

Νίκος Μ. Χατζηγιαννάκης

Η γλώσσα **Python** σε βάθος



Περιλαμβάνει εισαγωγή
στην επιστήμη των υπολογιστών
και τον προγραμματισμό

5η εργασία – Κρυφό μήνυμα σε εικόνα



5η Εργασία - Κρυφό μήνυμα σε εικόνα

ΣΚΕΠΤΙΚΟ

Το πρόγραμμα απόκρυψης πρέπει αρχικά να ζητάει από τον χρήστη να πληκτρολογήσει τόσο το όνομα του αρχείου κειμένου όσο και το όνομα του αρχείου εικόνας, μέσα στο οποίο θα κρυφτεί το κείμενο. Στη συνέχεια, το πρόγραμμα θα διαβάσει την κεφαλίδα του αρχείου εικόνας και θα ελέγχει αν το συγκεκριμένο αρχείο έχει τη δυνατότητα απόκρυψης κειμένου. Για να έχει αυτή τη δυνατότητα, πρέπει να είναι τύπου *bmp*, με βάθος χρώματος 24 bit και με τις κατάλληλες διαστάσεις ώστε να διαθέτει *byte* πλήρωσης. Ανάλογα με τις διαστάσεις της εικόνας, υπολογίζουμε τα διαθέσιμα *byte* πλήρωσης και τοποθετούμε τον δείκτη εγγραφής στην αρχή του πρώτου από αυτά τα *byte*. Στη συνέχεια γράφουμε στη θέση αυτή τόσους χαρακτήρες όσο είναι το πλήθος των *byte* πλήρωσης, τους οποίους διαβάζουμε από το αρχείο κειμένου. Κατόπιν, τοποθετούμε τον δείκτη εγγραφής στο επόμενο διαθέσιμο *byte* πλήρωσης, που βρίσκεται στο τέλος της επόμενης σειράς *pixel*. Τα παραπάνω δύο βήματα επαναλαμβάνονται μέχρι να διαβάσουμε όλους τους χαρακτήρες από το αρχείο κειμένου ή μέχρι να εξαντληθούν τα διαθέσιμα *byte* πλήρωσης που βρίσκονται στην εικόνα.



Το πρόγραμμα που θα εξάγει ένα κρυμμένο κείμενο μέσα από μια εικόνα θα λειτουργεί με παρόμοιο τρόπο. Αρχικά ζητάει από τον χρήστη να πληκτρολογήσει το όνομα του αρχείου εικόνας η οποία περιέχει το κρυμμένο κείμενο, και έπειτα διαβάσει την κεφαλίδα του αρχείου εικόνας για να ελέγξει αν το συγκεκριμένο αρχείο έχει τη δυνατότητα απόκρυψης κειμένου. Αν δεν την έχει (δηλαδή, δεν είναι τύπου *bmp* με βάθος χρώματος 24 bit ή δεν διαθέτει *byte* πλήρωσης), εμφανίζει το κατάλληλο μήνυμα και η διαδικασία διακόπτεται. Ακολούθως, τοποθετεί τον δείκτη ανάγνωσης στην αρχή του πρώτου *byte* πλήρωσης και διαβάσει από τη θέση αυτή τόσους χαρακτήρες όσο είναι το πλήθος των *byte* πλήρωσης, τους εμφανίζει στην οθόνη και τους γράφει ταυτόχρονα σε ένα αρχείο εξόδου (το *out.txt*). Στη συνέχεια μετακινούμε τον δείκτη ανάγνωσης στο επόμενο διαθέσιμο *byte* πλήρωσης και διαβάζουμε τους επόμενους κρυμμένους χαρακτήρες. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να διαβάσουμε όλους τους κρυμμένους χαρακτήρες, δηλαδή μέχρι να εντοπίσουμε *byte* πλήρωσης με μηδενική τιμή ή μέχρι να εξαντληθούν όλα τα διαθέσιμα *byte* πλήρωσης της εικόνας.

Η υλοποίηση της εργασίας βασίζεται στην ιδιαίτερη δομή των αρχείων εικόνων τύπου *BMP* η οποία αναλύεται στα παραδείγματα Π11.10 και Π11.11. Ο κώδικας που χρησιμοποιείται είναι μια προσαρμογή του κώδικα των παραπάνω παραδειγμάτων.

Βήματα

- 1 Το πρόγραμμα απόκρυψης αρχικά ζητάει από τον χρήστη να πληκτρολογήσει το όνομα του αρχείου εικόνας μέσα στο οποίο θα κρυφτεί το κείμενο. Στη συνέχεια το πρόγραμμα διαβάζει την κεφαλίδα του αρχείου εικόνας και εμφανίζει κάποια από τα στοιχεία αυτά για να ενημερώσει τον χρήστη.
- 2 Μετά, το πρόγραμμα υπολογίζει το πλήθος των *byte* πλήρωσης (*padding*) που υπάρχουν σε κάθε γραμμή *pixel* της εικόνας, και στη συνέχεια ελέγχει αν το αρχείο είναι κατάλληλο για την απόκρυψη κειμένου. Συγκεκριμένα ελέγχει αν είναι τύπου *bmp* με βάθος χρώματος 24 bit και αν διαθέτει *byte* πλήρωσης. Αν το αρχείο είναι κατάλληλο, υπολογίζει και εμφανίζει το πλήθος των χαρακτήρων που μπορούμε να κρύψουμε μέσα στο αρχείο, διαφορετικά η διαδικασία σταματά και εμφανίζεται το κατάλληλο μήνυμα.
- 3 Στη συνέχεια ζητείται από τον χρήστη να πληκτρολογήσει το όνομα του αρχείου το οποίο περιέχει το κείμενο που θέλουμε να κρύψουμε μέσα στην εικόνα. Το αρχείο αυτό πρέπει να είναι με κωδικοποίηση Greek-ANSI (ISO-8859-7), ώστε κάθε χαρακτήρας είτε είναι ελληνικός είτε λατινικός να μετατρέπεται σε ένα *byte*.

- ❹ Με βάση την απόσταση (offset) στην οποία βρίσκονται τα δεδομένα των pixel της εικόνας, το πλάτος της εικόνας σε pixel (platos) και το πλήθος των byte πλήρωσης (padding), υπολογίζουμε τη θέση στην οποία βρίσκεται το πρώτο byte πλήρωσης κάθε γραμμής (g). Στη συνέχεια ο δείκτης εγγραφής τοποθετείται στη θέση του πρώτου byte πλήρωσης της γραμμής g.
- ❺ Μετά το πρόγραμμα διαβάζει από το αρχείο κειμένου έναν χαρακτήρα και τον γράφει μέσα στο αρχείο εικόνας στη θέση ενός byte πλήρωσης. Αυτό επαναλαμβάνεται τόσες φορές όσα είναι τα byte πλήρωσης κάθε γραμμής. Τα βήματα 4 και 5 επαναλαμβάνονται για κάθε γραμμή pixel της εικόνας, μέχρι να εξαντληθούν οι χαρακτήρες του αρχείου κειμένου ή τα byte πλήρωσης που υπάρχουν στην εικόνα.
- ❻ Τέλος, εμφανίζει το πλήθος χαρακτήρων που έχουν κρυφτεί μέσα στην εικόνα. Στην περίπτωση που τα byte πλήρωσης του αρχείου εικόνας δεν επαρκούν για την αποκρυψη του συνόλου των χαρακτήρων, τότε οι χαρακτήρες ενδέχεται να είναι λιγότεροι από τους χαρακτήρες του αρχείου κειμένου.

Κώδικας – Απόκρυψη κειμένου μέσα σε εικόνα

e5_a.py

```
import sys
```

```
arxio_eikonas=input('Αρχείο εικόνας:')
fp1=open(arxio_eikonas,'rb+') ←
```

❶ Ζητείται από τον χρήστη να πληκτρολογήσει το όνομα του αρχείου εικόνας μέσα στο οποίο θα κρύψουμε το κείμενο. Στη συνέχεια ανοίγει το αρχείο σε δυαδική μορφή για ανάγνωση και εγγραφή. Το αντικείμενο αρχείου ανατίθεται στη μεταβλητή fp1.

```
fp1.seek(2)
size_bytes=fp1.read(4)
size=int.from_bytes(size_bytes,'little')
print('Μεγεθος:',size) ←
```

❷ Διαβάζει και εμφανίζει τα στοιχεία από την κεφαλίδα του αρχείου εικόνας

```
fp1.seek(10)
offset_bytes=fp1.read(4)
offset=int.from_bytes(offset_bytes,'little')
print('Απόσταση δεδομένων pixel (offset):',offset)
```

```
fp1.seek(18)
platos_bytes=fp1.read(4)
ypsos_bytes=fp1.read(4)
platos=int.from_bytes(platos_bytes,'little')
print('Πλάτος:',platos)
ypsos=int.from_bytes(ypsos_bytes,'little')
print('Ύψος:',ypsos)
```

Δοκιμάστε το πρόγραμμα με το αρχείο εικόνας f.bmp και το αρχείο κειμένου tocode.txt που υπάρχουν στον ίδιο φάκελο με τον πηγαίο κώδικα του προγράμματος.

```
bytes_per_row=3*platos
if bytes_per_row%4==0:
    padding=0
else:
    padding=4-bytes_per_row%4
print('Byte πλήρωσης:',padding)
if padding==0:
    print('Συγνώμη αλλά το αρχείο εικόνας δεν διαθέτει byte πλήρωσης\n')
    print('Δοκίμασε εικόνα bmp 999x999 pixels\n')
    sys.exit()
```

❸ Υπολογίζει και εμφανίζει τα διαθέσιμα byte πλήρωσης που υπάρχουν σε κάθε γραμμή pixel της εικόνας.

```
fp1.seek(28)
bits_bytes=fp1.read(2)
bits=int.from_bytes(bits_bytes,'little')
print('Bits ανά pixel:',bits)
if bits!=24:
    print('Συγνώμη αλλά το αρχείο δεν είναι RGB 24bits')
    sys.exit()
```

❹ Στην περίπτωση που το αρχείο δεν διαθέτει byte πλήρωσης ή δεν έχει βάθος χρώματος 24 bit, η διαδικασία σταματά και εμφανίζεται το ανάλογο μήνυμα.

❺ Ανάλογα με τις διαστάσεις της εικόνας και τα byte πλήρωσης ανά γραμμή, υπολογίζει και εμφανίζει το πλήθος των χαρακτήρων που μπορούμε να κρύψουμε σε αυτό το αρχείο.

```
print('--> Στο αρχείο εικόνας μπορούμε να κρύψουμε μέχρι', ypsos*padding, 'χαρακτήρες')
count=0
arxio_keimenoy=input('\nΔώσε το αρχείο κειμένου:')
fp2=open(arxio_keimenoy, 'r', encoding='iso-8859-7')

# Εγγραφή κειμένου στο αρχείο εικόνας
print('\n'+40*'=')
for g in range(ypsos):
    padding_pos=offset+platos*3*(g+1)+padding*g
    fp1.seek(padding_pos)
    for i in range(1,padding+1):
        b=fp2.read(1)
        if b:
            set_byte=bytes(b, 'greek')
            fp1.write(set_byte)
            print(b, end='')
            count=count+1
        else:
            set_byte=bytes(1)
            fp1.write(set_byte)

fp1.close()
fp2.close()
print('\n'+40*'=')
print('Στην εικόνα κρύφτηκαν', count, 'χαρακτήρες !!!')
```

➤ Ζητείται από τον χρήστη να πληκτρολογήσει το όνομα του αρχείου κειμένου. Το αντικείμενο αρχείου ανατίθεται στη μεταβλητή **fp2**.

➤ Υπολογίζει τη θέση του πρώτου byte πλήρωσης της γραμμής **g** της εικόνας..

➤ Τοποθετεί τον δείκτη ανάγνωσης/εγγραφής στη θέση του πρώτου byte πλήρωσης της κάθε γραμμής.

➤ Διαβάζει από το αρχείο κειμένου (fp2) έναν χαρακτήρα και τον γράφει μέσα στο αρχείο εικόνας (fp1) στη θέση ενός byte πλήρωσης. Αυτό γίνεται τόσες φορές όσα είναι τα byte πλήρωσης κάθε γραμμής.

➤ Αν έχουμε φτάσει στο τέλος του αρχείου κειμένου, συνεχίζει να γράφει μηδενικά byte πλήρωσης.

➤ Εμφανίζει το πλήθος των χαρακτήρων που έχουν κρυφτεί μέσα στην εικόνα.

Κώδικας – Ανάκτηση κρυμμένου κειμένου από εικόνα

Για να ανακτήσουμε ένα κρυμμένο κείμενο που βρίσκεται μέσα σε μια εικόνα, ακολουθούμε τα παρακάτω βήματα:

Βήματα

- ➊ Το πρόγραμμα ανάκτησης του κειμένου αρχικά να ζητάει από τον χρήστη να πληκτρολογήσει το όνομα του αρχείου εικόνας η οποία περιέχει το κρυμμένο κείμενο. Στη συνέχεια, το πρόγραμμα διαβάζει την κεφαλίδα του αρχείου εικόνας και εμφανίζει κάποια από τα στοιχεία αυτά για να ενημερώσει τον χρήστη.
- ➋ Έπειτα υπολογίζει το πλήθος των byte πλήρωσης (padding) που υπάρχουν σε κάθε γραμμή pixel της εικόνας και στη συνέχεια ελέγχει αν το αρχείο είναι κατάλληλο για την αποκρυψη κειμένου. Συγκεκριμένα, ελέγχει αν είναι τύπου bmp με βάθος χρώματος 24 bit και αν διαθέτει byte πλήρωσης. Αν το αρχείο δεν είναι κατάλληλο, οπότε δεν είναι δυνατόν να περιέχει κρυμμένο κείμενο, η διαδικασία σταματά και εμφανίζεται ένα κατάλληλο μήνυμα.
- ➌ Κατόπιν, υπολογίζει τη θέση στην οποία βρίσκεται το πρώτο byte πλήρωσης κάθε γραμμής (g) και τοποθετεί εκεί τον δείκτη ανάγνωσης.
- ➍ Μετά, το πρόγραμμα διαβάζει τα byte πλήρωσης, τα οποία περιέχουν τους κρυμμένους χαρακτήρες, τους εμφανίζει στην οθόνη και τους γράφει ταυτόχρονα στο αρχείο εξόδου **out.txt**. Αυτό γίνεται τόσες φορές όσα είναι τα byte πλήρωσης κάθε γραμμής. Τα βήματα 3 και 4 επαναλαμβάνονται για κάθε γραμμή pixel της εικόνας και σταματά όταν διαβάσει μηδενικό byte πλήρωσης ή όταν εξαντληθούν οι γραμμές των pixel της εικόνας.
- ➎ Τέλος, εμφανίζει το πλήθος χαρακτήρων που διάβασε.

e5_b.py

```
import sys
```

```
arxio_eikonas=input('Αρχείο εικόνας:')
fp1=open(arxio_eikonas,'rb')
```

❶ Ζητείται από τον χρήστη να πληκτρολογήσει το όνομα του αρχείου εικόνας μέσα στην οποία είναι κρυμμένο το κείμενο. Στη συνέχεια ανοίγει το αρχείο σε δυαδική μορφή για ανάγνωση. Το αντικείμενο αρχείου ανατίθεται στη μεταβλητή **fp1**.

```
fp1.seek(2)
size_bytes=fp1.read(4)
size=int.from_bytes(size_bytes,'little')
print('Μεγεθος:',size)
```

Δοκιμάστε το πρόγραμμα με το αρχείο εικόνας f.bmp.

```
fp1.seek(10)
offset_bytes=fp1.read(4)
offset=int.from_bytes(offset_bytes,'little')
print('Απόσταση δεδομένων pixel (offset):',offset)
```

❷ Διαβάζει και εμφανίζει τα στοιχεία από την κεφαλίδα του αρχείου εικόνας

```
fp1.seek(18)
platos_bytes=fp1.read(4)
ypsos_bytes=fp1.read(4)
platos=int.from_bytes(platos_bytes,'little')
print('Πλάτος:',platos)
ypsos=int.from_bytes(ypsos_bytes,'little')
print('Ύψος:',ypsos)
```

```
bytes_per_raw=3*platos
```

```
if bytes_per_raw%4==0:
```

```
padding=0
```

```
else:
```

```
padding=4-bytes_per_raw%4
```

```
print('Byte πλήρωσης:',padding)
```

```
if padding==0:
```

```
print('Συγνώμη αλλά το αρχείο εικόνας δεν διαθέτει byte πλήρωσης')
```

```
print('Οπότε δεν μπορεί να περιέχει κρυμμένο κείμενο')
```

```
sys.exit()
```

❸ Υπολογίζει και εμφανίζει τα διαθέσιμα byte πλήρωσης που υπάρχουν σε κάθε γραμμή pixel της εικόνας.

```
fp1.seek(28)
bits_bytes=fp1.read(2)
bits=int.from_bytes(bits_bytes,'little')
print('Bits ανά pixel:',bits)
```

```
if bits!=24:
```

```
print('Συγνώμη αλλά το αρχείο δεν είναι RGB 24bits')
```

```
sys.exit()
```

❹ Στην περίπτωση που το αρχείο δεν διαθέτει byte πλήρωσης ή δεν έχει βάθος χρώματος 24 bit, η διαδικασία σταματά και εμφανίζεται το ανάλογο μήνυμα.

```
fp2=open('out.txt','w',encoding='iso-8859-7')
```

```
count=0
```

```
print('\n'+40*'=')
```

```
for g in range(ypsos):
```

```
padding_pos=offset+platos*3*(g+1)+padding*g
```

```
fp1.seek(padding_pos)
```

```
for i in range(1,padding+1):
```

```
b=fp1.read(1)
```

```
s=b.decode('iso-8859-7')
```

```
if s!='\x00':
```

```
print(s,end='')
```

```
count=count+1
```

```
else:
```

```
break
```

❺ Υπολογίζει τη θέση του πρώτου byte πλήρωσης της γραμμής **g** της εικόνας..

❻ Τοποθετεί τον δείκτη ανάγνωσης/εγγραφής στη θέση του πρώτου byte πλήρωσης της κάθε γραμμής.

```
fp1.close()
```

```
fp2.close()
```

```
print('\n'+40*'=')
```

```
print('Διαβάστηκαν',count,'χαρακτήρες και όλο το κείμενο βρίσκεται στο αρχείο out.txt')
```

❼ Διαβάζει από το αρχείο εικόνας (fp1), και συγκεκριμένα από τη θέση ενός byte πλήρωσης, έναν χαρακτήρα τον εμφανίζει στην οθόνη και τον γράφει στο αρχείο εξόδου (fp2). Αυτό γίνεται τόσες φορές όσα είναι τα byte πλήρωσης κάθε γραμμής. Σταματά όταν διαβάσει μηδενικό byte πλήρωσης ή όταν εξαντληθούν οι γραμμές των pixel της εικόνας.

❽ Εμφανίζει το πλήθος των κρυμμένων χαρακτήρων που έχουν εξαχθεί από το αρχείο εικόνας.

Προτάσεις

Η εργασία αυτή επιδέχεται πάρα πολλές τροποποιήσεις ανάλογα με τη δομή των αρχείων εικόνων. Μια ιδέα θα ήταν το κείμενο να μην κρύβεται μέσα στα byte πλήρωσης αλλά μέσα στα δεδομένα των pixel της εικόνας. Χρησιμοποιώντας την τεχνική του παραδείγματος Π11.11, θα μπορούσαμε να κρύψουμε μέσα σε κάθε pixel μιας εικόνας με βάθος χρώματος 24 bit τρεις χαρακτήρες. Αυτό θα έδινε τη δυνατότητα απόκρυψης πολύ περισσότερων χαρακτήρων. Για παράδειγμα, σε μια εικόνα διαστάσεων 1000x1000 pixel θα μπορούσαν να κρυφτούν μέχρι 3.000.000 χαρακτήρες!

Η τεχνική που προτείνεται είναι να εκτελείται η πράξη XOR (^) των byte που συνθέτουν το κάθε pixel με τους χαρακτήρες που θέλουμε να κρύψουμε, ένα προς έναν με τη σειρά. Φυσικά η εικόνα θα επηρεαζόταν. Ο παραλήπτης έπειτα θα λάμβανε δύο εικόνες, την αρχική και αυτή που προέκυψε μετά από την απόκρυψη των χαρακτήρων. Έπειτα θα εφαρμοζόταν πάλι η πράξη XOR (^) μεταξύ των byte των αντίστοιχων pixel των δύο εικόνων και θα εξαγόταν οι κρυμμένοι χαρακτήρες!