Εισαγωγή στον Προγ/μό Υπολογιστών

Διάλεξη 12

Ρεύματα *(streams),* αρχεία, επικοινωνία υπολογιστών

Περιεχόμενα

Ρεύματα (*streams*)

Streams

- **Ρεύματα (*streams*):** *ακολουθίες* όπου τα στοιχεία προσπελαύνονται στη σειρά με *lazy evaluation*
  - Πχ, *generators*

- Χρήσιμα στην
  - ανάγνωση/αποθήκευση αρχείων δεδομένων
  - ανάγνωση/εγγραφή σε συσκευές εισόδου/εξόδου
  - αποστολή/λήψη δεδομένων από άλλους υπολογιστές
  - αναπαράσταση ακολουθιών με άπειρα ή πάρα πολλά στοιχεία
  - επεξεργασία δεδομένων μεγάλου όγκου/ρυθμού παραγωγής

- **Επεξεργασία ρευμάτων (*stream processing)*:** τρόπος υπολογισμού που βασίζεται σε ρεύματα

Streams

- Βασική λειτουργία: προσπέλαση επόμενου στοιχείου
- Παράδειγμα: *generator* ακέραιων που αρχίζουν από `start`:

```
>>> def integers(start):
        x = start
        while True:
            yield x
            x = x + 1

>>> integer_stream = integers(1)
>>> next(integer_stream)
1
>>> next(integer_stream)
2
```

Streams

- Επεξεργασία ρευμάτων: νέα ρεύματα μπορούν να παραχθούν από άλλα, χωρίς να έχουν υπολογιστεί τα στοιχεία τους

  – Πχ, αφαίρεση πολλαπλασίων του `k` από την ακολουθία `stream`

```
>>> def remove_multiples(k, stream):
        return filter(lambda x: x % k != 0, stream)

>>> odd_stream = remove_multiples(2, integers(1))
>>> next(odd_stream)
1
>>> next(odd_stream)
3
>>> next(odd_stream)
5
```

Streams

- Παράδειγμα: ρεύμα πρώτων αριθμών

```
>>> def primes():
        numbers = integers(2)
        while True:
            first = next(numbers)
            yield first
            numbers = remove_multiples(first, numbers)
>>> prime_stream = primes()
>>> next(prime_stream)
2
>>> next(prime_stream)
3
>>> next(prime_stream)
5
```

Υλοποίηση

Υλοποίηση ρευμάτων

- Generator: αντικείμενο που ανταποκρίνεται στα μήνυματα `__next__` και `__iter__`

```
>>> class Stream:
        def __next__(self):
            pass
        def __iter__(self):
            return self
```

Υλοποίηση ρευμάτων

- Ρεύμα ακέραιων αριθμών:

```
>>> class integers(Stream):
        def __init__(self, start = 0):
            self.x = start
        def __next__(self):
            value, self.x = self.x, self.x + 1
            return value

>>> integer_stream = integers(1)
>>> integer_stream.__next__()
1
>>> next(integer_stream)
2
>>> next(integer_stream)
3
```

Υλοποίηση ρευμάτων

- Επεξεργασία ρευμάτων (φιλτράρισμα):

```
>>> class filter(Stream):
        def __init__(self, func, stream):
            self.func = func
            self.stream = stream
        def __next__(self):
            while True:
                value = next(self.stream)
                if self.func(value):
                    return value
>>> even_stream = filter(lambda x: x % 2 == 0, integers(1))
>>> next(even_stream)
2
>>> next(even_stream)
4
```

Υλοποίηση ρευμάτων

- Ρεύμα πρώτων αριθμών:

```
>>> class primes(Stream):
        def __init__(self):
            self.numbers = integers(2)
        def __next__(self):
            value = next(self.numbers)
            self.numbers = filter(lambda x:x%value!=0, \
                                  self.numbers)
            return value

>>> prime_stream = primes()
>>> next(prime_stream)
2
>>> next(prime_stream)
3
>>> next(prime_stream)
5
```

Υλοποίηση ρευμάτων

- Εμφάνιση πρώτων αριθμών εως 50:

```
>>> for prime in primes():
        if prime > 50:
            break
        print(prime)
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
```

Αρχεία

Αρχεία

- **Αρχείο (*file*):** ακολουθία που έχει αποθηκευτεί στο *σύστημα αρχείων* ενός *λειτουργικού συστήματος*
  - *Mutable* δεδομένο
  - Έχει όνομα
  - Συνήθως έχει πεπερασμένο μέγεθος
  - Μπορεί να αναπαριστά συσκευή εισόδου/εξόδου ή μνήμη
  - *Persistent* (επίμονο) δεδομένο: διατηρείται μετά τον τερματισμό του προγράμματος ή προϋπάρχει της έναρξης

- **Σύστημα αρχείων (*file system*):** σύνολο αρχείων και τρόπος διαχείρισης και οργάνωσης τους σε καταλόγους και αποθηκευτικά μέσα

- Η διαχείριση, δημιουργία, αποθήκευση και ανάγνωση αρχείων γίνεται μέσω *ρευμάτων*:
  - Προσπέλαση επόμενων δεδομένων: `read(n)` (αντί με `__next__()` όπως στους *generators*)

Αρχεία

- Κατασκευή ρεύματος για την *ανάγνωση* χαρακτήρων από αρχείο

```
>>> stream = open('shakespeare.txt')
>>> stream
<_io.TextIOWrapper name='shakespeare.txt' mode='r' encoding='UTF-8'>
>>> stream.read(1)
'A'
>>> stream.read(10)
' MIDSUMMER'
>>> rest_of_text = stream.read()
>>> len(rest_of_text)
4538512
>>> stream.close()
```

Αρχεία

- Κατασκευή ρεύματος για *εγγραφή* χαρακτήρων σε αρχείο:

```
>>> stream = open('famous_lines.txt', 'w')
>>> stream
<_io.TextIOWrapper name='famous_lines.txt' mode='w' encoding='UTF-8'>
>>> stream.write('To be')
5
>>> stream.write(' or not to be.')
>>> stream.close()
```

- Mode ανοίγματος αρχείου (2η παράμετρος της `open`)
- `'w'`: εγγραφή. (Δημιουργία αρχείου αν δεν υπάρχει, αν υπάρχει διαγράφονται τα παλιά δεδομένα και η εγγραφή αρχίζει από την αρχή.)
- `'a'`: προσθήκη (append). (Δημιουργία αρχείου αν δεν υπάρχει – αν υπάρχει η εγγραφή αρχίζει από το τέλος των παλιών δεδομένων.)
- `'w+'`: εγγραφή και ανάγνωση
- `'r+'`: προσθήκη και ανάγνωση

Αρχεία

- Η εγγραφή στο *μέσο αποθήκευσης* γίνεται ετεροχρονισμένα (άλλη μια μορφή *lazy evaluation*)

```
>>> f = open('test.out', 'w')
>>> f.write('hello')
>>> g = open('test.out')
>>> g.read()
''
>>> f.close()
>>> g.read()
'hello'
```

- Η κλήση της μεθόδου `close` ολοκληρώνει τυχόν εκκρεμείς εγγραφές

Αρχεία

- Η κλήση της μεθόδου `close` μπορεί να αποτραπεί από κάποια εξαίρεση

```
>>> try:
        f = open('test.out', 'w')
        f.write('hello')
        1 / 0
        f.close()
    except Exception:
        print('Something bad happened...')
5
Something bad happened!
>>> g = open('test.out')
>>> g.read()
''
```

Αρχεία

- Επικεφαλίδα `finally`: εκτελούνται οι εντολές στο μπλόκ της όταν τελειώσει το μπλοκ `try` ή κάποιο `except` (αν υπάρξει χειρισμός εξαίρεσης)
  - Ακόμα και αν περιλαμβάνουν εντολή `return`, `break` ή `continue`

```
>>> try:
        f = open('test.out', 'w')
        f.write('hello')
        1 / 0
    except Exception:
        print('Something bad happened...')
    finally:
        f.close()
```

Αρχεία

- Εντολή `with`:

```
>>> with open('test.txt', 'w') as f:
        f.write('hello')
        1 / 0
5
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
ZeroDivisionError: division by zero
>>> g = open('test.txt')
>>> g.read()
'hello'
```

- Εκτελείται `f.close()` αυτόματα μετά το τέλος του μπλοκ είτε συμβεί εξαίρεση είτε όχι, όπως με τα μπλοκ `try...finally`
  - Ο χειρισμός τυχούσας εξαίρεσης γίνεται μετά το `f.close()`

Αρχεία

- Δημιουργία ρεύματος για εγγραφή δεδομένων *bytes* σε αρχείο:

```
>>> with open('data', 'wb') as stream
        stream.write(b'hello')
        stream.write((2019).to_bytes(2, 'big'))
```

- Ανάγνωση:

```
>>> with open('data', 'rb') as stream
        s = stream.read(5).decode()
        x = int.from_bytes(stream.read(2), 'big')
>>> s
'hello'
>>> x
2019
```

Συσκευές εισόδου/εξόδου

Συσκευές εισόδου/εξόδου

- *Συσκευές εισόδου*: πληκτρολόγιο, ποντίκι, μικρόφωνο, αισθητήρες κλπ.
- *Συσκευές εξόδου*: οθόνη, τερματικό, ηχεία, κλπ.

- Οι συσκευές εισόδου/εξόδου αναπαρίστανται από το λειτουργικό σύστημα ως αρχεία
  - Τα προγράμματα επικοινωνούν με τις συσκευές χρησιμοποιώντας *ρεύματα*

Συσκευές εισόδου/εξόδου

- **Standard συσκευές εισόδου/εξόδου:** αρχικοποιημένα *ρεύματα* (χαρακτήρων) για βασική επικοινωνία προγραμμάτων με προκαθορισμένες συσκευές εισόδου εξόδου
  - *standard είσοδος* (`stdin`): ρεύμα εισόδου δεδομένων από χρήστη
  - *standard έξοδος* (`stdout`): ρεύμα εξόδου προγράμματος
  - *standard έξοδος λαθών* (`stderr`): ρεύμα εξόδου για εμφάνιση μηνυμάτων λάθους

```
>>> import sys
>>> sys.stdin
<_io.TextIOWrapper name='<stdin>' mode='r' encoding='UTF-8'>
>>> sys.stdout
<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>
>>> sys.stderr
<_io.TextIOWrapper name='<stderr>' mode='w' encoding='UTF-8'>
```

- Η χρήση των ρευμάτων αυτών ή των προκαθορισμένων συσκευών είναι προαιρετική. Μπορεί να υπάρξει επικοινωνία και με άλλες συσκευές

Συσκευές εισόδου/εξόδου

- Standard έξοδος

```
>>> from sys import stdout
>>> stdout.write('hello\n')
hello
6
```

- Standard είσοδος

```
>>> from sys import stdin
>>> user_data = stdin.read(5)
hello world
>>> user_data
'hello'
>>> user_data = stdin.readline()
To be or not to be
>>> user_data
'To be or not to be\n'
```

Συσκευές εισόδου/εξόδου

- Standard έξοδος λαθών

```
>>> import sys
>>> sys.stderr = open('errors.log','a')
>>> print('This goes here')
This goes here
>>> 1/0
>>> print('Error!!!', file = sys.stderr)
>>> sys.stderr.flush()
```

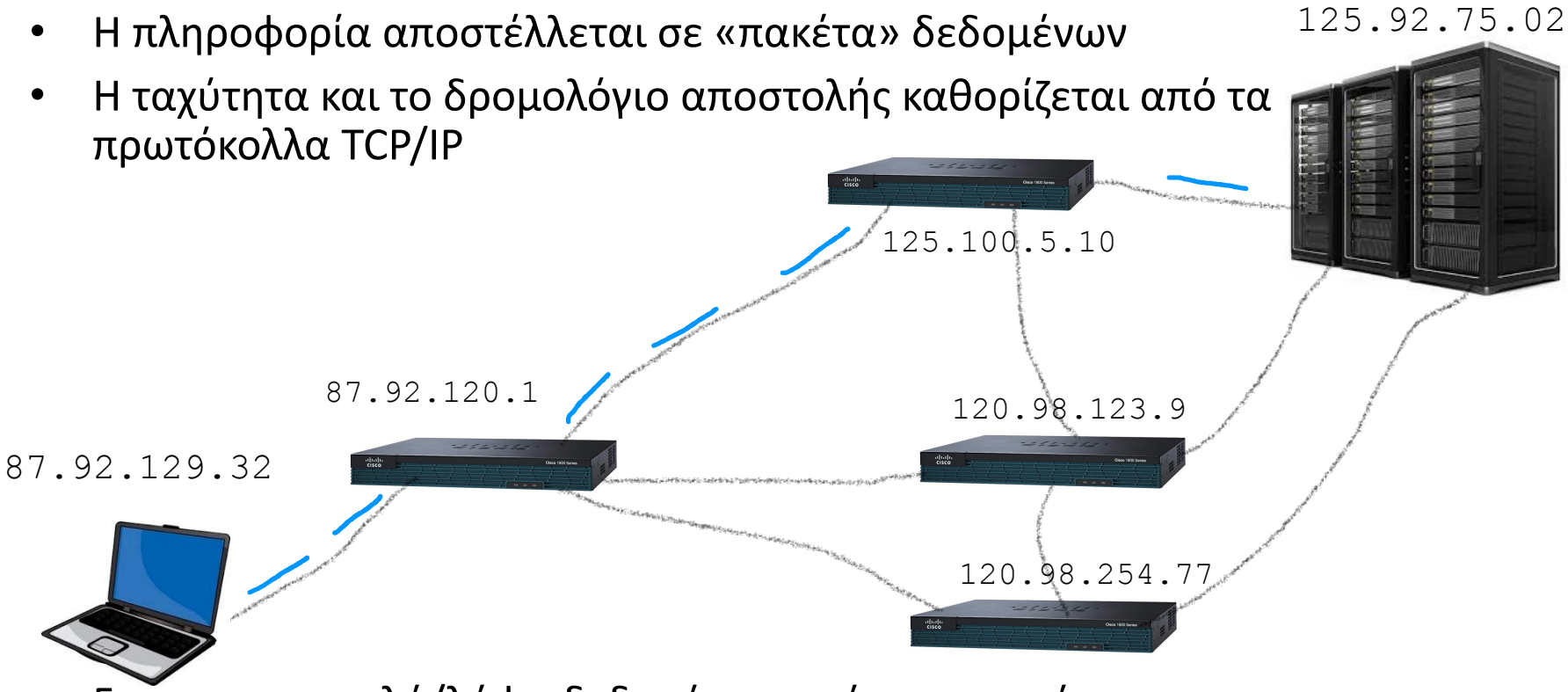
Αρχείο `errors.log`

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
Error!!!
```

Επικοινωνία μεταξύ υπολογιστών

Επικοινωνία μεταξύ υπολογιστών

- Οι περισσότεροι υπολογιστές διασυνδέονται στο διαδίκτυο (internet)
- Κάθε υπολογιστής ή δρομολογητής (router) έχει μια διεύθυνση IP
- Η πληροφορία αποστέλλεται σε «πακέτα» δεδομένων
- Η ταχύτητα και το δρομολόγιο αποστολής καθορίζεται από τα πρωτόκολλα TCP/IP

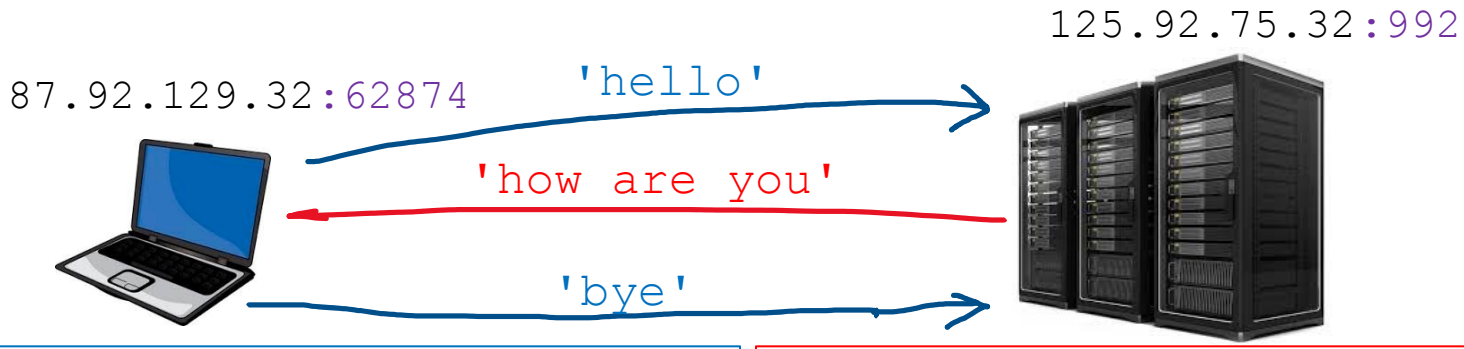


- Για την αποστολή/λήψη δεδομένων από τα προγράμματα στους υπολογιστές στις άκρες του δικτύου χρησιμοποιούνται ρεύματα

Επικοινωνία μεταξύ υπολογιστών

- **socket:** σύνθετο δεδομένο για την αποστολή και λήψη δεδομένων
  - Είναι ρεύμα εισόδου *και* εξόδου
    - Είσοδος = λήψη δεδομένων (`recv(n)` αντί `read(n)`)
    - Έξοδος = αποστολή (`send(x)` αντί `write(x)`)
  - Διαθέτει επιπλέον λειτουργίες εγκατάστασης επικοινωνίας:
    - Αντιστοίχιση σε τοπική διεύθυνση IP (`bind`)
    - Σύνδεση στην απομακρυσμένη διεύθυνση IP (`connect`)
    - Αναμονή για αιτήσεις σύνδεσης (`listen`, `accept`)
    - Τερματισμός σύνδεσης (`close`)

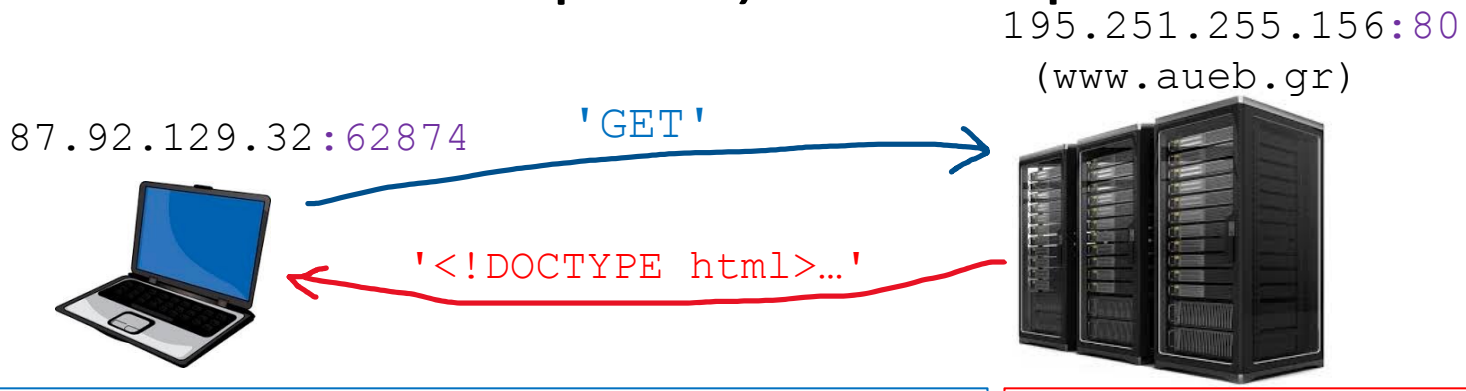
Επικοινωνία μεταξύ υπολογιστών




```
>>> from socket import *
>>> r = socket()
>>> r.connect(('125.92.75.32', 992))
>>> r.send(b'hello')
5
>>> r.recv(5).decode()
'how a'
>>> r.recv(10).decode()
're you?'
>>> r.send(b'bye')
3
>>> r.close()
```

```
>>> from socket import *
>>> s = socket()
>>> s.bind(('125.92.75.32', 992))
>>> s.listen()
>>> res = s.accept()
>>> res[0].recv(5).decode()
'hello'
>>> res[0].send(b'how are you?')
12
>>> res[0].recv(5).decode()
'bye'
>>> res[0].close()
```

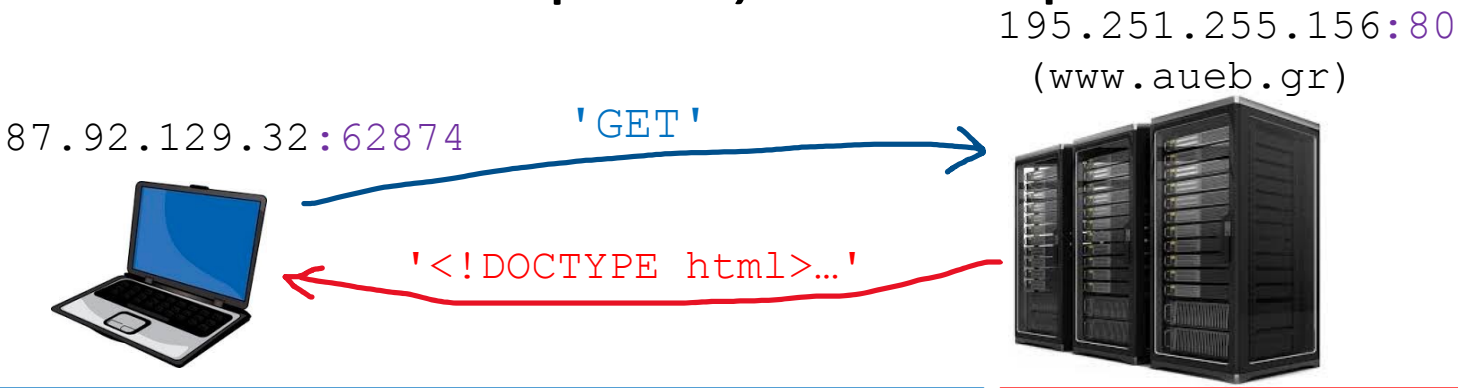
Επικοινωνία μεταξύ υπολογιστών




```
>>> from urllib.request import urlopen
>>> response = urlopen('https://www.aueb.gr/')
>>> response.readline()
b'<!DOCTYPE html>\n'
>>> response.readline().decode()
'<html lang="el" dir="ltr">\n'
>>> response.readline().decode()
'<head>\n'
>>> response.readline().decode()
'<meta name="description" content="Οικονομικό
Πανεπιστήμιο Αθηνών | Athens University of
Economics and Business">\n'
```

πρόγραμμα webserver

Επικοινωνία μεταξύ υπολογιστών

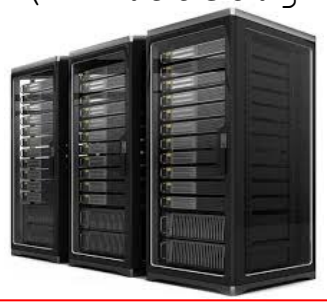


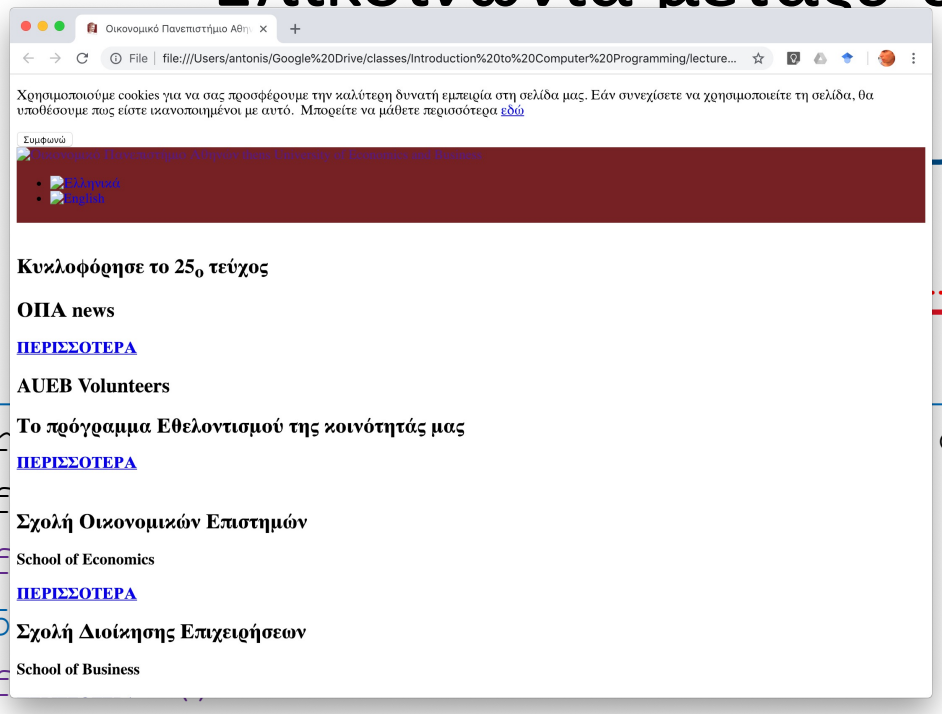

```
>>> response = urlopen('https://www.aueb.gr/')
>>> f = open('result.html','w')
>>> f.write(response.read().decode())
508354
>>> f.close()
```

πρόγραμμα webserver

Επικοινωνία μεταξύ υπολογιστών

195.251.255.156:80
(www.aueb.gr)






```
>>> r                                    gr/')
>>> f
>>> f
50835
>>> f
>>> from os import system
>>> system('/Applications/Google\
Chrome.app/Contents/MacOS/Google\ Chrome
result.html')
Opening in existing browser session.
0
```

πρόγραμμα webserver