2][0])
```

```
print(bin(data[0]), bin(data[1]))
```

```
for d in data[0:10]:  
    print(bin(d), ' ', end='')
```

```
print('\n', bin(data[0]) and bin(data[1]))
```

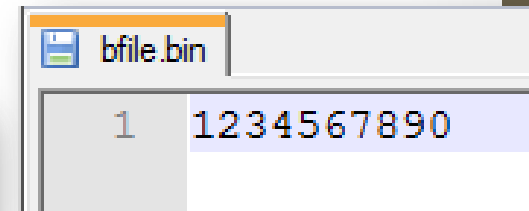

Αλλάζοντας την αναπαράσταση των bytes objects

1/2

- Όταν χρησιμοποιείται η `print()` (ή και `pprint()`) για την εμφάνιση αντικειμένων τύπου **bytes** εμφανίζεται η αλφαριθμητική αναπαράστασή τους (με βάση κάποια κωδικοποίηση (ASCII ή UTF-8) τοποθετώντας μπροστά το πρόθεμα **b'** ώστε να δηλώνεται ο τύπος του αντικειμένου που ακολουθεί
- Πχ. ο κώδικας

```
with open('bfile.txt', 'rb') as binfile:
    data = binfile.read()

print(type(data))
print(data)
```



- Εμφανίζει...

```
<class 'bytes'>
b'1234567890'
```

- Πώς μπορούμε να αλλάξουμε αυτή την αναπαράσταση και να 'δούμε' τα αντικείμενα τύπου **bytes** με διαφορετικό τρόπο;

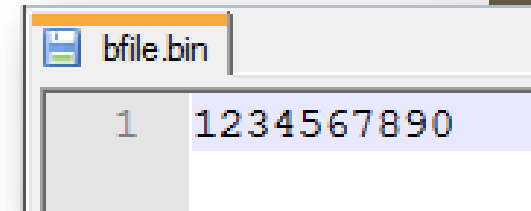
Αλλάζοντας την αναπαράσταση των bytes objects

2/2

- Χρησιμοποιήστε τις μεθόδους **bin()**, **hex()** ή **oct()** για να εμφανίσετε αντίστοιχα τη δυαδική, δεκαεξαδική ή οκταδική αναπαράσταση των bytes
- Πχ. ο κώδικας

```
with open('bfile.txt', 'rb') as binfile:
    data = binfile.read()

for i in data:
    print(bin(i))
```



- Εμφανίζει τη δυαδική αναπαράσταση:

```
>>>
0b110001
0b110010
0b110011
0b110100
0b110101
0b110110
0b110111
0b111000
0b111001
0b110000
```

Διαχείριση δεδομένων από δυαδικό αρχείο – Τύπος **bytearray**

1/2

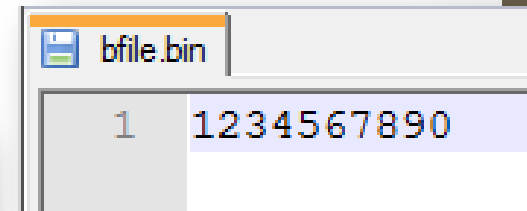
- Όταν διαβάζονται δεδομένα από δυαδικό αρχείο αυτά περνάνε σε ένα αντικείμενο τύπου **bytes** το οποίο όμως είναι **μόνον ανάγνωσης** (read only).
- Για να αλλάξετε τιμές στα δυαδικά δεδομένα μετατρέψτε τα πρώτα σε αντικείμενο **bytearray** όπως δείχνει το παράδειγμα:

```
with open('bfile.bin', 'rb') as binfile:
    data = binfile.read()

bdata = bytearray(data)

print(bdata)
print(type(bdata))
```

```
>>>
bytearray(b'1234567890')
<class 'bytearray'>
>>>
```



Διαχείριση δεδομένων από δυαδικό αρχείο – Τύπος **bytearray**

2/2

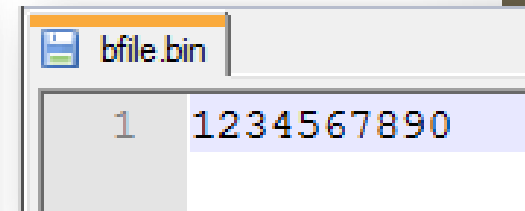
- Τα αντικείμενα **bytearray** δέχονται ανάθεση τιμής απλού δεδομένου (item assignment) και μπορείτε να αλλάξετε τις τιμές των bytes που διαβάστηκαν:

```
with open('bfile.bin', 'rb') as binfile:
    data = binfile.read()

bdata = bytearray(data)
print(bdata[0])

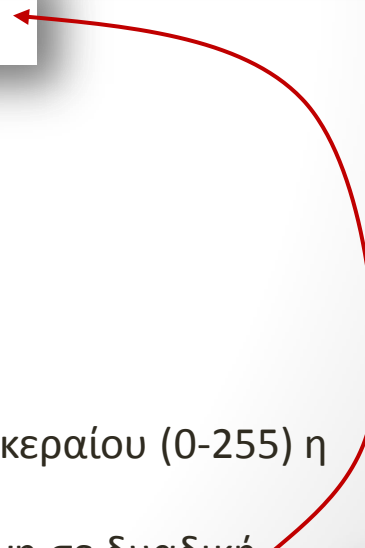
bdata[0] = 65

print(bdata)
```



bfile.bin	
1	1234567890

bdata[0] = 0b00110011



```
49
bytearray(b'A234567890')
```

- Παρατηρήστε ότι για να αλλάξουμε τιμή στο byte περνάμε μια τιμή ακεραίου (0-255) η οποία 'μεταφράζεται' από την print(bdata) σε χαρακτήρα
- Εναλλακτικά μπορούμε να περάσουμε την κατάλληλη τιμή εκφρασμένη σε δυαδική αναπαράσταση



Εγγραφή δεδομένων σε δυαδικό αρχείο 1/2

- Για εγγραφή δεδομένων σε δυαδικό αρχείο θα χρησιμοποιήσουμε πάλι την εντολή `write()` έχοντας φυσικά ανοίξει το αρχείο για εγγραφή, όπως φαίνεται παρακάτω:

```
with open('bfile.txt', 'wb') as binfile:  
    binfile.write(<byte representation>)
```

- Το `<byte representation>` υποδεικνύει πως το περιεχόμενο της `write()` θα πρέπει να είναι σε κάποια μορφή που αναπαριστά bytes
- Δείτε τα παραδείγματα που ακολουθούν

Εγγραφή δεδομένων σε δυαδικό αρχείο 2/2

- Στον παρακάτω κώδικα εγγραφής δεδομένων σε δυαδικό αρχείο:

```
with open('bfile.txt', 'wb') as binfile:  
    binfile.write(abyte)
```

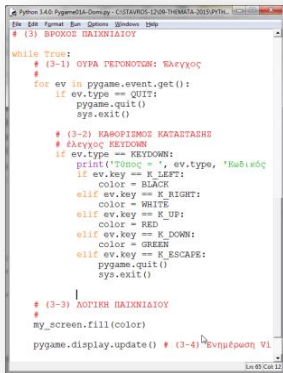
- .. Το αντικείμενο δυαδικών δεδομένων με όνομα `abyte` θα μπορούσε να είναι:
- `abyte = b'Hello'` # δυαδική αναπαράσταση αλφαριθμητικού με κωδικοποίηση ASCII
- `abyte = bytearray(b'Hello')` # το ίδιο όπως πριν σε τύπο `bytearray`
- `abyte = bytearray(u'αβγδ', encoding='UTF-8')` # δυαδική αναπαράσταση αλφαριθμητικού με κωδικοποίηση Unicode
- `abyte = bytearray([65,66,67])` # αντικείμενο `bytearray` δημιουργημένο από λίστα
-

Διατήρηση (pickling)

Αποθήκευση δεδομένων σε αρχεία

pickle (αλλά όχι ... τουρσί)

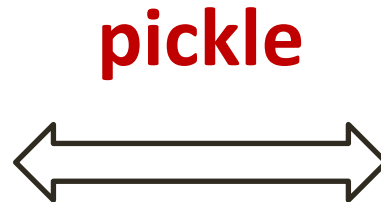
- Η βιβλιοθήκη **pickle** είναι ένα εργαλείο της Python που επιτρέπει την **άμεση αποθήκευση** σε **αρχείο** κάθε δομής δεδομένων χωρίς καμιά άλλη απαίτηση επεξεργασίας από πλευράς προγραμματιστή



```
# (3-1) ΠΡΟΚΟΣ ΠΑΙΧΝΙΔΙΟΥ
while True:
    # (3-1) ΟΤΑ ΓΕΥΟΝΟΣΗ: Έλεγχος
    #
    for ev in pygame.event.get():
        if ev.type == QUIT:
            pygame.quit()
            sys.exit()

    # (3-2) ΕΛΕΓΧΟΣ ΚΑΤΑΣΤΑΣΗΣ
    # Έλεγχος KEYDOWN
    if ev.type == KEYDOWN:
        print("Τύπος = ", ev.type, "Χαράκις")
        if ev.key == K_LEFT:
            color = BLACK
        elif ev.key == K_RIGHT:
            color = WHITE
        elif ev.key == K_UP:
            color = RED
        elif ev.key == K_DOWN:
            color = GREEN
        elif ev.key == K_ESCAPE:
            pygame.quit()
            sys.exit()

    # (3-3) ΛΟΓΙΣΜΟΣ ΠΑΙΧΝΙΔΙΟΥ
    #
    my_screen.fill(color)
    pygame.display.update() # (3-4) Ενημέρωση V1
```



- Η **pickle** περιλαμβάνει μεθόδους οι οποίες:
- Α) κωδικοποιούν τη δομή που θα αποθηκευθεί με **σειριακή** μορφή (data serialization), και..
- Β) μεταφέρουν **προς και από το αρχείο** τη ροή των σειριακών δεδομένων
- Έτσι μια σύνθετη και πολυεπίπεδη δομή δεδομένων (λίστες, λεξικά, λίστες λεξικών, κλπ.) μπορεί να **διατηρείται** (δηλ. να αποθηκεύεται) σε αρχείο **χωρίς να γίνεται χρήση βάσης δεδομένων**


```
import pickle

# data: σύνθετη δομή δεδομένων

data = [{(1,2):[{(3,4):None, 'data':[0]}]}]

with open('datafile.pkl', 'wb') as f:
    pickle.dump(data, f)

with open('datafile.pkl', 'rb') as f:
    data = pickle.load(f)

print(data)
```

- **Μεταφορά** με την `pickle.dump()` του λεξικού `di` προς το αρχείο
- **Επαναφόρτωση** με την `pickle.load()` του λεξικού `di` από το αρχείο

pickle

Παράδειγμα διατήρησης --2

```
import random
import pickle

value_list = [chr(random.randint(65,100)) for i in range(10)]
key_list = [random.randint(65,100) for i in range(10)]
di = {key_list[k]:value_list[k] for k in range(10)}
lista = [di for i in range(5)]

with open('datafile.pkl', 'wb') as f:
    pickle.dump(lista, f)

with open('datafile.pkl', 'rb') as f:
    E = pickle.load(f)

print(E[0])
```

Μεταφορά...

Επαναφόρτωση...



Διαχείριση εξαιρέσεων

Exceptions management

`try..except`

Διαχείριση εξαιρέσεων

- Μια **εξαίρεση (exception)** είναι μια κατάσταση σφάλματος που ανακύπτει την ώρα της εκτέλεσης.
- Η Python διακόπτει την εκτέλεση και αναφέρει ένα σχετικό μήνυμα (*'raise exception' – 'εγείρει ή εκκινεί εξαίρεση'*)
- Παράδειγμα

```
x=int(input('Number: '))  
print(x)
```

- Αν δοθεί είσοδος χαρακτήρας, πχ. 'e'

```
ValueError: invalid literal for int() with base 10: 'e'
```

- Πώς μπορούμε να **ελέγχουμε τις εξαιρέσεις χωρίς διακοπή της εκτέλεσης του κώδικα** ;

try..except..else

```
try:
    # Κώδικας που ελέγχεται για εξαιρέσεις
except:
    # Κώδικας που εκτελείται όταν εμφανιστεί μια εξαίρεση
else:
    # Κώδικας που εκτελείται εφόσον το try-block εκτελεστεί σωστά
```

```
try:
    x=int(input('Ακέραιος: '))
except:
    print('Παρακαλώ πληκτρολογήστε ακέραιο αριθμό')
else:
    print('Αριθμός = ', x)
```

Ελέγξτε το άνοιγμα αρχείων

```
try:
    with open('afile.txt', 'r') as f:
        lines = f.read()
except:
    print('Σφάλμα κατά το άνοιγμα του αρχείου')
else:
    print(lines)
```

Άλλες μορφές της try..except

```
try:
    with open('afile.txt', 'r') as f:
        lines = f.read()
except FileNotFoundError:
    print('Σφάλμα: Δεν υπάρχει τέτοιο αρχείο')
except:
    print('Σφάλμα διαχείρισης του αρχείου')
else:
    print(lines)
```

```
try:
    with open('somefile.txt', 'r') as f:
        lines = f.read()
except FileNotFoundError as err:
    print('Σφάλμα τύπου: ',err,' Δεν υπάρχει τέτοιο αρχείο')
except:
    print('Σφάλμα διαχείρισης του αρχείου')
else:
    print(lines)
```