ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Εισαγωγή στους Υπολογιστές

Εργαστήριο 7

Καθηγητές: Αβούρης Νικόλαος, Παλιουράς Βασίλης, Κουκιάς Μιχαήλ, Σγάρμπας Κυριάκος

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

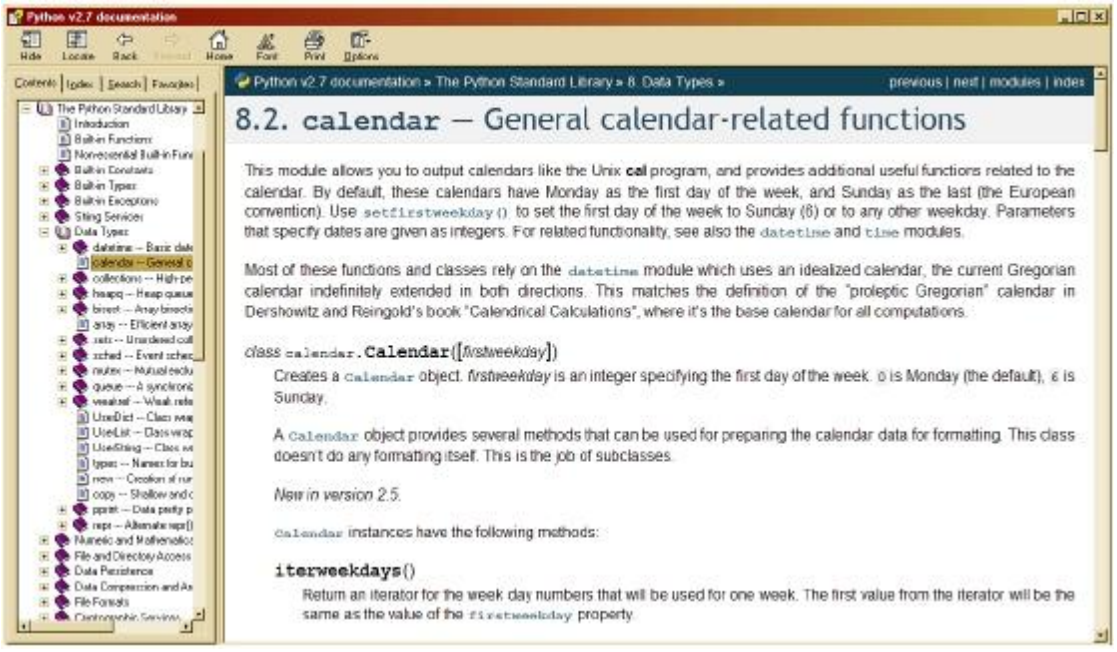
Εργαστήριο 7: Τέταρτη Άσκηση Προγραμματισμού Python

7.1 Γενικά

Η άσκηση αυτή αφορά **δημιουργία και διαχείριση αρχείων** στη γλώσσα Python, **κλήσεις στο λειτουργικό σύστημα** και **αποστολή emails**, με χρήση κατάλληλων βιβλιοθηκών συναρτήσεων.

7.2 Ημερολόγια και Αρχεία

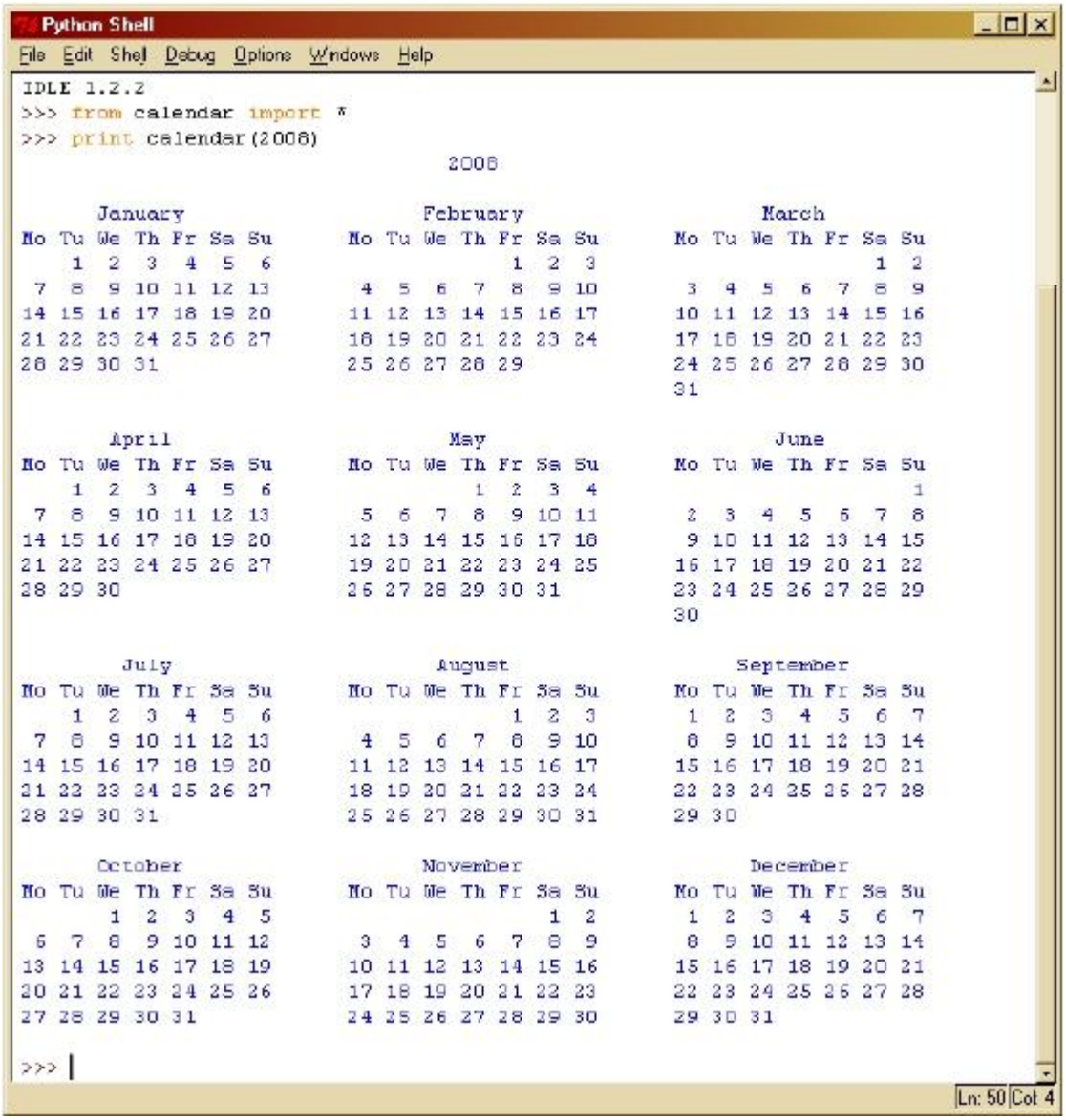
Έστω ότι θέλουμε να τυπώσουμε ένα ημερολόγιο (π.χ. του έτους 2008). Κανονικά θα έπρεπε να υπολογίσουμε πόσες μέρες έχει κάθε μήνας, αν το έτος είναι δίσεκτο, ποιο όνομα ημέρας αντιστοιχεί σε κάθε ημερομηνία κλπ. Όμως η Python διαθέτει βιβλιοθήκη calendar η οποία περιέχει συναρτήσεις για όλα τα παραπάνω, όπως μπορείτε να δείτε στο on-line-help (πατώντας F1 και μετά επιλέγοντας "Library Reference > Data Types >calendar"):



Επιπλέον, η βιβλιοθήκη διαθέτει και την ομώνυμη συνάρτηση calendar() η οποία εμφανίζει ένα πλήρες ημερολόγιο μόνο με μια εντολή. Δηλαδή αν γράψουμε:

```
>>> from calendar import *
>>> print calendar(2008)
```

... θα δούμε κάτι τέτοιο:



Αξίζει να επισημάνουμε ότι αυτό που επιστρέφει η συνάρτηση calendar() είναι ένα αλφαριθμητικό (string). Εμείς επιλέξαμε να το τυπώσουμε με την print. Θα μπορούσαμε να το βάλουμε σε μια μεταβλητή, πχ.:

```
>>> a=calendar(2008)
>>> print a
```

... ή να το σώσουμε σε ένα αρχείο και να το τυπώσουμε στον εκτυπωτή ή να το στείλουμε με email σε κάποιο φίλο μας. Όλα αυτά (και πολλά περισσότερα) μπορούμε να τα κάνουμε μέσα από την Python. Στη συνέχεια θα δούμε πως γράφουμε πληροφορίες σε αρχεία, τα οποία στη συνέχεια μπορούν να διαβαστούν από άλλες εφαρμογές.

Για να δημιουργήσουμε ένα αρχείο θα πρέπει πρώτα να γνωρίζουμε τον τρόπο με τον οποίο κωδικοποιούνται τα δεδομένα σε αυτό. Για παράδειγμα, ένα αρχείο PDF έχει μια επικεφαλίδα στην οποία έχει καταγραφεί η εφαρμογή με την οποία δημιουργήθηκε, η ημερομηνία, το όνομα του χρήστη και διάφορα χαρακτηριστικά (πχ. αν επιτρέπεται η εκτύπωση του αρχείου ή όχι, κλπ). Στη συνέχεια υπάρχει μια σειρά από bytes στα οποία καταγράφονται οι γραμματοσειρές (fonts) που χρησιμοποιούνται μέσα στο αρχείο, μετά ακολουθούν οι σελίδες με πληροφορίες για τις διαστάσεις

(και ενδεχομένως το χρώμα) της κάθε μιας, το κείμενο της κάθε σελίδας κι αν μέσα στο κείμενο περιέχονται και εικόνες, υπάρχει επιπλέον κωδικοποιημένη πληροφορία για την κάθε μια.

Συνεπώς για να δημιουργήσουμε ένα αρχείο κάποιας σύνθετης εφαρμογής χρειάζεται να γνωρίζουμε όλες αυτές τις λεπτομέρειες (οι οποίες για τα ανοικτού τύπου αρχεία είναι ελεύθερα διαθέσιμες στο διαδίκτυο) ή να φορτώσουμε κάποια κατάλληλη βιβλιοθήκη η οποία θα αναλάβει να απλοποιήσει τη διαδικασία για μας.

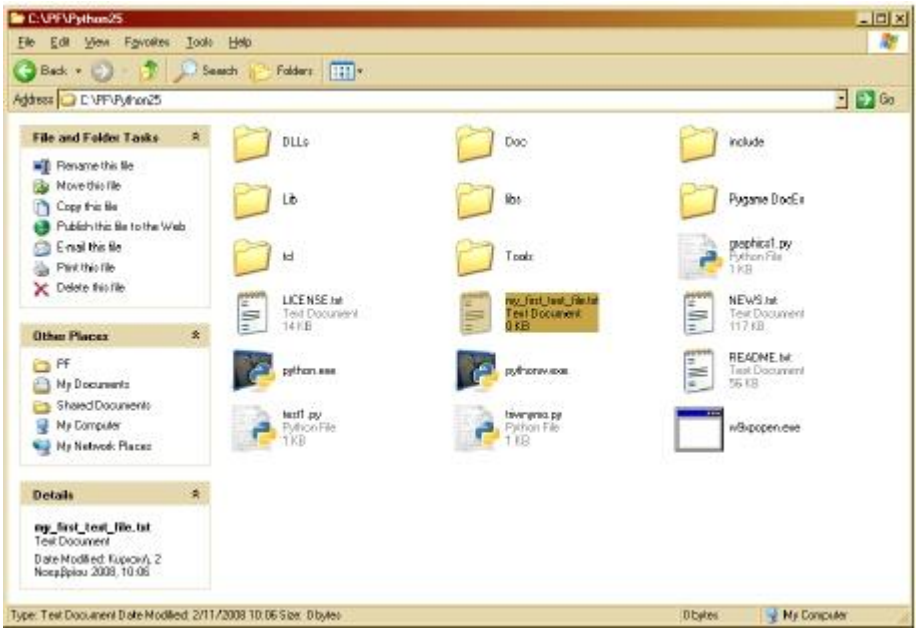
Τα αρχεία με την πιο απλή κωδικοποίηση είναι τα αρχεία κειμένου (.txt), τα οποία δημιουργούνται και διαβάζονται από διορθωτές κειμένου (text editors). Τα αρχεία αυτά δεν έχουν επικεφαλίδα και κάθε byte τους αντιστοιχεί σε έναν ASCII χαρακτήρα. Για να δημιουργήσουμε ένα αρχείο κειμένου με την Python χρησιμοποιούμε τη συνάρτηση open(), ως εξής:

```
>>> my_file=open("my_first_text_file.txt","w")
```

Καταρχήν η συνάρτηση open() είναι ενσωματωμένη στην Python και δε χρειάζεται να φορτώσουμε κάποια βιβλιοθήκη. Δέχεται δυο παραμέτρους:

- Το όνομα του αρχείου ("my_first_text_file.txt) είναι το όνομα με το οποίο αναγνωρίζει το λειτουργικό σύστημα το αρχείο μας.
- Έναν κωδικό που καθορίζει αν το αρχείο πρόκειται να το χρησιμοποιήσουμε για εγγραφή ή για ανάγνωση ("w" για εγγραφή, "r" για ανάγνωση)

Η μεταβλητή my_file είναι το όνομα με το οποίο θα αναγνωρίζει η Python το αρχείο. Επειδή τα πραγματικά ονόματα των αρχείων συχνά είναι μεγάλα και σύσχρηστα, συνηθίζουμε να χρησιμοποιούμε απλούστερα μνημονικά ονόματα μέσα στον κώδικά μας. Μόλις εκτελεστεί η open() δημιουργείται το αρχείο και μπορούμε να το δούμε από το λειτουργικό σύστημα:



Εδώ βλέπουμε δύο πράγματα:

- Το αρχείο έχει δημιουργηθεί στον κατάλογο (φάκελο) που είναι εγκατεστημένη η Python. Αν θέλουμε να δημιουργηθεί κάπου αλλού θα πρέπει να δώσουμε το πλήρες path μαζί με το όνομα του αρχείου στην πρώτη παράμετρο της open().
- Το αρχείο φαίνεται ότι έχει μέγεθος 0 bytes. Απόλυτα λογικό, αφού δεν έχουμε γράψει τίποτα ακόμη σε αυτό.

Για να γράψουμε κάτι, χρησιμοποιούμε τη μέθοδο write(). Ως μέθοδος, η write() επιδρά στη μεταβλητή του αρχείου και δέχεται μια παράμετρο (πχ. αλφαριθμητικό, μια μεταβλητή, κλπ), τα περιεχόμενα της οποίας θα γραφτούν στο αρχείο. Μπορούμε να τη χρησιμοποιήσουμε όσες φορές θέλουμε, διαδοχικά:

```
>>> my_file.write("Τν παξαθάησ εκεξνιόγην έρεη δεκηνπξγεζεί κε ηε γηώδδα Python:\n\n")
>>> a=calendar(2008)
>>> my_file.write(a)
>>> my_file.write("\n\nPython: Κάλεη ηα πάληα...\n... αμζεί λα μέμεην πώο.\n(Read the
Manual!!!)")
```

Το "\n" μέσα στα αλφαριθμητικά προκαλεί αλλαγή γραμμής (σα να έχουμε πατήσει το πλήκτρο "Enter/Return").

Εναλλακτικά, αντί για τη μέθοδο write() μπορούμε να χρησιμοποιήσουμε την εντολή print, με τη σύνταξη:

```
>>> print >>my_file, a
```

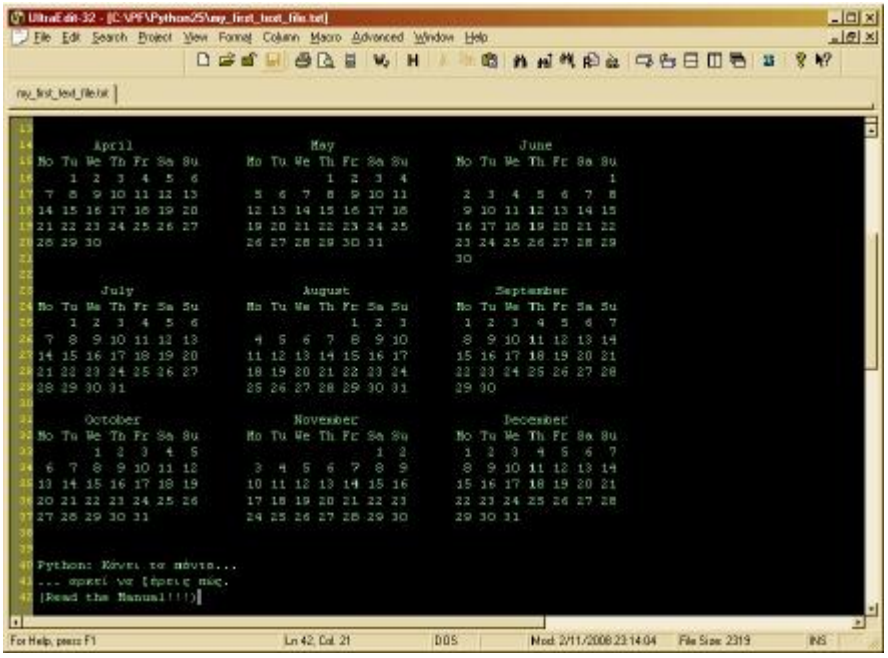
Δουλεύει όπως η write() με τη διαφορά ότι δε χρειάζεται να βάζουμε τους χαρακτήρες "\n" για να τυπωθούν οι αλλαγές γραμμής στο τέλος των strings. Για παράδειγμα, τα περιεχόμενα του ίδιου αρχείου με εντολές print μπορούν να γραφτούν ως εξής:

```
>>> print >>my_file, "Τν παξαθάησ εκεξνιόγην έρεη δεκηνπξγεζεί κε ηε γηώδδα Python:"
>>> print >>my_file, ""
>>> a=calendar(2008)
>>> print >>my_file, a
>>> print >>my_file, ""
>>> print >>my_file, "Python: Κάλεη ηα πάληα... "
>>> print >>my_file, "... αμζεί λα μέμεην πώο. "
>>> print >>my_file, "Read the Manual!!!)"
```

Με όποιον τρόπο κι αν γράψουμε το αρχείο, μόλις τελειώσουμε τις εγγραφές χρειάζεται να καλέσουμε τη μέθοδο close() ώστε να κλείσει το αρχείο, δηλαδή να γραφτούν όλα τα περιεχόμενα της RAM στο σκληρό δίσκο.

```
>>> my_file.close()
```

Και τώρα μπορούμε να το διαβάσουμε με οποιονδήποτε editor:



Φυσικά, μπορούμε να το διαβάσουμε και μέσα από την Python, ως εξής:

```
>>> my_file=open("my_first_text_file.txt","r")
```

Τώρα η open() ανοίγει το αρχείο με την παράμετρο "r", που σημαίνει ότι επιτρέπει μόνο την ανάγνωσή του. Στη συνέχεια η μέθοδος read() μεταφέρει όλα τα περιεχόμενα στη μεταβλητή contents, την οποία μετά χειριζόμαστε όπως θέλουμε:

```
>>> contents=my_file.read()
>>> print contents
```

Προσοχή, αυτή η διαδικασία λειτουργεί σωστά μόνο για αρχεία κειμένου. Αν το αρχείο περιέχει άλλα δεδομένα χρειαζόμαστε κατάλληλη βιβλιοθήκη για να το διαβάσουμε. Μόλις πάψουμε να χρειαζόμαστε άλλο το αρχείο, το κλείνουμε με τη μέθοδο close():

```
>>> my_file.close()
```

7.3 Κλήσεις στο Λειτουργικό Σύστημα

Σύμφωνα με όσα είπαμε παραπάνω, για να αντιγράψουμε ένα αρχείο μπορούμε να κάνουμε κάτι τέτοιο:

```
>>> file1=open("my_first_text_file.txt","r")
>>> file2=open("my_second_text_file.txt","w")
>>> contents=file1.read()
>>> file2.write(contents)
>>> file2.close()
>>> file1.close()
```

Όμως είναι ακόμα καλύτερα να χρησιμοποιήσουμε τις συναρτήσεις της βιβλιοθήκης os, που δίνει πρόσβαση σε κλήσεις του λειτουργικού συστήματος:

```
>>> from os import *
```

Η βιβλιοθήκη os περιέχει χρήσιμες συναρτήσεις, όπως η getcwd() με την οποία βλέπουμε σε ποιον κατάλογο (φάκελο) δουλεύουμε:

```
>>> print getcwd()
C:\PF\Python25
```

... η listdir() που μας δείχνει σε μια λίστα τα περιεχόμενα ενός καταλόγου:

```
>>> d=getcwd()
>>> print listdir(d)
['DLLs', 'Doc', 'graphics1.py', 'include', 'Lib', 'libs', 'LICENSE.txt', 'my_first_text_file.txt',
'my_second_text_file.txt', 'NEWS.txt', 'Pygame DocEx', 'python.exe', 'pythonw.exe', 'README.txt'
,'tcl', 'test1.py', 'Tools', 'triwnymo.py', 'w9xpopen.exe']
```

... όμως ίσως η πιο χρήσιμη συνάρτηση στον os είναι η system(), η οποία μας δίνει πρόσβαση σε όλες τις εντολές του Command Line (DOS prompt). Οι εντολές δίνονται σε ένα αλφαριθμητικό ως παράμετρος στην system(). Πχ:

```
>>> system("copy my_first_text_file.txt my_third_text_file.txt")
0
>>> system("copy my_third_text_file.txt my_forth_text_file.txt")
0
```

Μετονομασία:

```
>>> system("ren my_forth_text_file.txt my_fourth_text_file.txt")
0
```

Διαγραφή:

```
>>> system("del my_third_text_file.txt")
0
```

Όλες αυτές οι συναρτήσεις επιστρέφουν 0 όταν έχουν εκτελεστεί επιτυχώς.

7.4 Συμπίεση Αρχείων

Με άλλες βιβλιοθήκες της Python μπορούμε να χειριστούμε αρχεία ειδικών τύπων. Για παράδειγμα, οι βιβλιοθήκες zipfile και zlib επιτρέπουν τη δημιουργία και ανάγνωση συμπιεσμένων αρχείων. Να πως συμπιέζουμε αρχεία στα προγράμματά μας:

```
>>> from zipfile import *
>>> from zlib import *
>>> fz=ZipFile("AllFiles.zip","w",ZIP_DEFLATED)
```

Η μέθοδος ZipFile() λειτουργεί όπως η open(). Εδώ ανοίξαμε ένα αρχείο με όνομα "AllFiles.zip" για εγγραφή ("w"). Η σταθερά ZIP_DEFLATED δηλώνει τον αλγόριθμο συμπίεσης που θα χρησιμοποιηθεί. Αν την παραλείψουμε, θα γίνει καταχώρηση των αρχείων χωρίς συμπίεση. Στη συνέχεια γράφουμε μέσα στο fz όσα αρχεία θέλουμε να περιέχει:

```
>>> fz.write("my_first_text_file.txt")
>>> fz.write("my_second_text_file.txt")
>>> fz.write("my_fourth_text_file.txt")
```

...και κλείνουμε το αρχείο:

```
>>> fz.close()
```

Τώρα το συμπιεσμένο αρχείο AllFiles.zip μπορεί να διαβαστεί από οποιαδήποτε εφαρμογή διαχειρίζεται συμπιεσμένα αρχεία:

7.5 Αποστολή e-mail

Τα αρχεία κειμένου που δημιουργούμε μπορούμε να τα στέλνουμε με e-mail μέσα από την Python, χωρίς να μεσολαβεί κάποια εφαρμογή αποστολής e-mail. Η βιβλιοθήκη που χρησιμοποιούμε λέγεται smtplib και όλες οι δυνατότητές της εξηγούνται στο help της Python. Εδώ χρησιμοποιούμε μόνο όσες μεθόδους χρειάζονται για να στείλουμε ένα απλό e-mail:

```
>>> from smtplib import *
>>> server=SMTP("patreas.upatras.gr")
```

Η μέθοδος SMTP() παίρνει ως παράμετρο το όνομα του server ο οποίος θα στείλει το e-mail μας. Στη συνέχεια, βάζουμε σε μια μεταβλητή το μήνυμά μας. Πχ. Αν έχει γραφτεί σε κάποιο αρχείο:

```
>>> file=open("my_first_text_file.txt","r")
>>> message=file.read()
>>> file.close()
```

... ή, αν το γράφουμε κατευθείαν:

```
>>> message="""Απηό είλαη έλα e-mail πνπ ζα ζηαιεί
από ηνλ: ecexxxx@ece.upatras.gr
ζηνλ: ecexxxx@ece.upatras.gr
κέζσ ηεο γηώζζαο Python
κε ηε βηβηηνζήθε smtplib."""
```

Προσέξτε ότι μπορούμε να γράψουμε ένα αλφαριθμητικό που επεκτείνεται σε πολλές γραμμές, αν το κλείσουμε σε τριπλά εισαγωγικά. Η Python θα διατηρήσει τη μορφοποίησή του.

Και μετά η μέθοδος sendmail() αναλαμβάνει να στείλει το e-mail:

```
>>> server.sendmail("ecexxxx@ece.upatras.gr", "ecexxxx@ece.upatras.gr", message)
```

Η πρώτη παράμετρος είναι το email του αποστολέα, η δεύτερη του παραλήπτη και η τρίτη το μήνυμα. Με την εκτέλεση δε θα δούμε κάποιο μήνυμα (εκτός αν γράψουμε κάτι λάθος). Αν θέλουμε να διαβάσουμε τα ενδιάμεσα ενημερωτικά μηνύματα του server, πριν την sendmail() πρέπει να δώσουμε:

```
>>> server.set_debuglevel(1)
```

Και τότε μόλις δώσουμε την sendmail() θα δούμε κάτι τέτοιο:

```
send: 'ehlo think.wcl.ee.upatras.gr\r\n'
reply: '250-patreas.upatras.gr patreas.upatras.gr Universty of Patras\r\n'
reply: '250-PIPELINING\r\n'
reply: '250-SIZE 10000000\r\n'
reply: '250 8BITMIME\r\n'
reply: retcode (250); Msg: patreas.upatras.gr patreas.upatras.gr Universty of Patras
PIPELINING
SIZE 10000000
8BITMIME
send: 'mail FROM:<sender@upatras.gr> size=400\r\n'
reply: '250 ok\r\n'
reply: retcode (250); Msg: ok
send: 'rcpt TO:<recipient@upatras.gr>\r\n'
reply: '250 ok\r\n'
reply: retcode (250); Msg: ok
send: 'data\r\n'
reply: '354 Please start mail input.\r\n'
reply: retcode (354); Msg: Please start mail input.
data: (354, 'Please start mail input.')
send: 'test\r\n.\r\n'
reply: '250 Mail queued for delivery.\r\n'
reply: retcode (250); Msg: Mail queued for delivery.
data: (250, 'Mail queued for delivery.')
{}
```

Σε κάθε περίπτωση, μόλις τελειώσουμε την αποστολή των email θα πρέπει να κλείσουμε τη σύνδεσή μας με την εντολή:

```
>>> server.quit()
send: 'quit\r\n'
reply: '221 Closing connection. Good bye.\r\n'
reply: retcode (221); Msg: Closing connection. Good bye.
```

7.6 Ασκήσεις

Αφού τρέξετε τα παραδείγματα των ενοτήτων που προηγήθηκα, εκπονήστε τις παρακάτω ασκήσεις. Αναρτήστε τα αρχεία σας (προγράμματα και αποτελέσματα) ως ένα συμπιεσμένο αρχείο.

Άσκηση #1

Γράψτε πρόγραμμα σε Python που θα αποθηκεύει όλα τα ετήσια ημερολόγια του 21ου αιώνα (δηλαδή από το έτος 2001 έως το 2100) σε ξεχωριστά αρχεία με ονόματα της μορφής xxxx.txt, όπου xxxx το έτος (πχ. 2001.txt, 2002.txt κλπ). Στη συνέχεια, τα 100 αρχεία που θα δημιουργηθούν θα τα συμπιέζει σε ένα αρχείο ZIP με όνομα 21century.zip και μετά θα τα σβήνει από τον σκληρό δίσκο.

Άσκηση #2

Η ημερομηνία της Κυριακής του Πάσχα για τους ορθόδοξους είναι 3 Απριλίου + p ημέρες, όπου το p εξαρτάται από το έτος και υπολογίζεται από την εξής διαδικασία:

$p = v_1+v_2$
$v_1 = (6*v_2+m_4+m_2) \bmod 7$
$v_2 = (16+m_{19}) \bmod 30$
$m_2 = 2*(\text{ΕΤΟΣ} \bmod 4)$
$m_4 = 4*(\text{ΕΤΟΣ} \bmod 7)$
$m_{19} = 19*(\text{ΕΤΟΣ} \bmod 19)$

Με (x mod y) συμβολίζεται το υπόλοιπο της διαίρεσης x/y. Με βάση τα παραπάνω γράψτε πρόγραμμα σε Python που θα υπολογίζει τις ημερομηνίες του Πάσχα για κάθε έτος του 21ου αιώνα και θα τις στέλνει με ένα email στο λογαριασμό σας. Το κείμενο του email θα είναι της μορφής:

Η ημερομηνία του Πάσχα για το 2001 είναι 15 Απριλίου.
Η ημερομηνία του Πάσχα για το 2001 είναι 15 Απριλίου.
Η ημερομηνία του Πάσχα για το 2001 είναι 15 Απριλίου.
...

Υπόδειξη: Αν στα προγράμματα σας φορτώνεται πολλές βιβλιοθήκες, ενδεχομένως κάποια ονόματα συναρτήσεων/μεθόδων να συμπίπτουν. Αν διαπιστώσετε κάτι τέτοιο τότε φορτώστε τις βιβλιοθήκες με την εντολή "import libraryname", αντι για "from libraryname import *". Στη συνέχεια, για να προσδιορίσετε τη συνάρτηση, αντί να γράφετε σκέτο το όνομά της, θα γράφετε (με dot notation) libraryname.function().

Σημειώματα

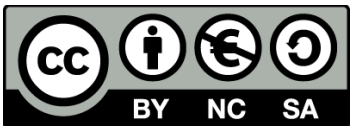
Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

- Έκδοση 1.0 διαθέσιμη εδώ.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης



Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

Εικόνες: Προέρχονται από Python IDLE.

Πίνακες

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.