ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Εισαγωγή στους Υπολογιστές

Εργαστήριο 4

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

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Εργαστήριο 4: Πρώτη Άσκηση Προγραμματισμού Python

4.1 Γενικά

Στην άσκηση αυτή γίνεται εισαγωγή στον προγραμματισμό με χρήση της γλώσσας Python. Η Python είναι ίσως η απλούστερη γλώσσα με την οποία θα μπορούσε να ξεκινήσει κάποιος να μαθαίνει προγραμματισμό, αλλά δεν υστερεί καθόλου σε δυνατότητες. Είναι σχετικά πρόσφατη γλώσσα. Ξεκίνησε να αναπτύσσεται στις αρχές της δεκαετίας του 1990 από το μαθηματικό Guido van Rossum. Η γλώσσα έχει γίνει πλέον ιδιαίτερα δημοφιλής στον ακαδημαϊκό και στον επιστημονικό χώρο για τους εξής λόγους:

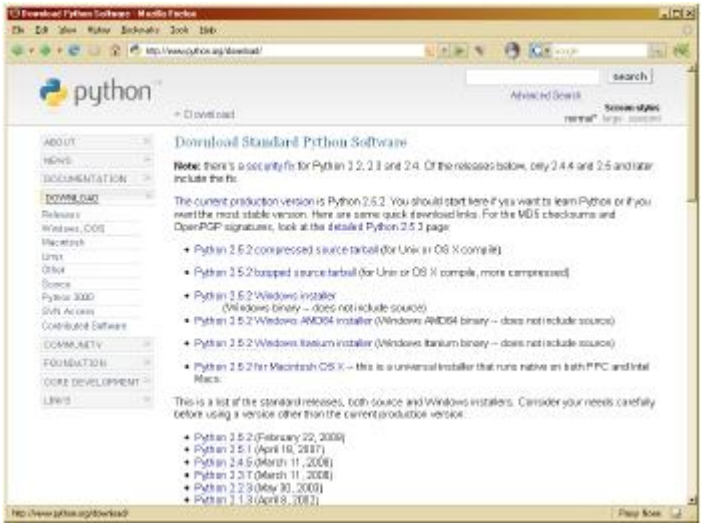
- Μπορεί κανείς να την χρησιμοποιήσει διαδραστικά με τον ίδιο τρόπο που χρησιμοποιεί μια αριθμομηχανή. Κι αυτός είναι ο πιο εύκολος τρόπος να αρχίσει κανείς να μαθαίνει προγραμματισμό.
- Αν και είναι γλώσσα υψηλού επιπέδου, διαθέτει πολύ απλή σύνταξη. Έτσι επιτρέπει στον προγραμματιστή να ασχοληθεί με την επίλυση του προβλήματος κι όχι με τις ιδιαιτερότητες της. Αν και φτάνει σε πολύ υψηλό επίπεδο δυνατοτήτων (πχ. ενσωματώνει συναρτησιακό και αντικειμενοστραφή προγραμματισμό), η απλή της σύνταξη κρατά κρυμμένες αυτές τις δυνατότητες από τους αρχάριους προγραμματιστές και δεν τους αναγκάζει να προγραμματίσουν με κάποιο συγκεκριμένο μοντέλο.
- Μπορεί να συνδυαστεί με άλλες δημοφιλείς γλώσσες **όπως C, C++** και **Java.**
- Αποτελεί ελεύθερο λογισμικό και διατίθεται δωρεάν από το επίσημο site https://www.python.org/ . Υπάρχουν εκδόσεις για κάθε λειτουργικό σύστημα (ακόμα και για κινητά τηλέφωνα). Στο διαδίκτυο μπορεί κανείς να βρει επίσης πληθώρα βιβλιοθηκών και υποστηρικτικού υλικού για την Python που έχουν κατασκευαστεί από διάφορους προγραμματιστές και διατίθενται επίσης ελεύθερα.

4.2 Εγκατάσταση και Εκτέλεση

Κατά πάσα πιθανότητα θα βρείτε την Python ήδη εγκατεστημένη σε κάθε υπολογιστή του υπολογιστικού κέντρου. Όμως για κάθε ενδεχόμενο (πχ. σε περίπτωση που θέλετε να την εγκαταστήσετε στον προσωπικό σας υπολογιστή) η διαδικασία έχει ως εξής:

- Συνδεθείτε στην ιστοσελίδα https://www.python.org/
- Επιλέξτε **"DOWNLOAD"** (από το μενού στα αριστερά της σελίδας) και κατεβάστε το αρχείο που αντιστοιχεί στο λειτουργικό σας σύστημα. Προσέξτε ότι μπορεί να υπάρχει νεώτερη έκδοση[1] και το αρχείο να έχει λίγο διαφορετικό όνομα από αυτό που φαίνεται στην εικόνα.

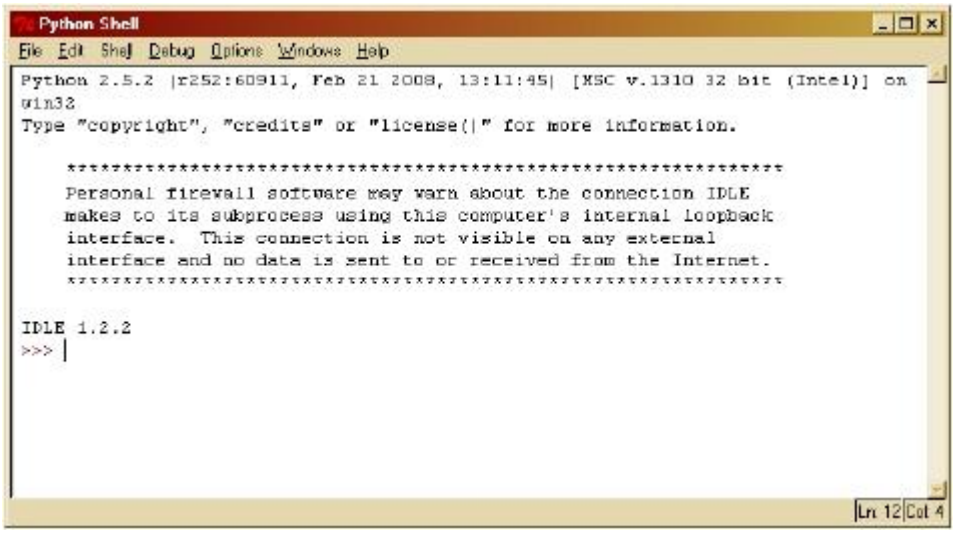
[1] **Περί εκδόσεων**: Στο site της Python θα δείτε ότι υπάρχουν δυο παράλληλες γραμμές εκδόσεων, η 2.x και η 3.x. Προτιμήστε να κατεβάσετε και να εγκαταστήσετε την έκδοση 2 και όχι την 3 (οι ακριβείς λόγοι θα εξηγηθούν στο μάθημα). Η πιο πρόσφατη έκδοση είναι η 2.7 κι από ότι έχει ανακοινωθεί η έκδοση 2 θα σταματήσει εκεί. όμως όταν πρωτογράφτηκαν (2008) αυτές οι ασκήσεις η πιο πρόσφατη έκδοση ήταν η 2.5 και για αυτό θα δείτε τα περισσότερα σχήματα να αναφέρουν αυτή.



- Τρέξτε το εκτελέσιμο αρχείο που κατεβάσατε και ακολουθήστε τις οδηγίες. Το εκτελέσιμο εγκαθιστά και οδηγίες βοήθειας (**on-line help**), αλλά μπορείτε να κατεβάσετε από το ίδιο site αρκετούς οδηγούς μελέτης και εκμάθησης σε εκτυπώσιμη μορφή.

Αμέσως μετά είστε σε θέση να τρέξετε το περιβάλλον της γλώσσας. Από το μενού του λειτουργικού συστήματος επιλέξτε την Python. Υπάρχουν δύο περιβάλλοντα. Η **κονσόλα** (Python Command Line) και το **Integrated Development Enviroment** (IDLE Python GUI). Επιλέξτε το IDLE.

Θα δείτε μία εικόνα όπως η ακόλουθη:



Είστε έτοιμοι να προγραμματίσετε σε Python.

4.3 Χρήση της Python ως Αριθμομηχανής

Μπορούμε να φανταστούμε τη γλώσσα ως μια αριθμομηχανή όπου οι τέσσερις βασικές πράξεις λειτουργούν με τα σύμβολα +, -, *, /. Κάθε φορά που θέλουμε να τυπώσουμε το αποτέλεσμα μιας πράξης γράφουμε την εντολή print. Για παράδειγμα:

```
>>> print 5+3
8
>>> print (7-2)*4
20
```

Η γλώσσα επιτρέπει τη χρήση παρενθέσεων. Επίσης μπορούμε να τυπώνουμε πολλά αποτελέσματα με την ίδια εντολή, αρκεί να χωρίσουμε με κόμματα:

```
>>> print 7+2, 7-2, 7*2, 7/2
9 5 14 3
```

Μια μικρή προσοχή χρειάζεται με την πράξη της διαίρεσης. Η Python εκτελεί ακέραια διαίρεση, εκτός αν δηλώσουμε ότι θέλουμε να κάνει διαίρεση πραγματικών αριθμών. Ο πιο απλός τρόπος να το δηλώσουμε αυτό είναι να γράψουμε τον διαιρετέο ή τον διαιρέτη (ή και τους δύο) ως πραγματικούς. Πχ.:

```
>>> print 7/2, 7/2.0, 7.0/2, 7.0/2.0, 7./2, 7/2., 7./2.
3 3.5 3.5 3.5 3.5 3.5 3.5
```

Και για να συμπληρώσουμε τις βασικές πράξεις, η ύψωση σε δύναμη δηλώνεται ως **. Πχ.:

```
>>> print 9**2, 9**0.5
81 3.0
```

Αξίζει να σημειώσουμε ότι οι ακέραιοι αριθμοί στην Python έχουν απεριόριστο μέγεθος. Πχ.:

```
>>> print 2**1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361
22493198378815695858127594672917553146825187145285692314
04359845775746985748039345677748242309854210746050623711418779541821530464749835
81941267398767559165543946077062914571196477686542167660429831652624386837205668
069376
```

Για να φυλάξουμε ενδιάμεσα αποτελέσματα, οι περισσότερες αριθμομηχανές διαθέτουν έναν ή περισσότερους **καταχωρητές μνήμης** (Memory Register = MR). Η **Python** επιτρέπει πρακτικά απεριόριστο πλήθος καταχωρητών. Τους ονομάζει **μεταβλητές** (variables) και ο προγραμματιστής τους δίνει ένα όνομα (με έναν ή περισσότερους χαρακτήρες) και μια τιμή ως εξής:

```
>>> a=5
```

Δηλώθηκε η μεταβλητή a με τιμή 5.

```
>>> metavliti2=34.45
```

Ομοίως, δηλώθηκε η μεταβλητή metavliti2 με τιμή 34.45. Και τώρα μπορούμε να τις χρησιμοποιήσουμε σε εκφράσεις σα να ήταν αριθμοί:

```
>>> print a, metavliti2, metavliti2/a
>>> 5 34.45 6.89
```

Αυτό που αξίζει να αναφέρουμε τώρα είναι ότι η Python εκτός από ακέραιους και πραγματικούς αριθμούς, χειρίζεται εξίσου καλά και μιγαδικούς. Οι μιγαδικοί δηλώνονται με τη μορφή a+bj όπου το j δηλώνει τη φανταστική μονάδα. Πχ.:

```
>>> n=5+2j
>>> m=2+3j
>>> print n, m, n+m, n-m, n*m, n/m
(5+2j) (2+3j) (7+5j) (3-1j) (4+19j) (1.23076923077-0.846153846154j)
```

Προσέξτε ότι στη διαίρεση μιγαδικών αριθμών η μετατροπή των συνιστωσών σε πραγματικά μεγέθη γίνεται αυτόματα. Όμως δεν γίνεται αυτόματα η μετατροπή πραγματικών σε μιγαδικούς όπου αυτό απαιτείται. Για παράδειγμα, αν θέλουμε να υπολογίσουμε την τετραγωνική ρίζα του 1, αυτό:

```
>>> print (-1)**0.5
Traceback (most recent call last):
  File "<pyshell#44>", line 1, in <module>
    (-1)**0.5
ValueError: negative number cannot be raised to a fractional power
```

Βγάζει λάθος. Όμως αυτό:

```
>>> print (-1+0j)**0.5
(6.12323399574e-17+1j)
```

δίνει μια πολύ καλή προσέγγιση[2] της φανταστικής μονάδας (το πραγματικό μέρος είναι 6.123×10^{-17}).

[2] **Σχετικά με τις προσεγγίσεις**: Συχνά το αποτέλεσμα που δίνει η Python για τα ψηφία μετά την υποδιαστολή δεν είναι ακριβώς αυτό που θα περιμέναμε. Αυτό συμβαίνει επειδή η Python (όπως και οι περισσότερες γλώσσες προγραμματισμού) αποθηκεύει τους πραγματικούς αριθμούς στο δυαδικό σύστημα (με αυστηρά καθορισμένο πλήθος bits) κι αυτή η μετατροπή οδηγεί σε μικρή απώλεια ακρίβειας. Φυσικά υπάρχουν αρκετοί τρόποι για να παρακαμφθεί αυτό το πρόβλημα, όμως δεν είναι αντικείμενο του παρόντος να τους συζητήσουμε.

Τέλος αξίζει να επισημάνουμε ότι η Python διαθέτει μία πληθώρα συναρτήσεων για πράξεις με διανύσματα και πίνακες, τριγωνομετρικούς υπολογισμούς, γραφικές παραστάσεις, κλπ, και ακόμα διαθέτει συναρτήσεις όχι μόνο για μαθηματικές λειτουργίες αλλά για οτιδήποτε μπορεί να κάνει ένας υπολογιστής (πχ. συναρτήσεις για αναπαραγωγή μουσικής και βίντεο, για κρυπτογράφηση αρχείων, για απομακρυσμένη σύνδεση με υπολογιστές στο διαδίκτυο κλπ). Οι συναρτήσεις ομαδοποιούνται σε βιβλιοθήκες ομοειδών συναρτήσεων και μπορείτε ψάχνοντας το on-line help ή σε ένα εγχειρίδιο της Python να ενημερωθείτε για αυτές. Από τη στιγμή που θα εντοπίσετε τη συνάρτηση που σας ενδιαφέρει και τη βιβλιοθήκη που την περιέχει, την εισάγετε στην Python με την εντολή import. Για παράδειγμα, για να χρησιμοποιήσουμε τη συνάρτηση του συνημίτονου (cos) ου βρίσκεται στη βιβλιοθήκη math, δίνουμε πρώτα την εντολή:

```
>>> from math import cos
```

Και στη συνέχεια χρησιμοποιούμε τη συνάρτηση:

```
>>> print cos(0.5)
0.87758256189
```

Αρκεί να εισάγουμε μόνο μια φορά τη συνάρτηση για να τη χρησιμοποιήσουμε στη συνέχεια όσες φορές θέλουμε, μέχρι να κλείσουμε το παράθυρο της Python. Αν θέλουμε να εισάγουμε όλες τις συναρτήσεις μιας βιβλιοθήκης βάζουμε το σύμβολο * στη θέση του ονόματος της συνάρτησης.Πχ.:

```
>>> from math import *
>>> print cos(0.5), sin(0.5), tan(0.5)
0.87758256189 0.479425538604 0.546302489844
```

Και βέβαια εκτός από τις ενσωματωμένες βιβλιοθήκες και συναρτήσεις, ο κάθε προγραμματιστής μπορεί να φτιάξει δικές του. Στο διαδίκτυο υπάρχει πληθώρα βιβλιοθηκών σε Python για οποιαδήποτε λειτουργία φανταστεί κανείς. Πολύ γρήγορα θα μάθετε να φτιάχνετε κι εσείς τις δικές σας.

4.4 Το Πρώτο μας Πρόγραμμα

Τώρα γνωρίζουμε αρκετά για να επιχειρήσουμε ένα απλό πρόγραμμα. Έστω ότι θέλουμε να φτιάξουμε ένα πρόγραμμα που θα υπολογίζει τις λύσεις της δευτεροβάθμιας εξίσωσης $\mathbf{2x^2-5x+2=0}$. Ας ξεκινήσουμε όπως γνωρίζουμε, δίνοντας τις εντολές μια-μια όπως σε μια αριθμομηχανή, κι ασ υπολογίσουμε πρώτα τη Διακρίνουσα:

```
>>> a=2
>>> b=-5
>>> c=2
>>> diak=b*b-4*a*c
>>> print diak
9
```

Βλέπουμε ότι η διακρίνουσα είναι θετική, επομένως μπορούμε να υπολογίσουμε την τετραγωνική της ρίζα:

```
>>> rd=diak**0.5
```

και να συνεχίσουμε κατά τα γνωστά:

```
>>> x1=(-b-rd)/(2*a)
```

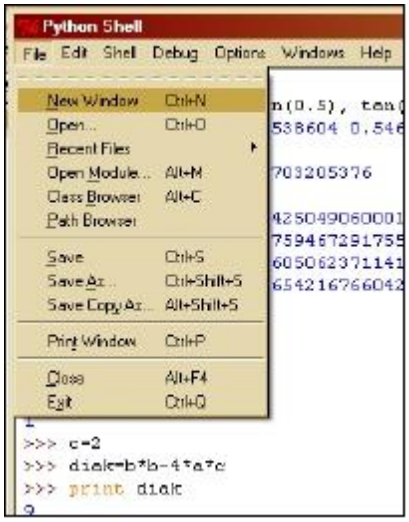
```
>>> x2=(-b+rd)/(2*a)
>> print x1, x2
0.5 2.0
```

Αυτές είναι οι λύσεις. Δείτε τώρα έναν πιο ωραίο τρόπο να τις τυπώσουμε:

```
>>> print 'x1=', x1, "        x2=", x2
x1= 0.5         x2= 2.0
```

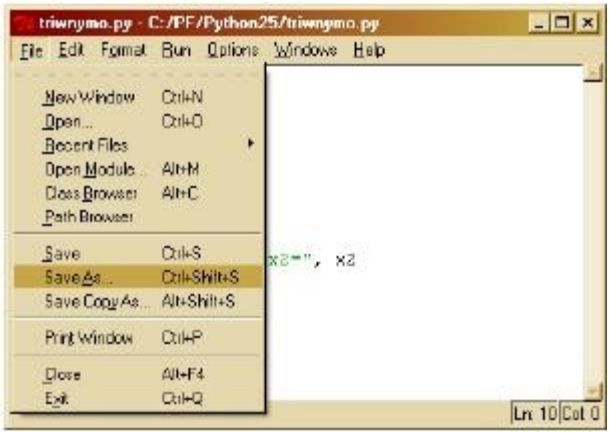
Η εντολή print δεν τυπώνει μόνο αριθμούς και περιεχόμενο μεταβλητών. Τυπώνει και οτιδήποτε βάλουμε μέσα σε (απλά ή διπλά) εισαγωγικά ως κείμενο. Ακόμα και τα κενά.

Και τώρα που λύσαμε την εξίσωση **$2x^2$-5x+2 =0** ας προσπαθήσουμε να λύσουμε άλλη μια παρόμοια, ας πούμε την **$2x^2$-5x+3 =0**. Προσέξτε ότι δεν αρκεί να αλλάξουμε την τιμή της μεταβλητής c από 2 σε 3. Θα πρέπει να ξαναδώσουμε όλες τις επόμενες εντολές μια-μια, κι ας είναι ακριβώς οι ίδιες! Σκεφτείτε πόσο βολικό θα ήταν να λέγαμε στην Python να χρησιμοποιεί τις ίδιες εντολές σε ένα αρχείο και να πούμε στη γλώσσα να τις εκτελέσει. Ο πιο απλός τρόπος είναι επιλέγοντας File>New window (Ctrl+N) από το μενού του IDLE...

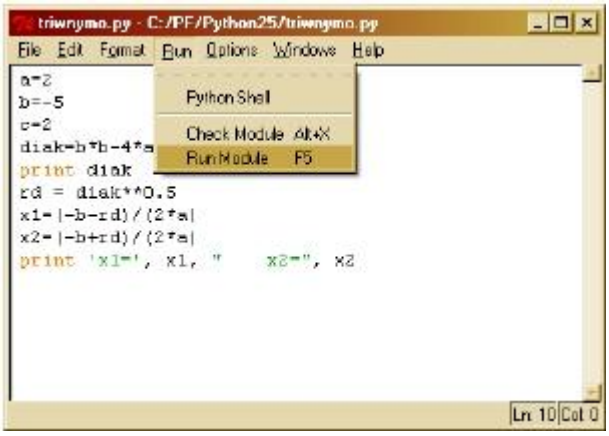


... και στη συνέχεια στο νέο παράθυρο που θα ανοίξει γράφουμε τις εντολές (ή τις κάνουμε copy-paste από το αρχικό, χωρίς όμως του χαρακτήρες '>>>' στην αρχή κάθε γραμμής) ...

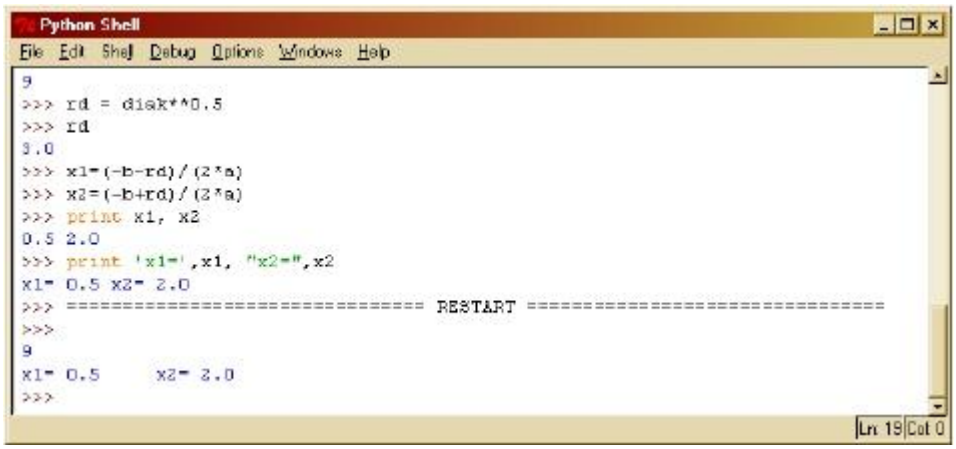
... και το σώζουμε με **Save As** ... (Ctrl+Shigt+S). Για όνομα αρχείου βάζουμε ότι θέλουμε, όμως η προέκταση πρέπει να είναι **.py**. Πχ. "**triwnymo.py**".



Και τώρα αρκεί να επιλέξουμε **Run>Run Module (F5)...**



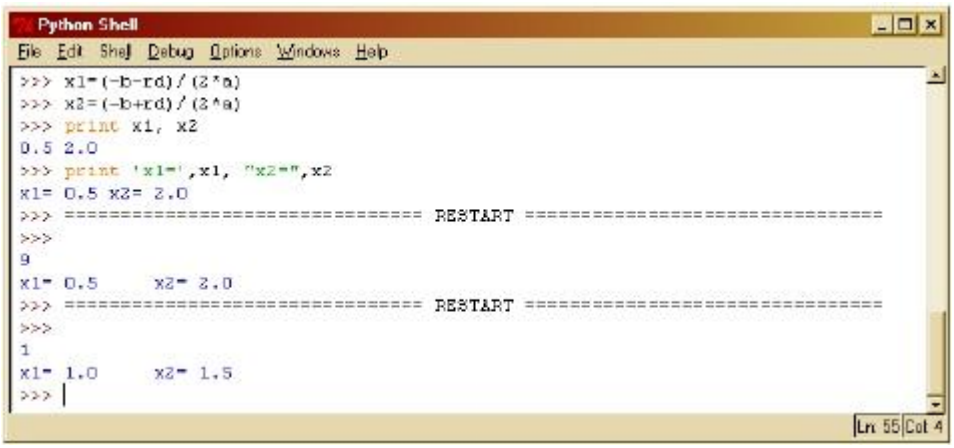
... και θα δούμε όλες τις εντολές να εκτελούνται στο αρχικό παράθυρο:



Αν τώρα αλλάξουμε το c=3 , ξανασώσουμε το αρχείο και να ξανατρέξουμε, θα έχουμε βρει τις λύσεις της 2ης εξίσωσης $\mathbf{2x^2-5x+3=0}$.

Λέμε ότι οι εντολές που έχουμε γράψει στο αρχείο triwnymo.py αποτελούν ένα πρόγραμμα. Φυσικά αν θέλουμε να λύσουμε ένα και μόνο πρόβλημα, δεν υπάρχει λόγος να γράψουμε πρόγραμμα. Αν όμως θέλουμε να λύσουμε μια κατηγορία ομοειδών προβλημάτων (πχ. να λύσουμε πολλές εξισώσεις

2ου βαθμού), τότε κερδίζουμε πολύ χρόνο φτιάχνοντας ένα κατάλληλο πρόγραμμα. Όπως είδαμε, αρκεί να αλλάξουμε μόνο τους συντελεστές a, b, c για να λύσουμε μια διαφορετική εξίσωση με το ίδιο πρόγραμμα.



Όμως για να το πετύχουμε αυτό θα πρέπει να είμαστε πολύ προνοητικοί κατά την κατασκευή του προγράμματος, ώστε να ανταποκρίνεται σωστά σε κάθε δυνατό συνδυασμό παραμέτρων. Το πρόγραμμα που έχουμε φτιάξει δε δουλεύει σωστά αν τύχει η διακρίνουσα να είναι αρνητική (δοκιμάστε το). Ένα πιο προνοητικά κατασκευασμένο πρόγραμμα μόλις εντόπιζε αρνητική διακρίνουσα θα τύπωνε κατάλληλο μήνυμα και δε θα προχωρούσε στον υπολογισμό της τετραγωνικής ρίζας. Ή θα ρωτούσε αν θέλουμε να υπολογίσει μιγαδικές λύσεις και θα προσωρούσε διαφορετικά.

Μια άλλη παράμετρος στην κατασκευή προγραμμάτων είναι ότι πολύ συχνά θα χρειαστεί να το χρησιμοποιούν και άλλοι άνθρωποι (συνήθως τους λέμε χρήστες) εκτός από τον προγραμματιστή. Οι χρήστες δεν έχουν καμία υποχρέωση να γνωρίζουν πως δουλεύει το πρόγραμμα, τί κάνει η κάθε μεταβλητή και ποιές ακριβώς τιμές θα πρέπει να αλλάξουν στον κώδικα για να λυθεί σωστά το πρόβλημά τους. Για αυτό το λόγο οι καλοί προγραμματιστές φτιάχνουν τα προγράμματά τους με τέτοιο τρόπο ώστε να μη χρειάζονται αλλαγές στον κώδικα κάθε φορά που χρησιμοποιούνται για την επίλυση ενός νέου προβλήματος. Για παράδειγμα αντί να αλλάζουν τις τιμές των μεταβλητών στον κώδικα, βάζουν το πρόγραμμα στην αρχή να κάνει τις κατάλληλες ερωτήσεις στο χρήστη και από τις απαντήσεις του διαμορφώνουν τις τιμές στις μεταβλητές. Όμως για να κάνουμε τέτοιες βελτιώσεις στα προγράμματά μας χρειάζεται να μάθουμε μερικές εντολές ακόμη, που είναι το αντικείμενο της επόμενης ενότητας.

4.5 Εντολές input, if, else

Είδαμε προηγουμένως την εντολή print ου χρησιμοποιείται για να τυπώσουμε δεδομένα στην οθόνη του υπολογιστή μας. Την αντίθετη λειτουργία πραγματοποιεί μια συνάρτηση με το όνομα input. Χρησιμοποιείται ως εξής:

```
>>> a=input()
4
>>> print a
4
```

Εδώ αντί να δώσουμε μια τιμή στη μεταβλητή a της λέμε να πάρει την τιμή που θα της δώσει η συνάρτηση input. Στη συνέχεια, ότι γράψουμε στο πληκτρολόγιο κατά την εκτέλεση της input περνάει ως τιμή στην a, κι όταν τυπώσουμε την a βλέπουμε την τιμή που δώσαμε.

Μέσα στην παρένθεση της input μπορούμε προαιρετικά να βάλουμε ένα μήνυμα που θα τυπώνεται τη στιγμή που εκτελείται:

```
>>> b=input('Γώζε έναν απιθμό:')
Δώσε έναν αριθμό:12
>>> print b
12
```

Συχνά κάποια εντολή χρειάζεται να εκτελεστεί υπό συνθήκη. Αν η συνθήκη ισχύει τότε η εντολή (ή οι εντολές -μπορεί να είναι και περισσότερες από μία) πρέπει να εκτελεστεί, διαφορετικά όχι. Αυτό το πετυχαίνει η εντολή if. Δείτε το παρακάτω τμήμα προγράμματος:

```
...
a=input(‘Δώσε τον διαιρετέο:’)
b=input(‘Δώσε τον διαιρέτη:’)
if b==0:
        print ‘Ο διαιρέτης δεν πρέπει να είναι μηδέν’
        b= input(‘Ξαναδώσε ηον διαιρέτη:’)
print ‘Το πηλίκο της διαίρεσης είναι:’, a/b
...
```

Αυτό το τμήμα κώδικα ζητά από το χρήστη να πληκτρολογήσει δύο αριθμούς, τον a (διαιρετέο) και τον b (διαιρέτη) με σκοπό να τους διαιρέσει και να τυπώσει το αποτέλεσμα. Όμως πριν γίνει η διαίρεση πρέπει να ελεγχθεί μήπως ο διαιρέτης είναι μηδέν. Αυτό κάνει η εντολή **if**. Ελέγχει τη συνθήκη b==0 (δηλαδή αν το b είναι μηδέν) και αν αυτή ισχύει, τότε εκτελεί τις δύο εντολές που είναι γραμμένες μια στήλη δεξιότερα, κάτω από την **if** (λέγονται εσωτερικές εντολές της **if**) και αμέσως μετά συνεχίζει στην εκτέλεση της **print**. Αν όμως η συνθήκη δεν ισχύει, οι δύο εσωτερικές εντολές δεν εκτελούνται και πηγαίνουμε κατευθείαν στην **print**. (**Σημείωση**: Στην **Python** τα TABs έχουν συντακτικό ρόλο. Με αυτόν τον τρόπο αποφεύγεται η χρήση συμβόλων για την αρχή και το τέλος των ομάδων εντολών και ο κώδικας διατηρείται απλοποιημένος κι ευανάγνωστος).

Προσέξτε ότι ο έλεγχος ισότητας συμβολίζεται με δύο σύμβολα "=". Το ένα όπως ξέρουμε σημαίνει καταχώρηση τιμής σε μεταβλητή, κι όχι έλεγχο ισότητας. Το διπλό "==" λέγεται τελεστής. Οι τελεστές που μπορούν να χρησιμοποιηθούν στην if είναι:

<	μικρότερο
>	μεγαλύτερο
<=	μικρότερο ή ίσο
>=	μεγαλύτερο ή ίσο
==	ίσο
!=	διαφορετικό

Μια παραλλαγή της if χρησιμοποιεί την λέξη else για να ορίσει δύο ομάδες εσωτερικών εντολών, μια που θα εκτελείται όταν ισχύει η συνθήκη και μια που θα εκτελείται όταν δεν ισχύει. Δείτε ένα παράδειγμα:

```
...
a=input("Δώσε έναν αριθμό:“)
if a>=0:
        print 'Ο αρθμός είναι θετικός (ή μηδέν).'
        b=a
else:
```

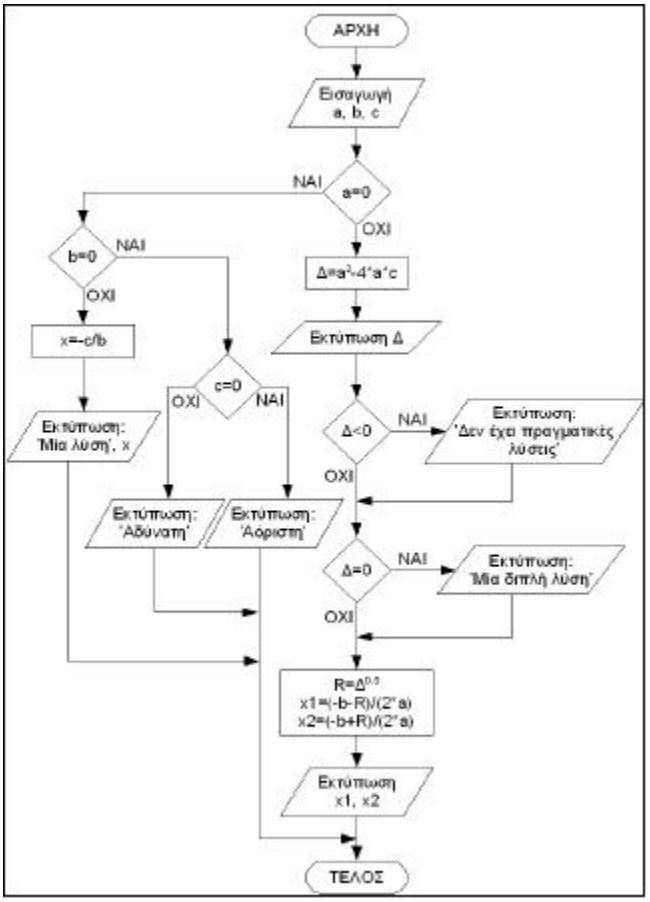
```
            print 'Ο αριθμός είναι αρνητικός'
            b=-a
    print 'Η απόλυτη τιμή του αριθμού είναι:', b
…
```

Αυτό το τμήμα κώδικα διαβάζει έναν αριθμό από το πληκτρολόγιο, ελέγχει αν είναι θετικός ή αρνητικός και τυπώνει την απόλυτη τιμή του. Οι δύο εσωτερικές εντολές που βρίσκονται κάτω από την if εκτελούνται μόνο όταν η συνθήκη a>=0 ισχύει. Αντίθετα, οι δύο εσωτερικές εντολές κάτω από την else εκτελούνται μόνο όταν η συνθήκη a>=0 δεν ισχύει. Η print εκτελείται και στις δύο περιπτώσεις, αφού δεν είναι εσωτερική εντολή.

Επίσης, αξίζει να σημειώσουμε ότι στις εσωτερικές εντολές μιας if μπορούν να βρίσκονται οποιεσδήποτε εντολές, ακόμα κι άλλες if, όπως θα δούμε στην επόμενη ενότητα.

4.6 Το Πλήρες Πρόγραμμα για την Επίλυση Εξισώσεων 2^{ου} Βαθμού

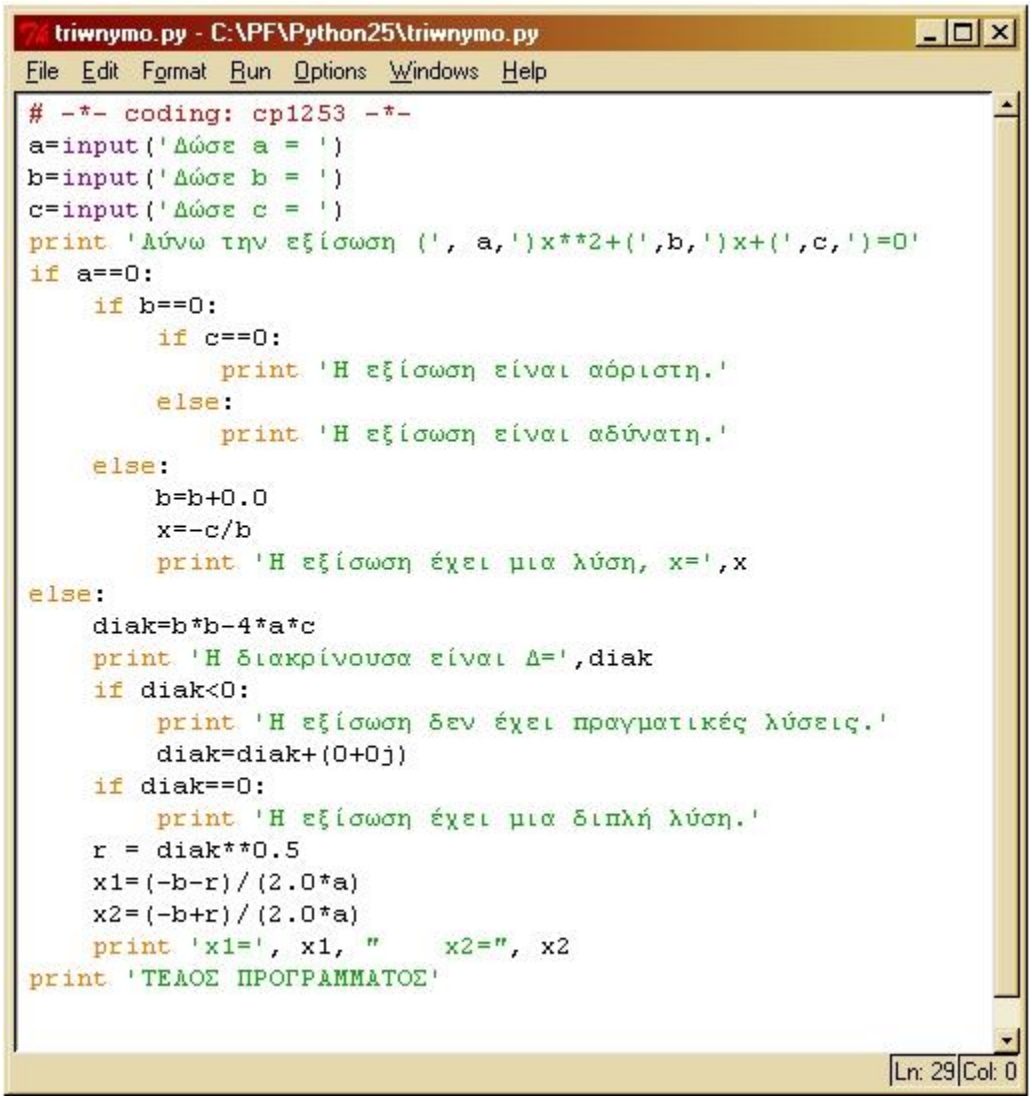
Τώρα γνωρίζουμε αρκετά για να φτιάξουμε ένα πλήρες πρόγραμμα για την επίλυση των εξισώσεων 2^{ου} βαθμού.



Αλλά πριν αρχίσουμε να γράφουμε τον κώδικα ενός σύνθετου προγράμματος, μια καλή πρακτική είναι να σχεδιάσουμε πρώτα το διάγραμμα ροής. Έτσι αποφεύγουμε να ξεχάσουμε κάποια βασική λειτουργία του προγράμματος και γλιτώνουμε από πολλές διορθώσεις στον κώδικα. Να πώς θα μπορούσαμε να σχεδιάσουμε το διάγραμμα ροής για την επίλυση της εξίσωσης 2^{ου} βαθμού:

Και τώρα είναι πολύ πιο εύκολο να γράψουμε τον κώδικα του προγράμματος. Τα καίρια σημεία του διαγράμματος είναι οι ρόμβοι. Κάθε ρόμβος αντιστοιχεί σε μία εντολή if με τον έλεγχο που περιέχει. Ότι βρίσκεται στον κλάδο **ΝΑΙ**, στον κώδικα θα εμφανιστεί στο πεδίο των εσωτερικών εντολών, κάτω

από την if. Ότι βρίσκεται στον κλάδο **ΟΧΙ**, στον κώδικα θα εμφανιστεί κάτω από την else. Ακολουθεί ο κώδικας[3]:



Παρατηρήστε ότι το διάγραμμα ροής καθορίζει τη δομή του προγράμματος κι όχι τις λεπτομέρειες του κώδικα. Δηλαδή δεν μας δεσμεύει στα ακριβή ονόματα των μεταβλητών που θα χρησιμοποιήσουμε τελικά (αν και καλό είναι να υπάρχει μια εμφανής αντιστοιχία), ούτε εξαντλεί όλα τα ενημερωτικά μηνύματα που τελικά αποφασίζουμε να εντάξουμε στο πρόγραμμα μας. Επίσης (και αυτό είναι το πιο σημαντικό) δεν ασχολείται καθόλου με τις προγραμματιστικές λεπτομέρειες που εξαρτώνται από τη γλώσσα και αφορούν τον τρόπο υπολογισμού των τιμών των μεταβλητών και τις ιδιαιτερότητες των επιμέρους εντολών και συναρτήσεων.

Για παράδειγμα, προσέξτε ότι στον υπολογισμό των x1 και x2 οι παρανομαστές στον κώδικα είναι **(2.0*a)** αντί **(2*a)** που φαίνεται στο διάγραμμα ροής. Όταν φτιάχνουμε το διάγραμμα ροής δε σκεφτόμαστε τις τεχνικές λεπτομέρειες του τρόπου που κάνει διαίρεση η Python, όμως στον κώδικα πρέπει να τις σκεφτούμε και να προνοήσουμε ώστε να δουλέψει σωστά. Για τον ίδιο λόγο, στην περίπτωση που **a=0** και b είναι μηδέν, στον κώδικα έχουμε βάλει την εντολή:

[3] Η πρώτη γραμμή (# -*- coding -*-) δηλώνει ότι στο αρχείο χρησιμοποιούνται ελληνικοί χαρακτήρες (codepage 1253). Δεν είναι απαραίτητη όταν δουλεύουμε με ένα μόνο αρχείο σε έναν υπολογιστή , όμως αν το πρόγραμμά μας αποτελείται από πολλά αρχεία ή πρόκειται να το μεταφέρουμε σε άλλον υπολογιστή, γλιτώνουμε από πολλές ασυμβατότητες αν δηλώσουμε την κωδικοποίηση με την οποία γράφτηκε, βάζοντας στην 1η ή στην 2η γραμμή του κάθε αρχείου αυτή τη δήλωση. Οι συνηθέστεροι κωδικοί είναι cp1253 (Windows Ελληνικά) και utf-8 (για Unicode).

b=b+0.0

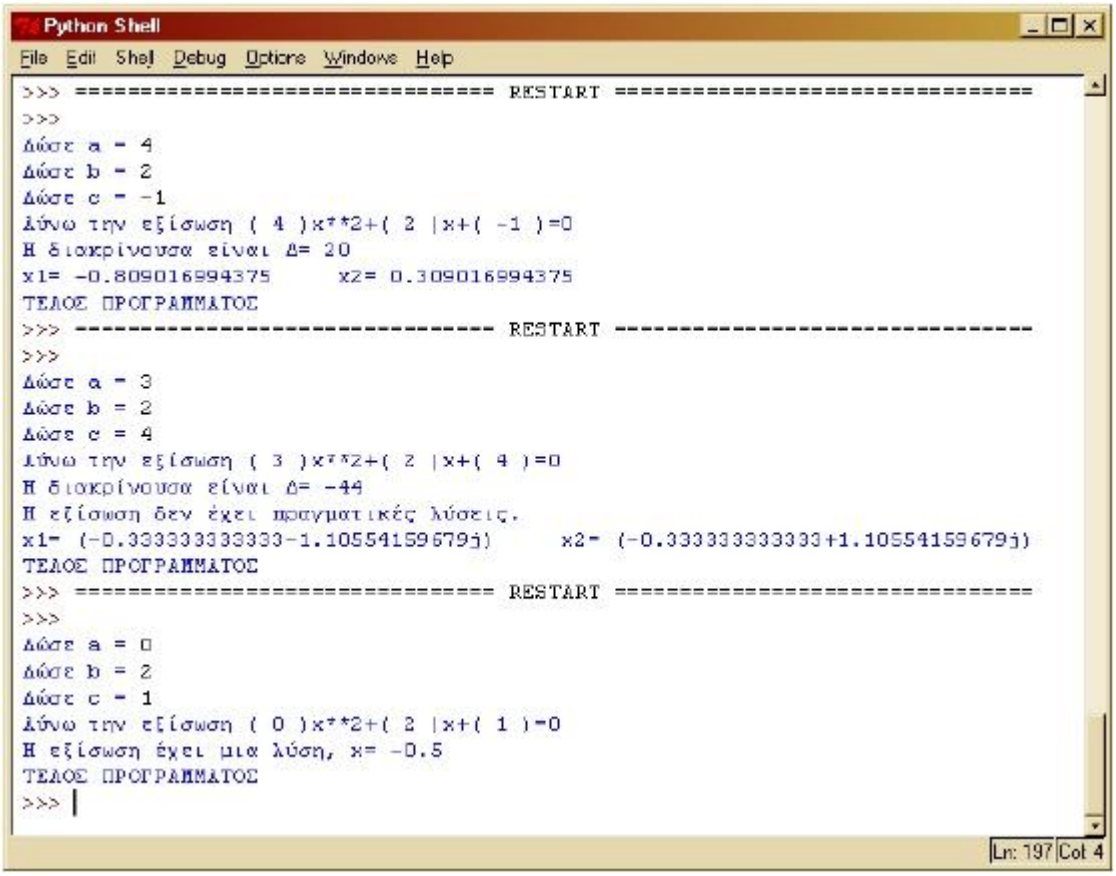
Ουσιαστικά δεν αλλάζουμε την τιμή της b, αφού προσθέτουμε μηδέν, όμως προσθέσαμε το μηδέν με πραγματική μορφή. Αυτός είναι ένας τρόπος να πούμε στην **Python** να θεωρεί στο εξής την b ως πραγματική μεταβλητή, ώστε να αποφευχθεί η ακέραια διαίρεση.

Στο ίδιο πνεύμα, προσέξτε ότι στο διάγραμμα ροής υπολογίζεται η τετραγωνική ρίζα του Δ ακόμα κι αν αυτό είναι αρνητικό. Όμως στον κώδικα η Διακρίνουσα χρειάζεται πρώτα να μετατραπεί σε μιγαδικό αριθμό για να λειτουργήσει σωστά η διαδικασία. Αυτή η μετατροπή γίνεται με την εντολή:

diak=diak +(0+0j)

Η εντολή αυτή δεν αλλάζει την τιμή της μεταβλητής **diak** αφού προσθέτει μηδέν, όμως προσθέτει το μηδέν με μιγαδική μορφή κι έτσι αναγκάζει την **diak** στο εξής να είναι μιγαδική μεταβλητή.

Ακολουθούν μερικά παραδείγματα από την εκτέλεση του προγράμματος:



4.7 Ασκήσεις

Εκπονήστε τις παρακάτω ασκήσεις. Ομαδοποιήστε τα αρχεία με τις απαντήσεις σας σε ένα ενιαίο συμπιεσμένο αρχείο και παραδώστε το στην ηλεκτρονική θυρίδα του Moodle σύμφωνα με τις οδηγίες που θα σας δοθούν στον εργαστήριο. Δώστε ιδιαίτερη προσοχή στην πληρότητα και την εμφάνιση των απαντήσεων σας.

Άσκηση #1

Πληκτρολογήστε τον κώδικα της ενότητας **4.6** και τρέξτε τον. Πειραματιστείτε με τη λειτουργία και αν θέλετε βελτιώστε έτσι ώστε να επαναλαμβάνεται η διαδικασία μέχρι ο χρήστης να δώσει την εντολή τερματισμού.

Άσκηση #2

Παρακάτω περιγράφεται η διαδικασία[4] επίλυσης της πολυωνυμικής εξίσωσης 3ου βαθμού **$Ax^3+Bx^2+Cx+D=0$**. Χρησιμοποιήστε το Synergo και αποτυπώστε την σε ένα διάγραμμα ροής.

Πώς λύνουμε την εξίσωση $Ax^3+Bx^2+Cx+D=0$

Αν **A=0** τότε η εξίσωση εκφυλίζεται σε 2ου βαθμού και λύνεται σύμφωνα με τον κώδικα της Άσκησης #1. Διαφορετικά διαιρούμε όλους τους συντελεστές δια **A** και η εξίσωση γίνεται:

$x^3+ax^2+bx+c=0$, όπου $\mathbf{a} = \frac{\mathbf{B}}{\mathbf{A}},\ \mathbf{b} = \frac{\mathbf{C}}{\mathbf{A}},\ \mathbf{c} = \frac{\mathbf{D}}{\mathbf{A}}$.

Η διακρίνουσα υπολογίζεται ως: **$\Delta=18abc-4a^3c+a^2b^2-4b^3-27c2$**

- Αν **Δ>0**, η εξίσωση έχει 3 πραγματικές λύσεις.
- Αν **Δ=0**, έχει λιγότερες από 3 πραγματικές λύσεις (κάποιες συμπίπτουν).
- Αν **Δ<0**, η εξίσωση έχει 1 πραγματική λύση και 2 μιγαδικές.

Για να βρούμε τις λύσεις, υπολογίζουμε πρώτα τις ενδιάμεσες τιμές:

$$\mathbf{p} = \mathbf{b} - \frac{\mathbf{a}^2}{\mathbf{3}} \qquad \mathbf{q} = \mathbf{c} + \frac{\mathbf{2a^3-9ab}}{\mathbf{27}} \qquad u = \sqrt[3]{-\frac{\mathbf{q}}{\mathbf{2}} \pm \sqrt{\frac{\mathbf{q^2}}{\mathbf{4}} + \frac{\mathbf{p^3}}{\mathbf{27}}}}$$

Προσοχή, η τελευταία σχέση μας δίνει 6 τιμές για το u, όχι μόνο 2 (για κάθε τιμή του πρόσημου της τετραγωνικής ρίζας, η κυβική ρίζα δίνει 3 τιμές με διαφορά φάσης 270 μοίρες στο μιγαδικό επίπεδο).

Οι λύσεις τις εξίσωσης υπολογίζονται τελικά από τη σχέση: $\mathbf{x} = -\frac{\mathbf{p}}{\mathbf{3u}} + \mathbf{u} - \frac{\mathbf{a}}{\mathbf{3}}$

Βάζοντας τις 6 τιμές του u στην παραπάνω σχέση, κάποιες τιμές συμπίπτουν και τελικά εμφανίζονται το πολύ 3 διαφορετικές τιμές για το x.

Άσκηση #3

Με τη βοήθεια του διαγράμματος ροής που φτιάξατε στην Άσκηση #2, γράψτε ένα πρόγραμμα στη γλώσσα **Python** που θα λύνει την εξίσωση 3ου βαθμού σύμφωνα με τους συντελεστές **A, B, C, D** που θα δίνει ο χρήστης[5]. Γράψτε όσο το δυνατόν πληρέστερο πρόγραμμα.

---

[4] Για περισσότερες λεπτομέρειες, δείτε: **http://en.wikipedia.org/wiki/Cubic_polynomial**

[5] Αν επιθυμείτε να ελέγξετε τη λύση σας μπορείτε να χρησιμοποιήσετε το εξής πρόγραμμα επίλυσης εξισώσεων (μέχρι και 4ου βαθμού): http://www.freewebs.com/brianjs/ultimateequationsolver.htm

Σημειώματα

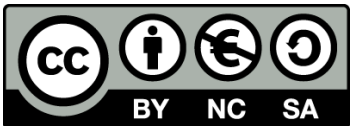
Σημείωμα Ιστορικού Εκδόσεων Έργου

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 σχεδιαστικό εργαλείο Synergo και 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.