Τμήμα Ηλεκτρολόγων Μηχανικών
& Μηχανικών Υπολογιστών
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ
ΗΜΜΥ
Πάτρα

ΤΕΧΝΙΚΕΣ ΠΡΟΓΡΑΜΜΑΤΙΜΟΥ ΥΠΟΛΟΓΙΣΤΩΝ

Διάλεξη VI

JavaScript από πιο Κοντά: Πηγαίνοντας στο Web & Μια Πρώτη Γνωριμία με την Python

Γιάννης Τζήμας, Καθηγητής

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ
UNIVERSITY of the PELOPONNESE

Τι θα δούμε σήμερα...

- Σε αυτό το μάθημα θα :
  - Κάνουμε ένα ακόμα βήμα δημιουργώντας μία εφαρμογή με JavaScript για τον Παγκόσμιο Ιστό και,
  - Θα κάνουμε μία πρώτη γνωριμία με τη γλώσσα Python.

Ας υπολογίσουμε για αρχή τη βαθμολογία μας με κλίμακες…

- Έστω ότι ισχύουν οι ακόλουθες κλίμακες:
- Και ας ξεκινήσουμε γράφοντας το test.

Κλίμακες Βαθμολογιών	
90 έως 100	A
80 έως <90	B
70 έως <80	C
60 έως <70	D
<60	F

gradeTests.js | grades.js | program.js

```
1  var book = require("../lib/grades").book;
2
3  exports["setUp"] = function(callback) {
4      book.reset();
5      callback();
6  }
7
8  exports["Can compute letter grade of A"] = function(test){
9      book.addGrade(100);
10     book.addGrade(90);
11
12     var result = book.getLetterGrade();
13
14     test.equal(result, 'A');
15     test.done();
16 };
```

Το επόμενο βήμα λοιπόν είναι να δημιουργηθεί η getLetterGrade()

- Μία πρώτη προσέγγιση θα ήταν η ακόλουθη:

grades.js | gradeTests.js

```
1   var gradeBook = {
2
3       _grades: [],
4
5       addGrade: function(newGrade) {
6           this._grades.push(newGrade);
7       },
8
9       getLetterGrade: function(){
10          var average = this.getAverage();
11
12          if(average < 60){
13              return "F";
14          }
15          If… και προχωράω με τις
16      },  υπόλοιπες περιπτώσεις.
17
```

Η λάθος προσέγγιση θα ήταν να κάνω το αντίστροφο:

```
If(average > 70)
{
        return “D”;
{
```

Ας επιλέξουμε όμως την ακόλουθη προσέγγιση

gradeTests.js | grades.js | program.js

```
1  var gradeBook = {
2
3      _grades: [],
4
5      addGrade: function(newGrade) {
6          this._grades.push(newGrade);
7      },
8
9      getLetterGrade: function(){
10         var average = this.getAverage();
11
12         if(average >= 90){
13             return "A";
14         }
15         else if (average >= 80){
16             return "B";
17         }
18         else if (average >= 70){
19             return "C";
20         }
21         else if (average >= 60){
22             return "E";
23         }
24         return "F";
25     },
26
```

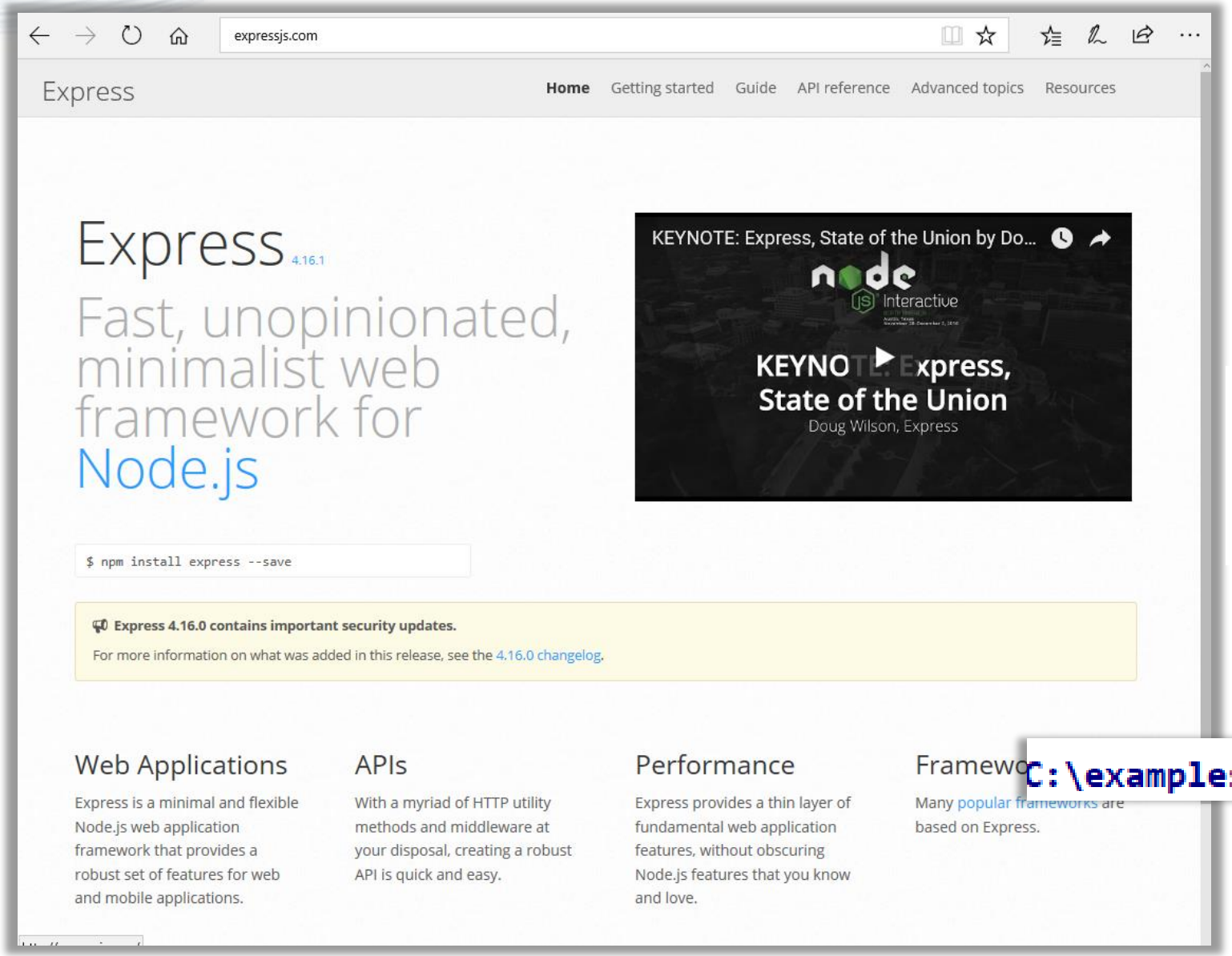
- Και πετύχαμε αυτό που θέλαμε.
- Δοκιμάστε στο σπίτι να γράψετε και τα υπόλοιπα test.

```
C:\examples\gradebook>nodeunit tests

gradeTests
√ Can compute letter grade of A
√ Can add new grade
√ Can average grades

OK: 3 assertions (8ms)
```

Πηγαίνοντας τώρα στον Παγκόσμιο Ιστό μπορούμε να εγκαταστήσουμε το express.js

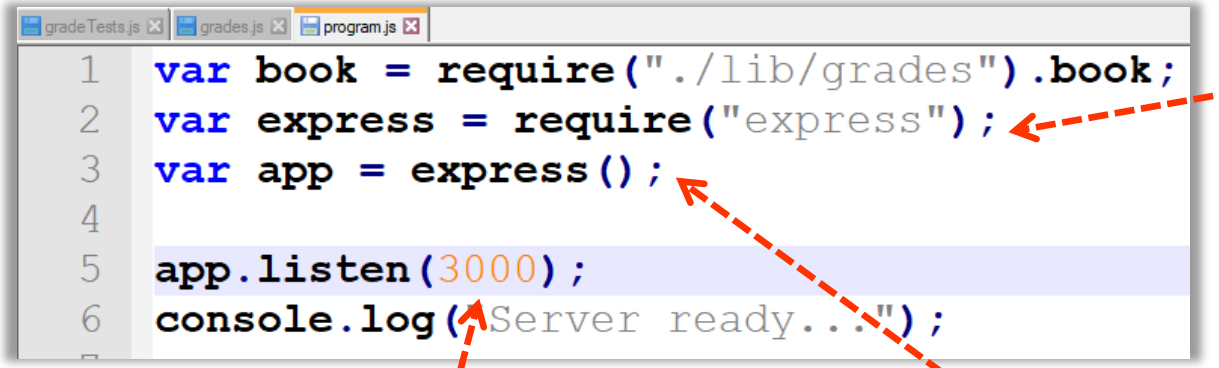


- Για να το εγκαταστήσετε γράψτε στο command prompt το ακόλουθο:
- **`npm install express`**

```
C:\examples\gradebook>npm install express
```

Θα αλλάξουμε το program.js

- …ώστε να το μετατρέψουμε σε Web Server που αλληλεπιδρά με ένα Web browser.



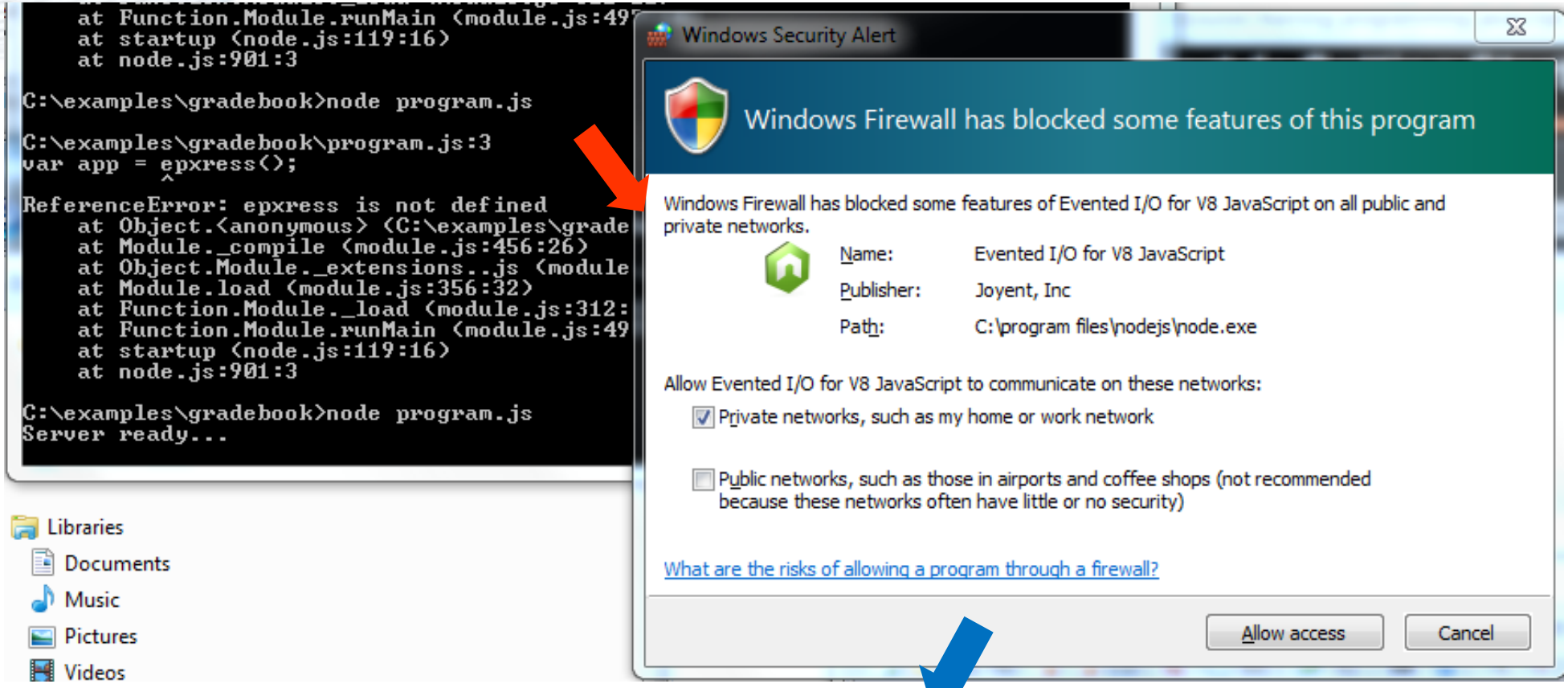
Χρησιμοποιούμε το express και καθώς είναι ένα ήδη εγκατεστημένο module αρκεί να κάνουμε αυτή τη δήλωση, το node φροντίζει για τα υπόλοιπα.

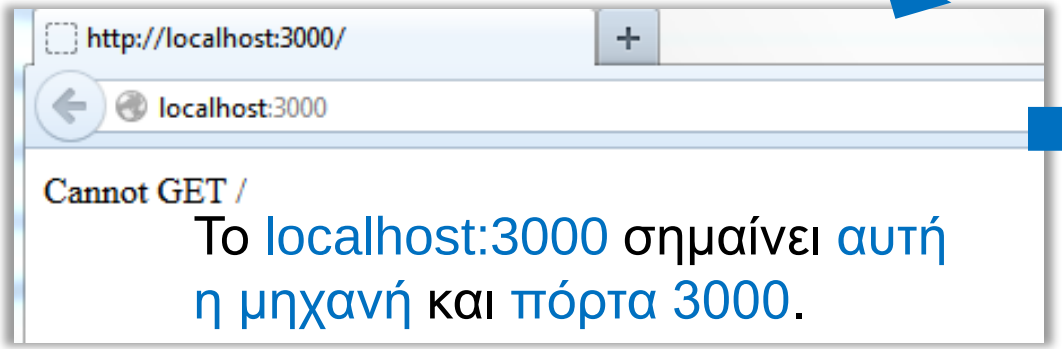
Λέμε στη εφαρμογή να «ακούει» για συνδέσεις στην πόρτα 3000.

Αυτό θα συνεχίσει να τρέχει εκτός και αν το σταματήσουμε ή επανεκκινήσει η μηχανή.

Ορίζουμε μία μεταβλητή app και αυτό που εξάγει το express είναι μία συνάρτηση την express() την οποία καλούμε και την αναθέτουμε στην app.

Αν το δοκιμάσουμε…





Το localhost:3000 σημαίνει αυτή η μηχανή και πόρτα 3000.

Ο Web Server τρέχει, αλλά δεν ξέρει τι να απαντήσει…

Λέγοντας όμως στο Web Server πώς να συμπεριφερθεί….

```
C:\examples\gradebook>node program.js
Server ready...
^C
C:\examples\gradebook>
```

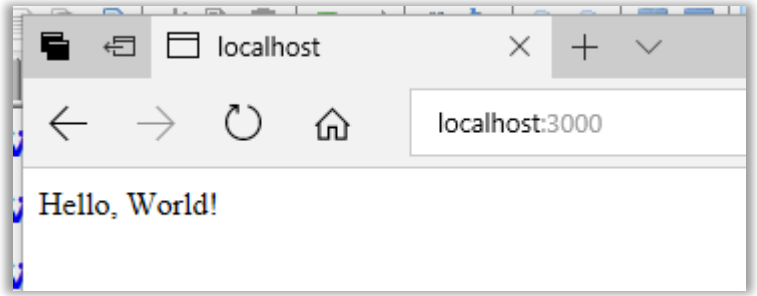
Πρώτα απ' όλα τον σταματάμε και μετά αλλάζουμε το program.js…

gradeTests.js | grades.js | program.js

```
1  var book = require("./lib/grades").book;
2  var express = require("express");
3  var app = express();
4
5  app.get("/", function(req, res){
6      res.send("Hello, World!");
7  })
8
9  app.listen(3000);
10 console.log("Server ready...");
```

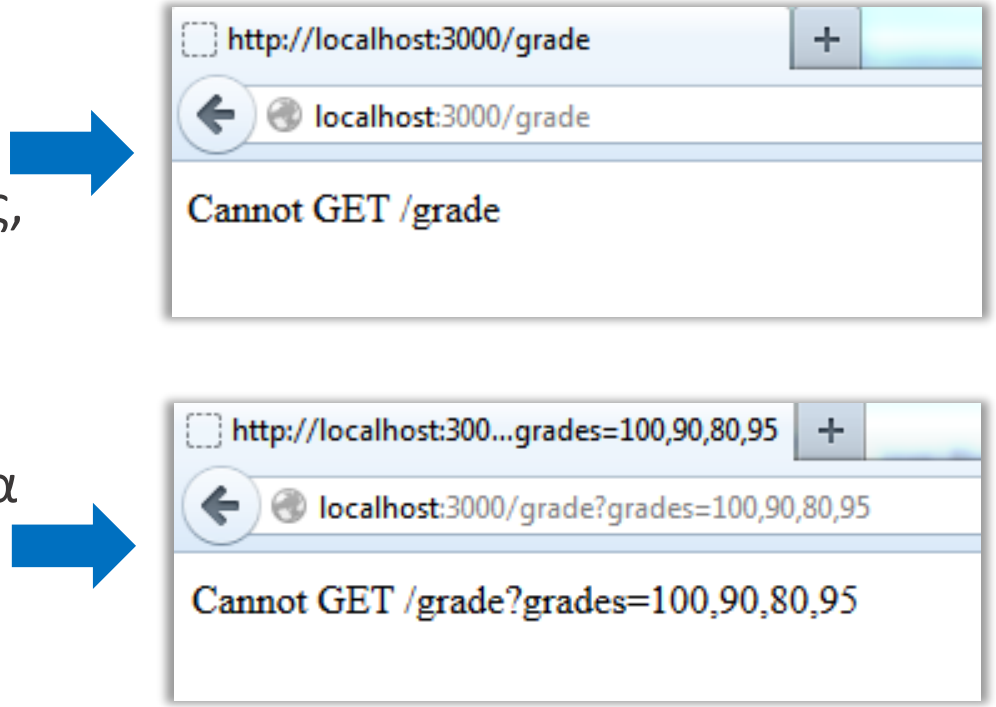
```
C:\examples\gradebook>node program.js
Server ready...
```

Είπαμε στον Web Server, αν κάποιος σου ζητήσει το root ("/"), απάντα Hello, World!



Το επόμενο βήμα

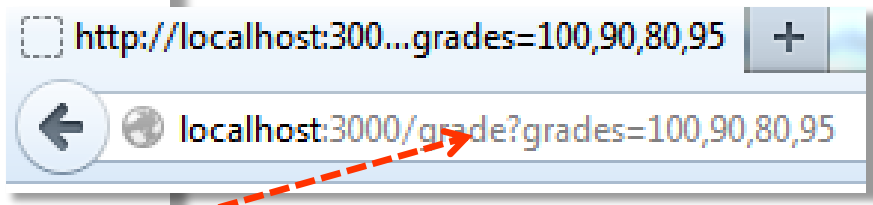
- Αφού έχουμε εγκαταστήσει τη βασική μας υποδομή θα θέλαμε:
  - να μπορούμε να έχουμε ένα διαφορετικό URL (Uniform Resource Locator) για την εφαρμογή μας,
  - και να μπορούμε να περάσουμε παραμέτρους σε αυτό το URL, ώστε το βαθμολόγιό μας να μπορεί να κάνει τη σχετική αξιολόγηση (μέση βαθμολογία και βαθμολόγιο με κλίμακες).
    Για να περάσουμε παραμέτρους χρησιμοποιούμε ένα query string, δηλαδή το χαρακτήρα «?» ακολουθούμενο από τις σχετικές παραμέτρους.



Ας το υλοποιήσουμε – Βήμα 1

gradeTests.js | grades.js | program.js

```
1  var book = require("./lib/grades").book;
2  var express = require("express");
3  var app = express();
4
5  app.get("/", function(req, res){
6      res.send("Hello, World!");
7  })
8
9  app.get("/grade", function(req, res){
10
11     // Το "1,2,3".split(",") οδηγεί στο array από strings [1,2,3]
12     var grades = req.query.grades.split(",");
13
14     Με διευκολύνει το express γιατί το αντικείμενο req έχει την
15     ιδιότητα query, η οποία μου επιστρέφει ότι περιέχει το
16     query string.
17     Και στη συνέχεια με τη split θα διαχωριστεί το string που
18
19     μου έχει επιστραφεί στα σημεία που υπάρχει "," ώστε τελικά
20 })    να έχω ένα array από strings.
21
22 app.listen(3000);
23 console.log("Server ready...");
```



- Πρώτα απ' όλα θα πρέπει να πάρω τους βαθμούς από το query string.

Βήμα 2

gradeTests.js | grades.js | program.js

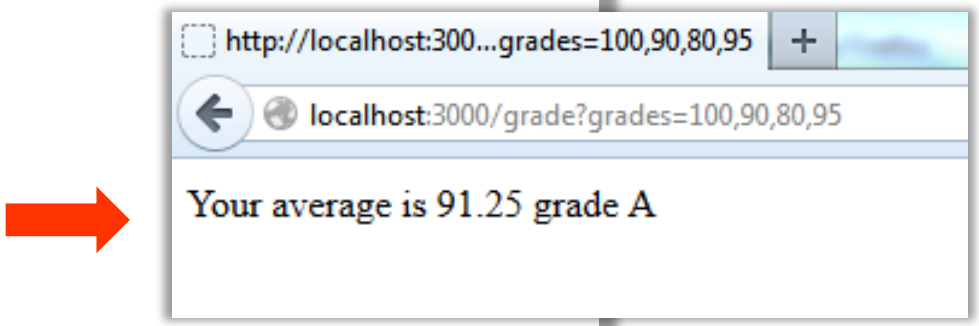
```
1  var book = require("./lib/grades").book;
2  var express = require("express");
3  var app = express();
4
5  app.get("/", function(req, res){
6      res.send("Hello, World!");
7  })
8
9  app.get("/grade", function(req, res){
10
11     // Το "1,2,3".split(",") οδηγεί στο array από strings [1,2,3]
12     var grades = req.query.grades.split(",");
13     for (var i = 0; i < grades.length; i+=1){
14         book.addGrade(parseInt(grades[i]));
15     }
16
17
18
19
20
21 })
22
23 app.listen(3000);
24 console.log("Server ready...");
```

Στη συνέχεια διαπερνάω το grades array ώστε να βάλω τους βαθμούς στο **gradeBook** με την **addGrade**.
Με την **parseInt()** μετατρέπουμε τα Strings σε Integers.

Βήμα 3

gradeTests.js | grades.js | program.js

```
1  var book = require("./lib/grades").book;
2  var express = require("express");
3  var app = express();
4
5  app.get("/", function(req, res){
6      res.send("Hello, World!");
7  })
8
9  app.get("/grade", function(req, res){
10
11     // Το "1,2,3".split(",") οδηγεί στο array από strings [1,2,3]
12     var grades = req.query.grades.split(",");
13     for (var i = 0; i < grades.length; i+=1){
14         book.addGrade(parseInt(grades[i]));
15     }
16     var average = book.getAverage();
17     var letter = book.getLetterGrade();
18
19     res.send("Your average is " + average + " grade " + letter);
20 })
21
22 app.listen(3000);
23 console.log("Server ready...");
```



- Και ολοκληρώνουμε υπολογίζοντας το μέσο όρο και τη βαθμολογία με γράμματα

Τι είδαμε μέχρι τώρα;

- Και το τέλος είναι μόνο η αρχή…
  Με τη JavaScript μπορείτε να:
  - Συνεχίσετε να προγραμματίζετε αξιοποιώντας το node και τον express.js ή να
  - Γράψετε εφαρμογές για ιστοσελίδες ή να
  - Κάνετε ερωτήματα σε Βάσεις Δεδομένων ή να
  - Φτιάξετε εφαρμογές για Windows και για το Microsoft Office και πολλά ακόμα…

- Αυτό που πρέπει συνολικά να θυμάστε είναι ότι όσο περισσότερο προγραμματίζετε, τόσα περισσότερα θα μάθετε.

Abstraction - Αφαίρεση

- Όσο προγραμματίζουμε τα προγράμματα γίνονται συνεχώς πιο πολύπλοκα.
- Ένα από τα κλειδιά του να χτίσει κάποιος λογισμικό είναι το να καταφέρει να διαχειριστεί την πολυπλοκότητά του.

- Για να διαχειριστούμε την πολυπλοκότητα αυτή χρησιμοποιούμε **αφαιρέσεις**.
  - Σπάμε μεγάλα τμήματα κώδικα σε μικρότερα τμήματα και τα μικρά αυτά τμήματα κρύβουν μέσα τους τις πολύπλοκες λεπτομέρειες δημιουργώντας έτσι μία **γενίκευση** του τι συμβαίνει.
  - Συνδυάζοντας αυτά τα μικρά τμήματα δημιουργούμε το επιθυμητό λογισμικό.
- Η διαδικασία αυτή μοιάζει με παιχνίδι από τουβλάκια Lego. Στην πορεία του μαθήματος θα δούμε πως το πετυχαίνουμε αυτό.

Python

- Στη συνέχεια θα ασχοληθούμε με μια επίσης δημοφιλή γλώσσα που τρέχει σε
  - Linux,
  - Windows και
  - Mac, την Python.

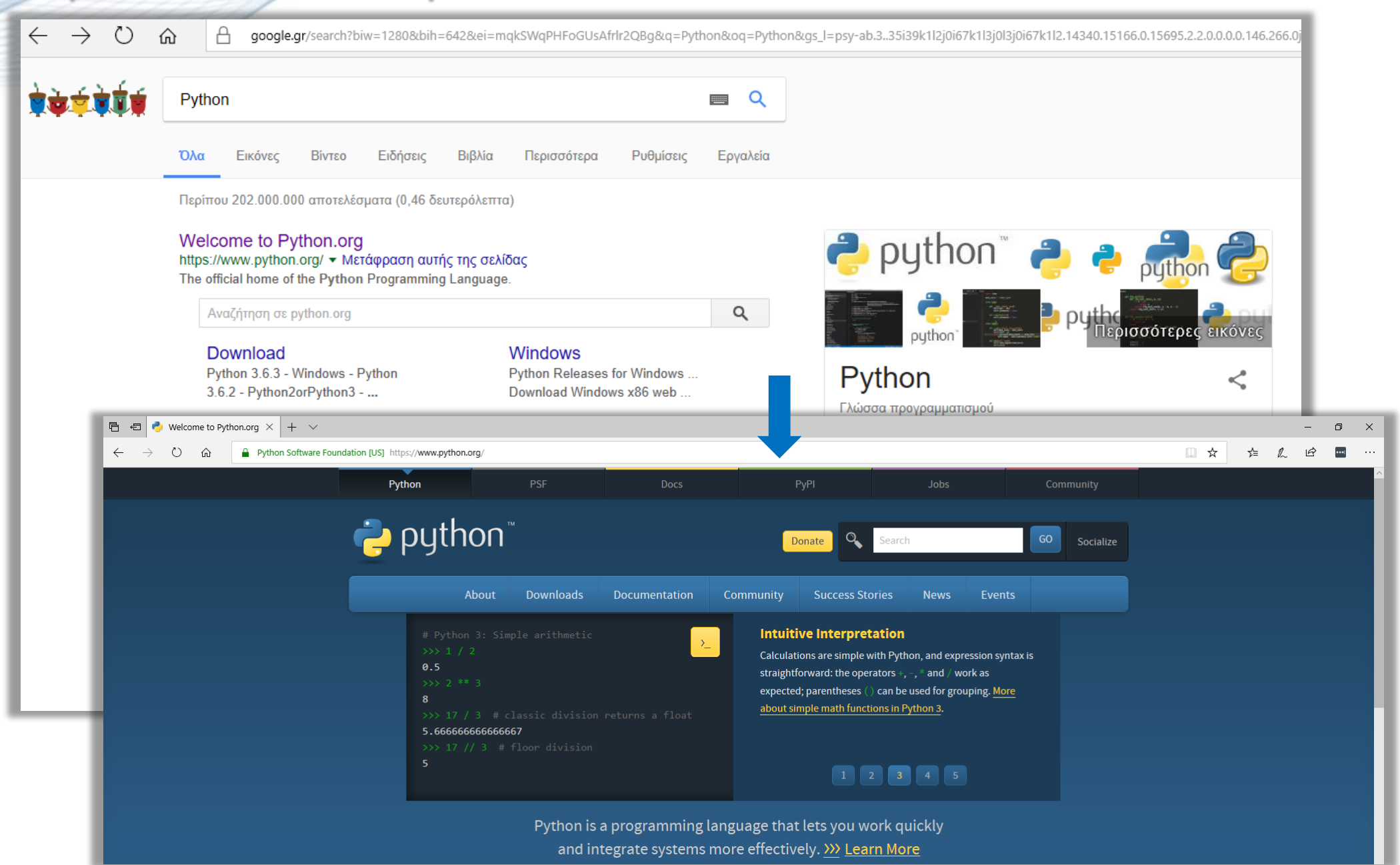
- Βλέποντας περισσότερες γλώσσες θα σας δώσει μία πιο ευρεία προοπτική σε σχέση με τον προγραμματισμό.

- Διαφορετικές γλώσσες μπορούν να σε οδηγήσουν στην επίτευξη του ίδιου σκοπού, χρησιμοποιώντας όμως διαφορετικό συντακτικό.

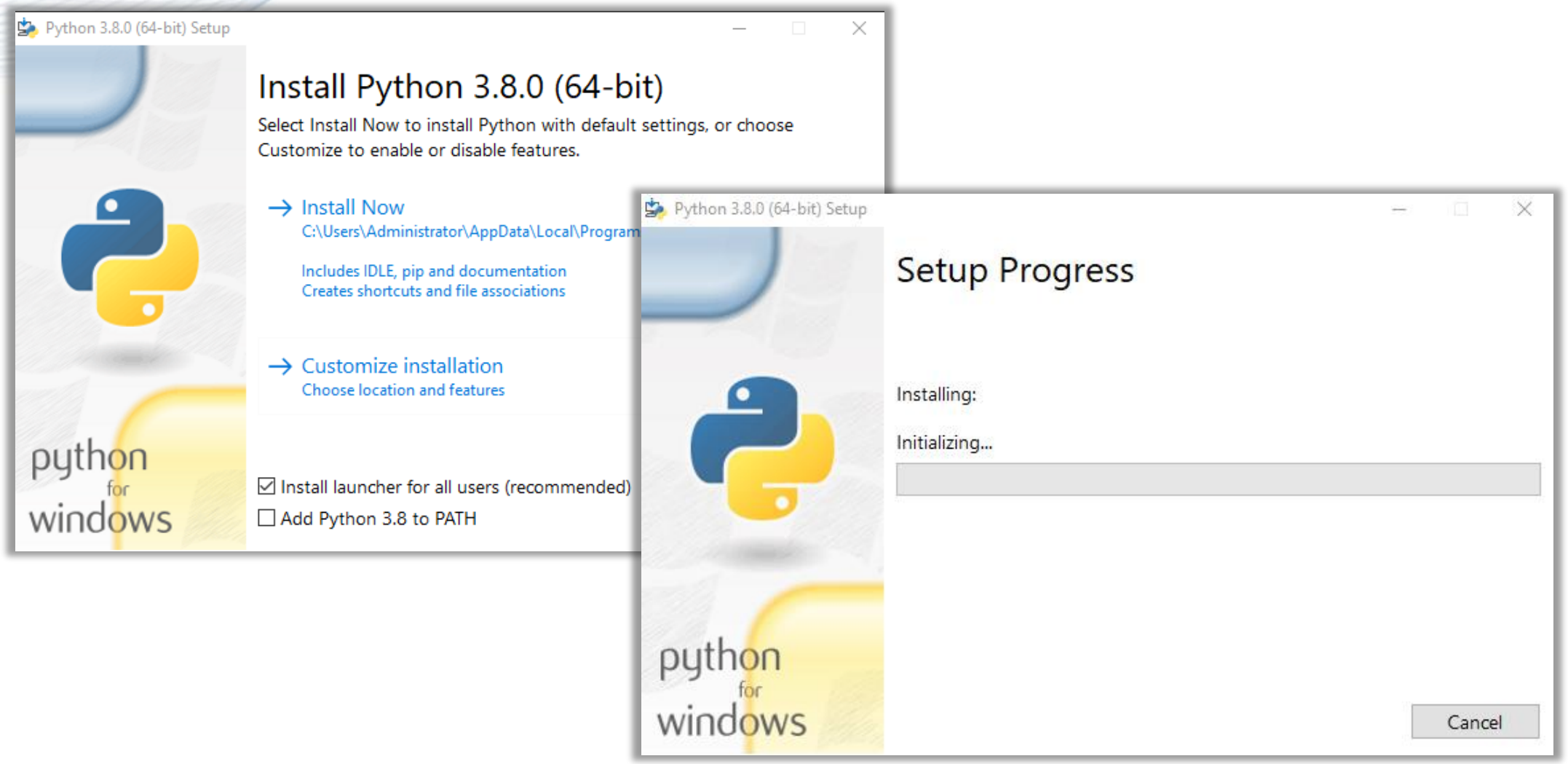
- Η Python έχει σχεδιαστεί ώστε να είναι **ευανάγνωστη** (readable).

- Ας την εγκαταστήσουμε…

Εγκατάσταση

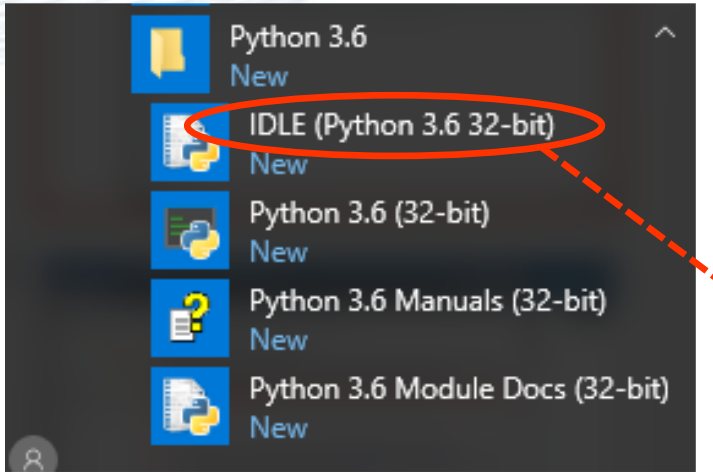


Εγκατάσταση

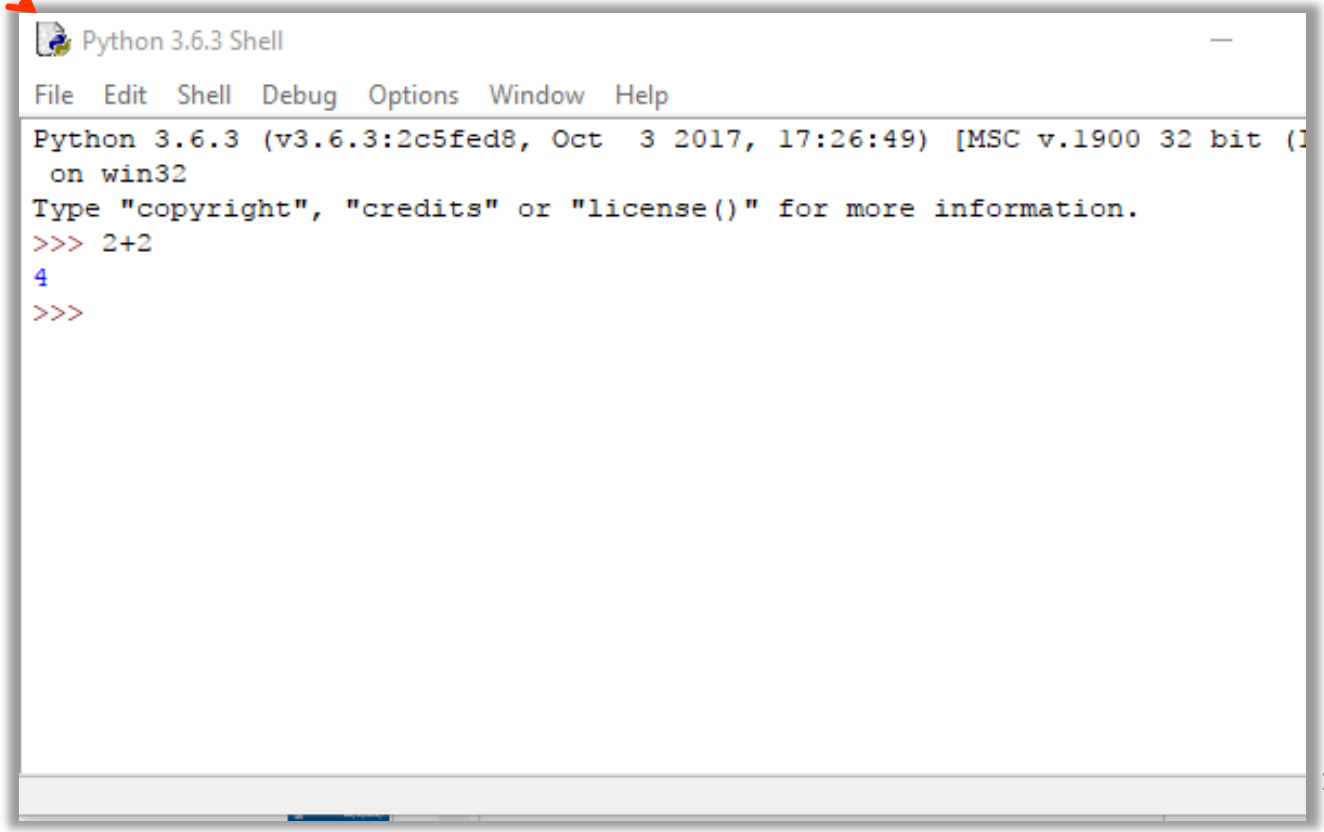


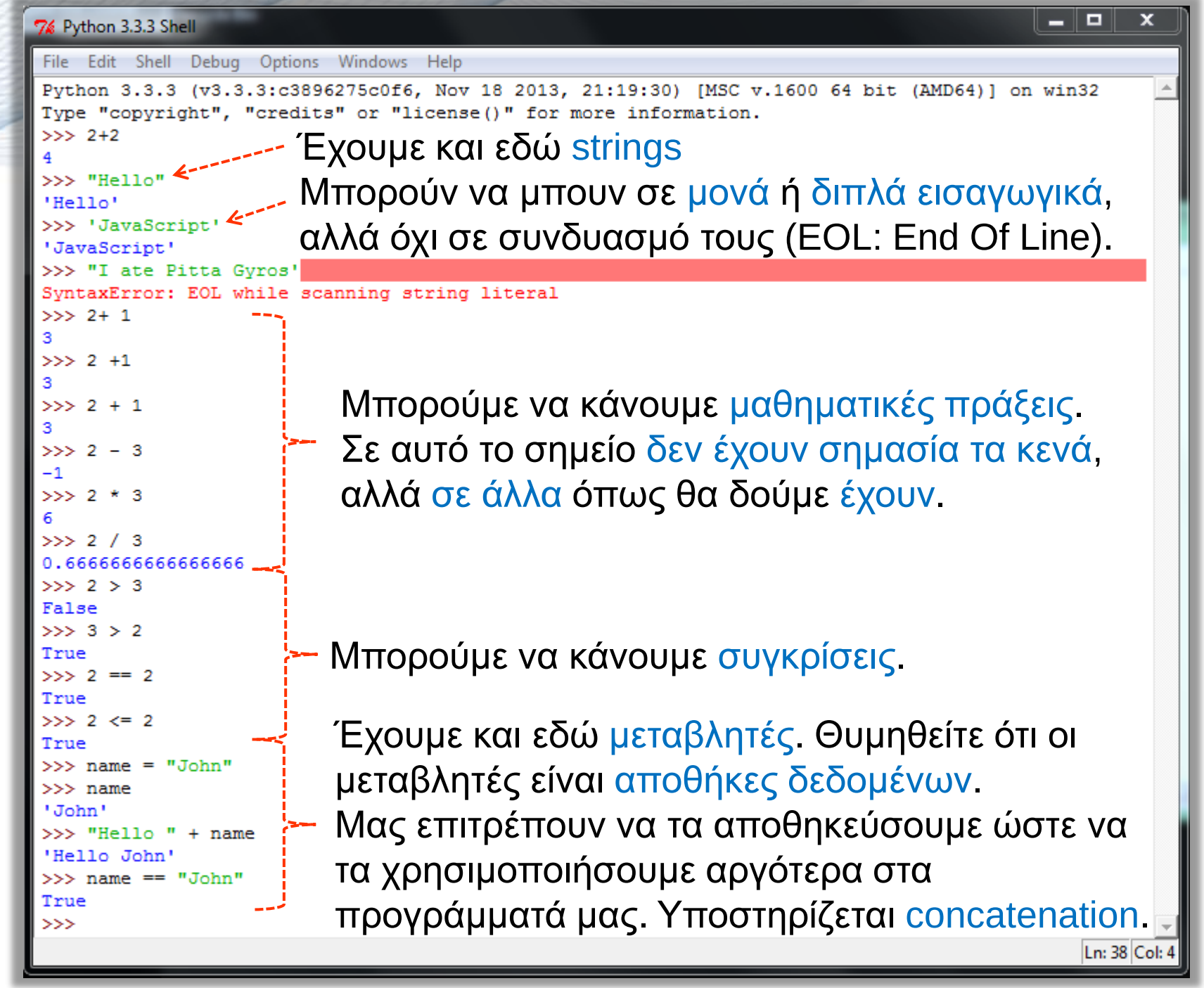
- Επιλέγετε τα προεπιλεγμένα και γίνεται η εγκατάσταση.

Για να την «τρέξουμε»…



- Είμαστε σε παρόμοιο περιβάλλον με το node REPL της JavaScript.
- Θα μπορούσαμε επίσης να την τρέξουμε σε command line όπως το node.



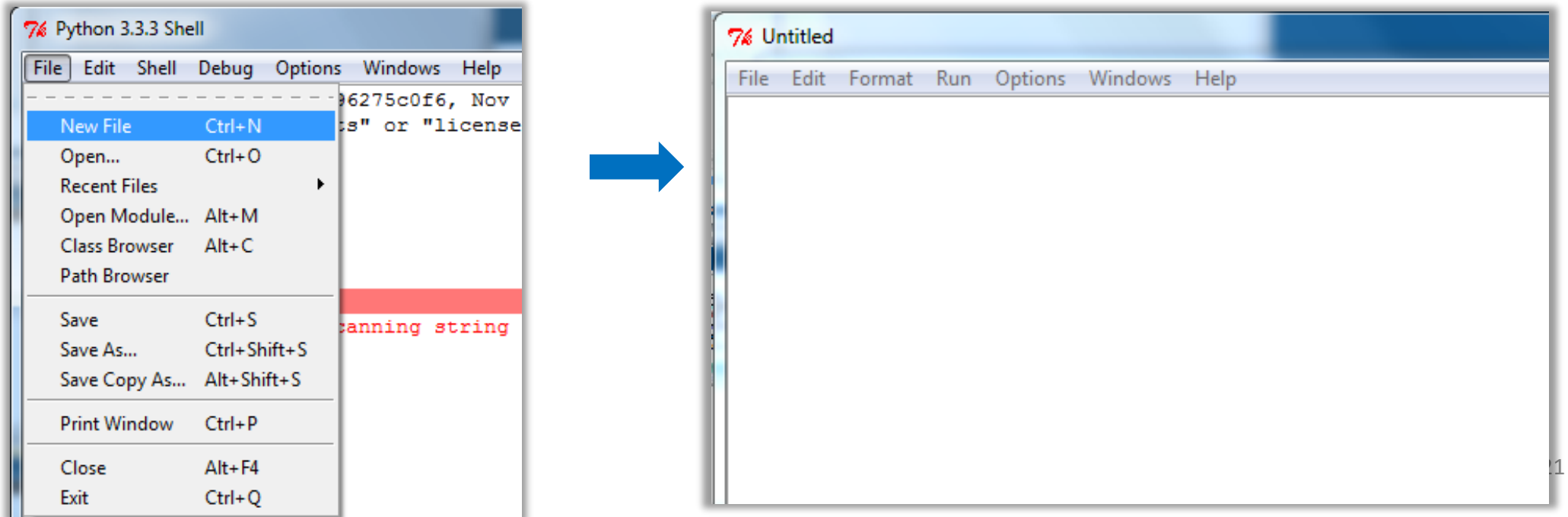


Πράγματα που ήδη ξέρουμε...

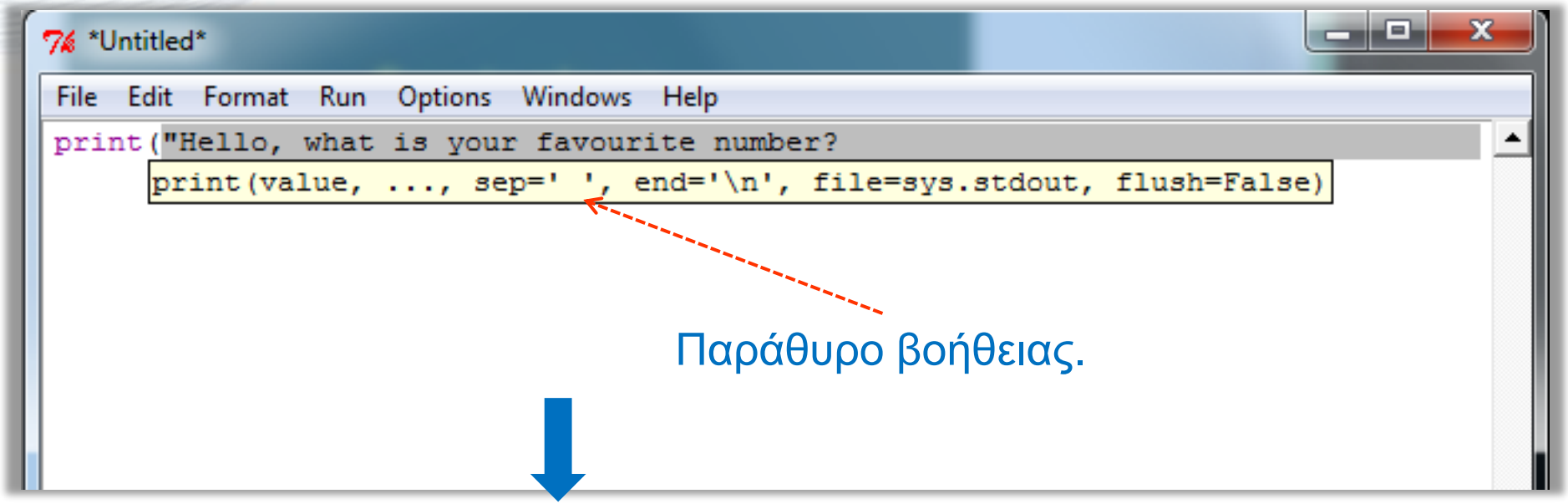
Ισχύουν όπως και στη JavaScript

Ας φτιάξουμε λοιπόν το πρώτο μας πρόγραμμα...

- Ο χρήστης θα πρέπει να μαντέψει έναν αριθμό.
- Το πρόγραμμα θα επιλέξει έναν αριθμό, ο χρήστης θα πρέπει να τον μαντέψει, και το πρόγραμμα θα απαντήσει αν αυτό που μάντεψε ο χρήστης είναι σωστό ή λάθος και θα πρέπει να ανέβει ή να κατέβει στην εκτίμησή του, αν έχει κάνει λάθος.
- Στην περίπτωση της Python δεν χρειάζεται να κατεβάσουμε editor για να γράψουμε ένα πρόγραμμα και να το σώσουμε. Απλά επιλέγουμε File -> New File



Ξεκινώντας να γράφουμε…

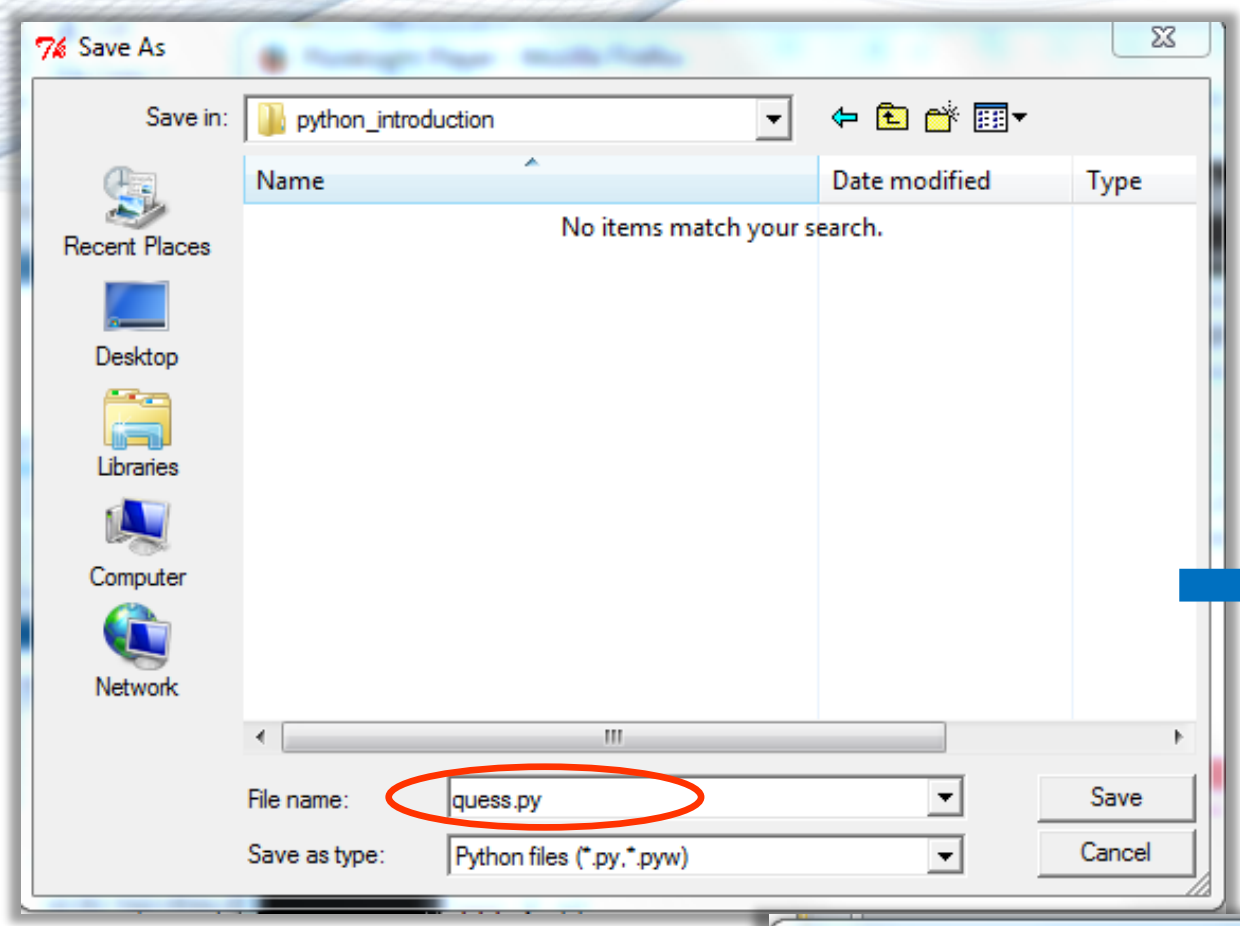


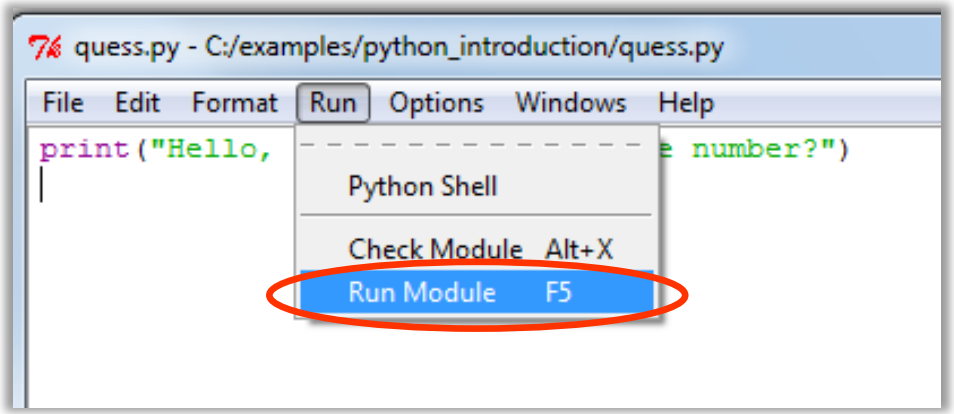

```
print("Hello, what is your favorite number?")
```

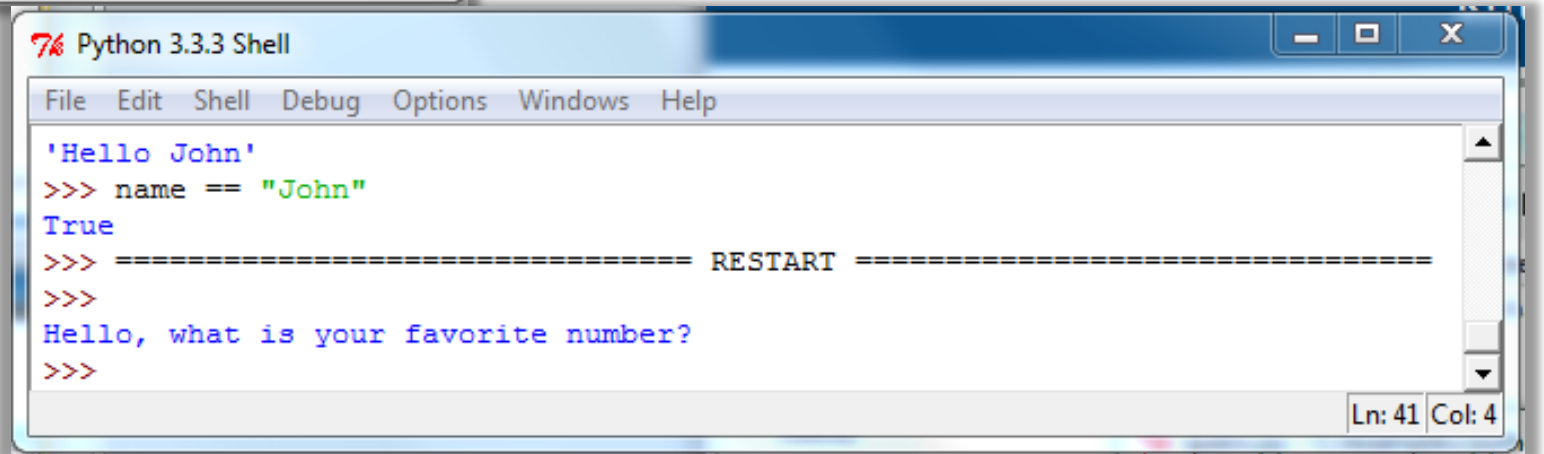
- Αντί για την **`console.log("xxx")`** της JavaScript τώρα καλούμε τη συνάρτηση **`print()`** – Έχουμε και εδώ παρενθέσεις όπως στη JavaScript.

- Παρατηρείστε ότι δεν υπάρχει ο χαρακτήρας «**;**» στο τέλος της εντολής.

Και αν το σώσουμε... μπορούμε να το τρέξουμε...

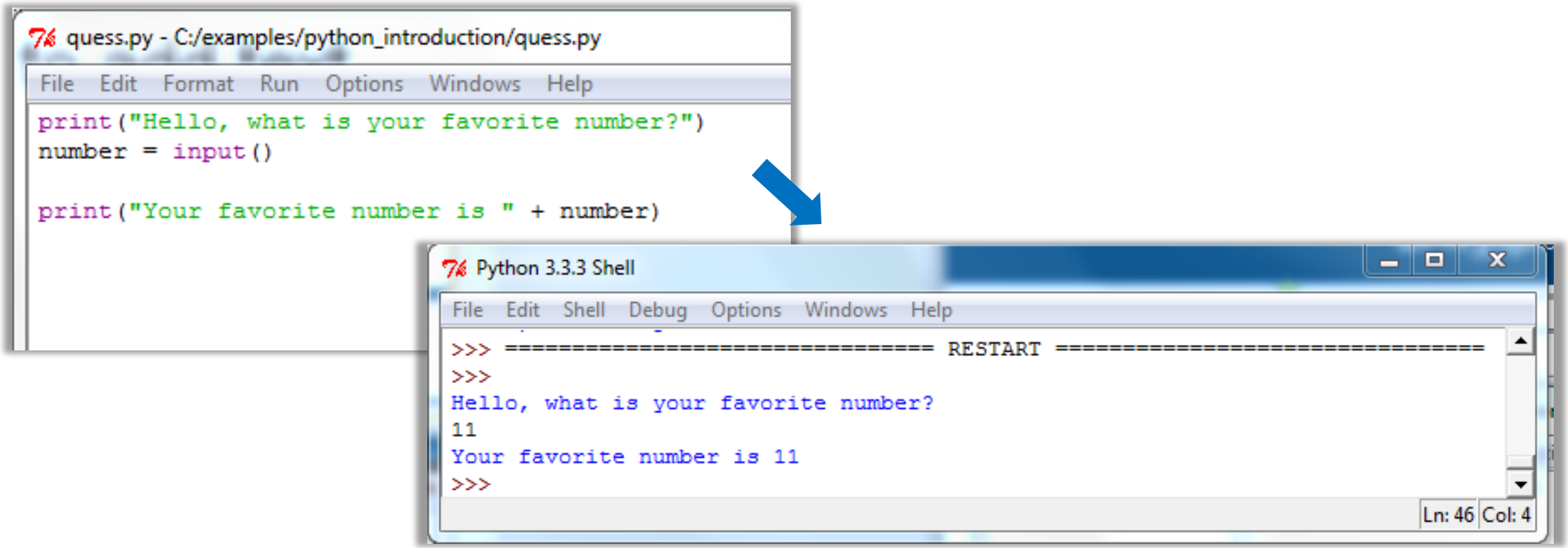






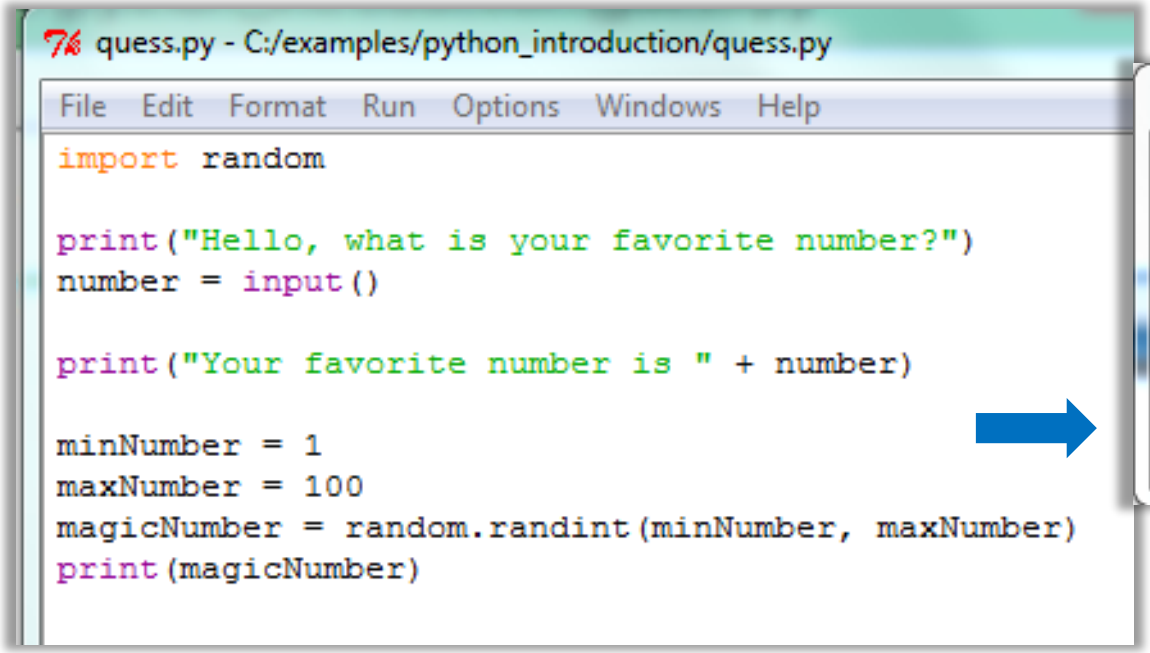
Συνεχίζουμε να γράφουμε κώδικα

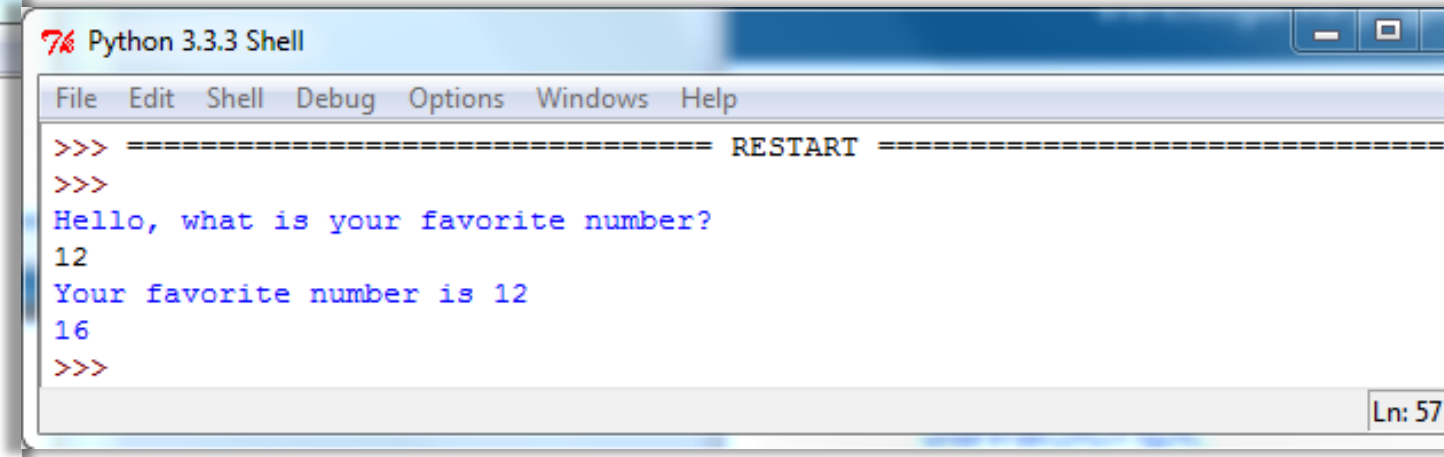
- Ορίζουμε μία μεταβλητή **`number`** και της αναθέτουμε την τιμή που επιστρέφει η συνάρτηση **`input()`**.
- Η **`input()`** περιμένει την είσοδο του χρήστη και αφού αυτός τη δώσει και πατήσει Enter την επιστρέφει στη μεταβλητή number.



Το επόμενο βήμα είναι να γεννηθεί ένας τυχαίος αριθμός

- Για να κάνουμε το ίδιο με JavaScript θυμηθείτε ότι χρησιμοποιούσαμε τη **`math.random()`**.
- Η έκφραση **`import`** είναι αντίστοιχη με την έκφραση **`require`** της JavaScript. Θυμηθείτε γράφαμε **`var app = require("express");`**
  Με τη **`require`** μπορούσαμε να χρησιμοποιήσουμε υπάρχουσες βιβλιοθήκες.





Καλό θα είναι να πούμε στο χρήστη ο τυχαίος αριθμός ανάμεσα σε ποιες τιμές βρίσκεται...

quess.py - C:/examples/python_introduction/quess.py

File Edit Format Run Options Windows Help

```
import random

print("Hello, what is your favorite number?")
number = input()

print("Your favorite number is " + number)

minNumber = 1
maxNumber = 100
magicNumber = random.randint(minNumber, maxNumber)

message = "The magic number is between {0} and {1}"
print(message.format(minNumber, maxNumber))
```

Python 3.3.3 Shell

File Edit Shell Debug Options Windows Help

```
>>> ================================ RESTART ================================
>>>
Hello, what is your favorite number?
14
Your favorite number is 14
The magic number is between 1 and 100
>>>
```

Ln: 63 Col: 4

- Με τις αγκύλες {} μπορούμε να φτιάξουμε strings στην Python αποφεύγοντας το concatenation.
- Αυτό που κάνουμε είναι ένα **template** με θέσεις τις οποίες μπορούμε να γεμίσουμε. Αυτό μοιάζει με τα arrays άλλων γλωσσών όπως η JavaScript.
- Εφαρμόζοντας παραμέτρους σε αυτό το template μπορούμε να είμαστε πιο αποτελεσματικοί. Το κάνουμε με τη μέθοδο **`format()`**.

Αν κάναμε κάποιο λάθος…

- Η Python δε θα μας άφηνε να προχωρήσουμε…

quess.py - C:/examples/python_introduction/quess.py

File Edit Format Run Options Windows Help

```
import random

print("Hello, what is your favorite number?")
number = input()

print("Your favorite number is " + number)

minNumber = 1
maxNumber = 100
magicNumber = random.randint(minNumber, maxNumber)

message = "The magic number is between {1} and {2}"
print(message.format(minNumber, maxNumber))
```

Python 3.3.3 Shell

File Edit Shell Debug Options Windows Help

```
>>> ================================ RESTART ================================
>>>
Hello, what is your favorite number?
15
Your favorite number is 15
Traceback (most recent call last):
  File "C:/examples/python_introduction/quess.py", line 13, in <module>
    print(message.format(minNumber, maxNumber))
IndexError: tuple index out of range
>>>
```

Ln: 72 Col: 4

Πηγαίνοντας στη βασική λογική του προγράμματος... Θα πρέπει να πάμε σε looping

quess.py - C:/examples/python_introduction/quess.py

File Edit Format Run Options Windows Help

```
import random

print("Hello, what is your favorite number?")
number = input()

print("Your favorite number is " + number)

minNumber = 1
maxNumber = 100
magicNumber = random.randint(minNumber, maxNumber)

message = "The magic number is between {0} and {1}"
print(message.format(minNumber, maxNumber))


found = False

while not found:
    print("Guess what it is?")
    guess = int(input())
    if guess == magicNumber:
        found = True
    if guess < magicNumber:
        print("Too low")
    if guess > magicNumber:
        print("Too high")
print("You got it")
```

- Για να πετύχουμε το στόχο μας φτιάχνουμε ένα βασικό loop.

Για να δούμε τις λεπτομέρειες

Το while θα κάνει loop μέχρι η έκφραση να πάρει τιμή false.

not true == false
not false == true

```
found = False

while not found:
    print("Guess what it is?")
    guess = int(input())
    if guess == magicNumber:
        found = True
    if guess < magicNumber:
        print("Too low")
    if guess > magicNumber:
        print("Too high")
print("You got it")
```

Οτιδήποτε είναι με εσοχή κάτω από ένα loop, ανήκει σε αυτό!

Εκτός του while loop

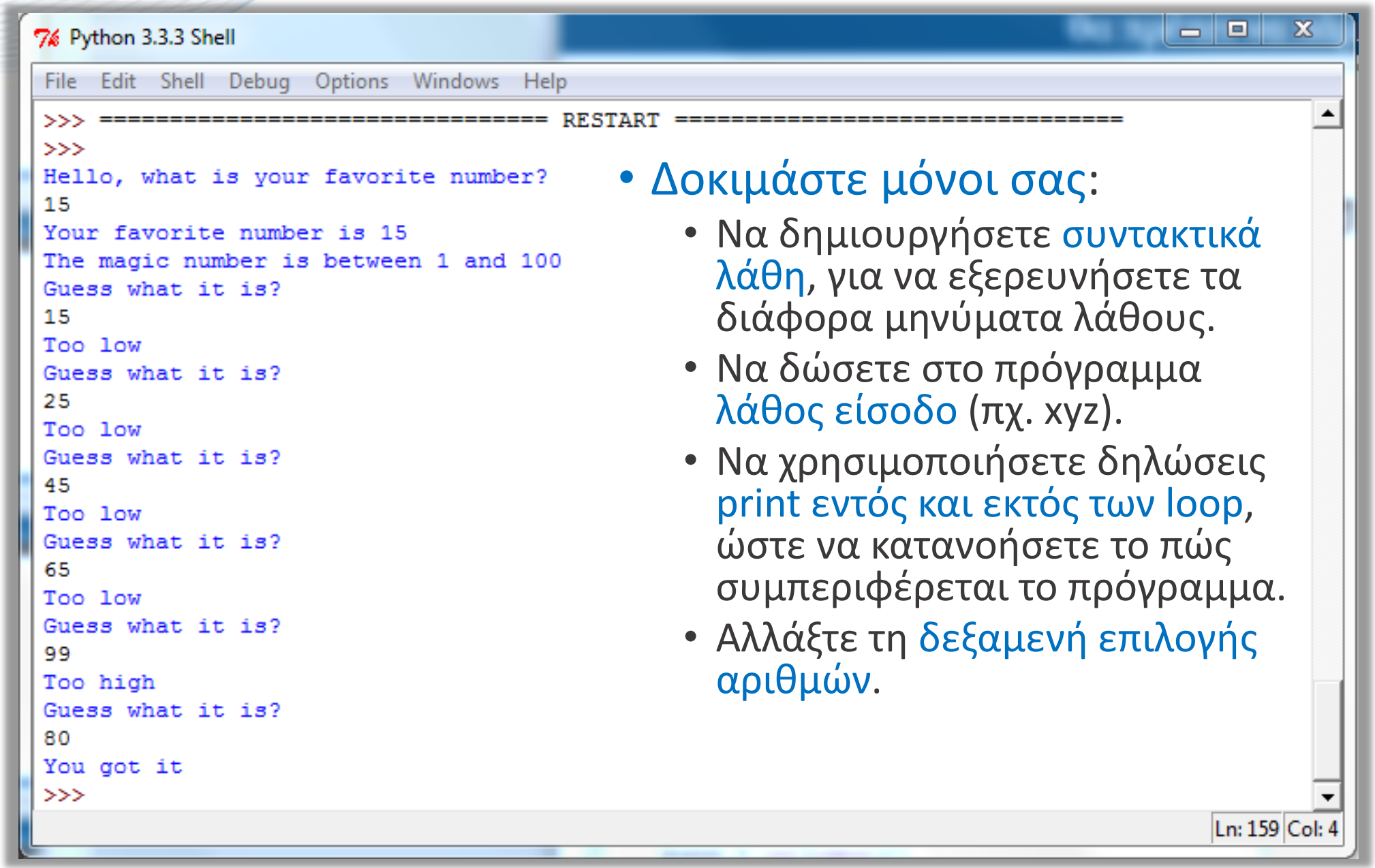
Προσοχή: Η είσοδος που παίρνουμε από το χρήστη είναι τύπου string.

Στη JavaScript θα γράφαμε:

```
If(true) {
    doSomething();
    doSomethingElse();
}
```

Στην Python στο τέλος της δήλωσης βάζουμε το χαρακτήρα «:».
Οι εσοχές παίζουν πολύ μεγάλο ρόλο. Βάζοντας εσοχή μπαίνουμε εντός του loop.

Και αν τρέξουμε το πρόγραμμά μας…



- Δοκιμάστε μόνοι σας:
  - Να δημιουργήσετε συντακτικά λάθη, για να εξερευνήσετε τα διάφορα μηνύματα λάθους.
  - Να δώσετε στο πρόγραμμα λάθος είσοδο (πχ. xyz).
  - Να χρησιμοποιήσετε δηλώσεις print εντός και εκτός των loop, ώστε να κατανοήσετε το πώς συμπεριφέρεται το πρόγραμμα.
  - Αλλάξτε τη δεξαμενή επιλογής αριθμών.

Τι είδαμε μέχρι τώρα;

- Φτιάξαμε ένα πρώτο πρόγραμμα σε Python και είδαμε ότι δεν είναι τόσο διαφορετική από τη JavaScript. Πολλές γλώσσες έχουν πολλά κοινά σημεία, αλλά η κάθε μία χρησιμοποιεί δική της σύνταξη.

- Συνήθως διαλέγουμε μία γλώσσα προγραμματισμού με βάση την προτίμησή μας ή το τι θέλουμε να πετύχουμε.

- Κάθε γλώσσα έχει τα δικά της θετικά και αρνητικά σημεία.

- Στη συνέχεια θα επικεντρωθούμε στην Python και θα ασχοληθούμε με τη δημιουργία και χρήση αφαιρέσεων.

JavaScript

```
var x = 3;
```

```
console.log("Hi!");
```

```
if(guess == number) {
   found = true;
}
```

Python

```
x = 3
```

```
print("Hi!")
```

```
if guess == number:
    found = True
```


python
JavaScript