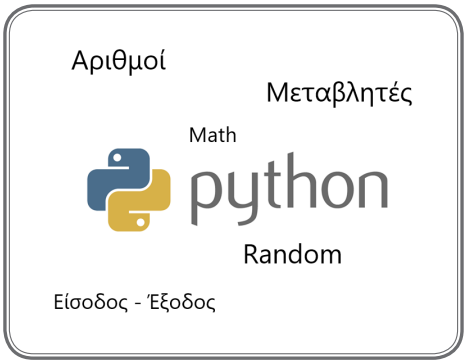
ΚΕΦΑΛΑΙΟ 2

ΜΕΤΑΒΛΗΤΕΣ & ΑΡΙΘΜΟΙ

1.1 Εισαγωγή

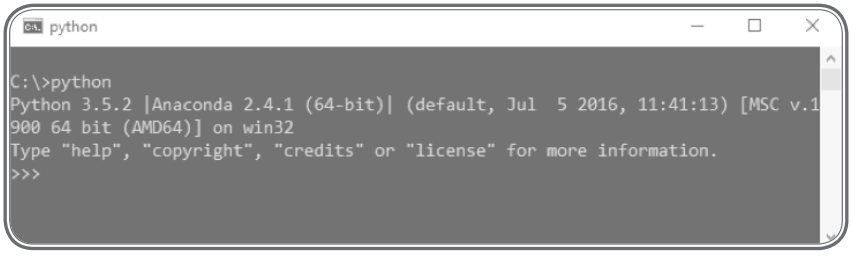


Στο κεφάλαιο αυτό περιγράφεται ο τρόπος που γίνονται οι αριθμητικές πράξεις στο κέλυφος της γλώσσας. Εισάγεται η έννοια της μεταβλητής και αναφέρονται οι βασικοί αριθμητικοί τύποι δεδομένων της γλώσσας, ώστε να μπορούν να γίνουν απλοί, αλλά και σύνθετοι αριθμητικοί και λογικοί υπολογισμοί. Παράλληλα, περιγράφονται οι τρόποι σύγκρισης δεδομένων που θα χρειαστούν στο Κεφάλαιο 3 για τη λήψη αποφάσεων. Σημειώνονται κάποια υπολογιστικά προβλήματα που δημιουργούνται και οι περιορισμοί που υπάρχουν, καθώς και τρόποι που μπορούν να τα αντιμετωπίσουν αποδοτικά.

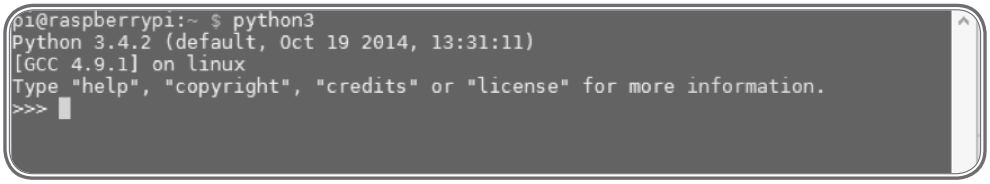
Περιγράφονται επίσης τρόποι εμφάνισης – παρουσίασης των δεδομένων ανάλογα με τις επιλογές του χρήστη. Δεν θα μπορούσε να λείψει μια πρώτη εισαγωγή στη χρήση βιβλιοθηκών της python και στην επέκταση των δυνατοτήτων της γλώσσας με την προσθήκη επιπλέον στοιχείων. Τέλος, γίνεται αναφορά σε μεθόδους χειρισμού στοιχειωδών πληροφοριών σε επίπεδο δυαδικών στοιχείων.

1.2 Αριθμητικές πράξεις

Το περιβάλλον της python (κονσόλα ή κέλυφος) είναι διαθέσιμο με πολλούς τρόπους για την πραγματοποίηση αριθμητικών υπολογισμών. Σε περιβάλλον windows, από τη γραμμή εντολών, με την πληκτρολόγηση της εντολής python ή python3.



Σε περιβάλλον linux (η επόμενη εικόνα είναι από το λειτουργικό raspbian σε Raspberry Pi 3), αφού ανοίξει μια κονσόλα – τερματικό, το περιβάλλον της python ανοίγει με την εντολή python3.



Με την είσοδο στο περιβάλλον της python μπορεί να γίνει ένας υπολογισμός γράφοντας τις κατάλληλες εντολές και πατώντας το πλήκτρο enter. Προκειμένου να γίνει οποιαδήποτε πράξη είναι απαραίτητη η χρήση τελεστών που περιγράφονται αμέσως μετά.

Η κονσόλα ή τερματικό επιτρέπει την εκτέλεση πράξεων και την πραγματοποίηση κάθε είδους υπολογισμού. Στην πραγματικότητα όμως είναι ένα πανίσχυρο εργαλείο που χρησιμοποιείται συνήθως ή για την εκτέλεση μεμονωμένων εντολών ή, κυρίως, για την άμεση δοκιμή μικρών κομματιών κώδικα για την υλοποίηση κάποιου αλγόριθμου (*prototyping*), πριν από την ενσωμάτωσή τους σε κάποιο μεγαλύτερο σενάριο – πρόγραμμα.

1.3 Κατηγορίες τελεστών

Κάθε γλώσσα έχει ειδικά σύμβολα και λέξεις για την πραγματοποίηση υπολογισμών κάθε είδους. Αυτά τα σύμβολα ή λέξεις ονομάζονται *τελεστές* (*operators*) και προσδιορίζουν κάθε φορά το είδος των υπολογισμών. Πιο συγκεκριμένα, υπάρχουν τρεις βασικές κατηγορίες τελεστών, χωρίς να αναφέρονται οι περιπτώσεις τελεστών – μεθόδων κλάσεων που χρησιμοποιούνται σε αντικείμενα (βλ. Κεφάλαιο 11: αντικειμενοστραφής προγραμματισμός). Αυτές οι κατηγορίες είναι:

- **Αριθμητικοί**: με βάση τους οποίους γίνονται όλες οι αριθμητικές πράξεις. Παρουσιάζονται αμέσως μετά.
- **Συγκριτικοί ή σχεσιακοί**: υπεύθυνοι για τη σύγκριση δεδομένων και τον έλεγχο συνθηκών. Αναλύονται στο Κεφάλαιο 3 και χρησιμοποιούνται σε εντολές ελέγχου – επιλογής και επανάληψης.
- **Λογικοί**: κατάλληλοι για τη δημιουργία λογικών ελέγχων ή συσχετίσεων πολλαπλών συγκρίσεων. Παρουσιάζονται στο Κεφάλαιο 3.

Η σειρά με την οποία αναφέρθηκαν είναι και η ιεράρχησή τους. Έτσι, σε μια σύνθετη έκφραση που περιέχει όλα τα παραπάνω, πρώτα γίνονται οι αριθμητικές πράξεις (προκύπτει αριθμητικό αποτέλεσμα), μετά οι συγκρίσεις (προκύπτει λογικό αποτέλεσμα: True/False) και μετά οι λογικές πράξεις.

Εκτός από τους πιο πάνω βασικούς τελεστές υπάρχουν και άλλοι οι οποίοι περιγράφονται στη συνέχεια:

- **Δυαδικοί** (bitwise, &, >> κ.λπ.): χειρισμός δεδομένων βάσει της δυαδικής αναπαράστασής τους.
- **Σύνθετοι τελεστές εκχώρησης** (+=, /= κ.λπ.): για την αποθήκευση δεδομένων στη μνήμη του υπολογιστή.
- **Τελεστές υποσυνόλων** (in – not in): για τον έλεγχο σχέσεων ανάμεσα σε ομάδες δεδομένων.
- **Τελεστές ταυτότητας** (is – is not): για τον αν δύο αντικείμενα είναι ταυτόσημα ή όχι.
- **Μοναδιαίοι** (unary): για την επισήμανση ή αντιστροφή πρόσημων αριθμών αλλά και χειρισμό συνόλων.

1.3.1 Αριθμητικοί τελεστές

Κάθε αριθμητική πράξη πραγματοποιείται με τη βοήθεια των αριθμητικών τελεστών. Αυτοί δέχονται πάντα δύο ορίσματα – αριθμούς από τις δύο πλευρές τους και μπορούν να εμφανίζονται είτε σε μεμονωμένες εντολές ή μέσα σε άλλες. Στον Πίνακα 2-1 εμφανίζονται οι αριθμητικοί τελεστές.

ΠΙΝΑΚΑΣ 2-1 | ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

Τελεστής	Λειτουργία	Παράδειγμα	Αποτέλεσμα
+	Πρόσθεση	`5+12`	17
-	Αφαίρεση	`12-3`	9
*	Πολλαπλασιασμός	`2*5`	10
/	Διαίρεση	`5/2`[1]	2.5
//	Ακέραια διαίρεση	`5//2`	2
%	Υπόλοιπο διαίρεσης	`5%2`	1
**	Δύναμη	`2**5`	32
( )	Παρενθέσεις	`(1+2+3)/3`	2

Η ιεραρχία μεταξύ των αριθμητικών πράξεων είναι:

1. **( )**: η χρήση παρενθέσεων γίνεται για την αλλαγή της προτεραιότητας των πράξεων. Περιττές παρενθέσεις δεν είναι λάθος αρκεί να τοποθετούνται στο σωστό σημείο. Όσες παρενθέσεις ανοίγουν από αριστερά, τόσες πρέπει και να κλείνουν από δεξιά. Τα άγκιστρα και οι αγκύλες [] και {} χρησιμοποιούνται για άλλο σκοπό και όχι για αριθμητικές πράξεις, π.χ. σε λίστες, συμβολοσειρές και λεξικά. Έτσι, ο υπολογισμός ενός μέσου όρου, π.χ. των 1, 2 και 3, που είναι 2, θα

1. Στην έκδοση 2.x της python χρειάζεται προσοχή όταν χρησιμοποιείται το σύμβολο "/" με ακέραιους αριθμούς. Σ' αυτή την περίπτωση, γίνεται ακέραια διαίρεση (με στρογγυλοποίηση προς τα κάτω). Π.χ. 3/2 δίνει 1 (ακέραιος) και όχι 1.5 (πραγματικός), επειδή και οι δύο αριθμοί στη διαίρεση είναι ακέραιοι. Για να γίνει η ακριβής διαίρεση πρέπει ο ένας να είναι πραγματικός, πχ 3.0/2 δίνει 1.5 (!!!)

γίνει `(1+2+3)/3` και όχι `1+2+3/3`. Η τελευταία παράσταση, αν και είναι συντακτικά σωστή (ο διερμηνευτής της γλώσσας δεν θα εμφανίσει κάποιο σφάλμα – θα επιστρέψει την τιμή 4), ωστόσο δεν λύνει το δοσμένο πρόβλημα αλλά κάτι άλλο (κάνει πρώτα διαίρεση και μετά τα αθροίσματα). Αυτό το είδος λάθους σε κώδικα ονομάζεται λογικό και δεν εντοπίζεται από τον διερμηνευτή της γλώσσας αλλά εντοπίζεται μόνο κατά την εκτέλεσή του.

2. **: δυνάμεις, π.χ. το $2^{\nu+1}$ θα γραφεί ως `2**(ν+1)` και όχι `2**ν+1` (γιατί;). Μπορεί να χρησιμοποιηθεί και για ρίζες, π.χ. το $\sqrt{25}$ μπορεί να δοθεί ως `25**(1/2)` και το $\sqrt[3]{125}$ ως `125**(1/3)`
3. *, /, //, %: έχουν την ίδια προτεραιότητα μεταξύ τους. Αν δεν υπάρχουν παρενθέσεις γίνονται οι πράξεις όπως και στα μαθηματικά: από αριστερά προς τα δεξιά. Π.χ. `6/3*4` κάνει 8 και αντιπροσωπεύει το $\frac{6}{3}4$, ενώ το `6/(3*4)` κάνει 0.5 από το $\frac{6}{3\times 4}$.
4. +, -: και αυτή η ομάδα έχει την ίδια ιεραρχία, οπότε αν δεν υπάρχουν παρενθέσεις οι πράξεις γίνονται από αριστερά προς τα δεξιά. Π.χ. `2-3+4*5` επιστρέφει 19.

Ακολουθούν μερικά παραδείγματα υπολογισμών στην κονσόλα της python

ΛΙΣΤΑ ΚΩΔΙΚΑ 2-1 | ΑΡΙΘΜΗΤΙΚΕΣ ΠΡΑΞΕΙΣ

```
01. >>> 1+2.0      #γίνεται μετατροπή του αποτελέσματος σε πραγματικό αριθμό
02. 3.0
03. >>> 1-3
04. -2
05. >>> 3/4*5
06. 3.75
07. >>> 6/2*3      #προσοχή στην προτεραιότητα των πράξεων
08. 9.0
09. >>> 2/2        #η διαίρεση ακεραίων αποδίδει πραγματικό πηλίκο(!)[2]
10. 1.0
11. >>> 6/2
12. 3.0
13. >>> (1+2+3)/3  #ο μέσος όρος των 1, 2 και 3
14. 2.0
15. >>> 1+2+3/3
16. 4.0
17. >>> 2**16      #το 2^16
18. 65536
19. >>> 7/3
20. 2.3333333333333335
21. >>> 7//3
22. 2
```

2. Όταν τα ορίσματα μιας διαίρεσης είναι ακέραιοι η python κάνει τη λεγόμενη *πραγματική διαίρεση* (*true division*) γιατί προσπαθεί να αναπαραστήσει το πηλίκο με τη μεγαλύτερη δυνατή ακρίβεια.

```
>>> 7%3
1
>>> 6/3          #η πραγματική διαίρεση αποδίδει πραγματικό αριθμό
2.0
>>> -7//3        #Ιδιαίτερη περίπτωση είναι οι αρνητικοί ακέραιοι
-3
>>> -7%3
2
>>> 6%3
0
>>> 1/10
0.1            #ο υπολογισμός φαίνεται σωστός, αλλά...
>>> 0.1+0.1+0.1-0.3     #ενώ θα έπρεπε να δίνει 0, τελικά δίνει
5.551115123125783e-17 #έναν πολύ μικρό αριθμό αλλά όχι 0![3]
```

Ιδιαίτερη αναφορά πρέπει να γίνει στις πράξεις της ακέραιας διαίρεσης (//, % - div, mod) γιατί, ενώ για θετικούς ακέραιους το αποτέλεσμα είναι το αναμενόμενο, γίνεται δηλαδή αποκοπή δεκαδικών και στρογγυλοποίηση προς το μηδέν (όπως στη C), για τους αρνητικούς αριθμούς γίνεται *διαίρεση πατώματος* (*floor division*), δηλαδή αποκοπή προς το μείον άπειρο και όχι προς το μηδέν. Στον Πίνακα 2-2 φαίνονται όλες οι περιπτώσεις και γίνεται σύγκριση με τα αντίστοιχα αποτελέσματα σε C.

ΠΙΝΑΚΑΣ 2-2 | ΑΚΕΡΑΙΑ ΔΙΑΙΡΕΣΗ

	Πράξεις	Συνδυασμοί τιμών			
	a	7	-7	7	-7
	b	3	3	-3	-3
Python	a//b	2	-3	-3	2
	a%b	1	2	-2	-1
C	a/b	2	-2	-2	2
	a%b	1	-1	1	-1

Η πράξη `7//(-3)` επιστρέφει `-3` ώστε με τη διαίρεση πατώματος να ικανοποιείται και για τους αρνητικούς αριθμούς η παρακάτω σχέση (Ταυτότητα Ευκλείδειας Διαίρεσης):

Διαιρέτης·Πηλίκο+Υπόλοιπο = Διαιρετέος
ή, για τα δεδομένα του πίνακα 2-2

```
>>>b* (a//b)+a%b==a
True
```

3. Αυτό οφείλεται στην περιορισμένη ακρίβεια που έχει η φυσική αναπαράσταση των πραγματικών αριθμών στη μνήμη του υπολογιστή. Περισσότερα σε επόμενη ενότητα (το πρόβλημα του 1/10).

Ο ίδιος ο Guido van Rossum δίνει την εξήγηση για τους λόγους που υιοθέτησε αυτή τη λογική στην ακέραια διαίρεση της γλώσσας, στο blog του για την «Ιστορία της Python»[4].

1.3.2 Δυαδικοί τελεστές

Σε πολλές περιπτώσεις είναι αναγκαίο να γίνει χειρισμός σε επίπεδο δυαδικών τιμών. Αυτό γίνεται με τη χρήση των δυαδικών τελεστών. Στα παραδείγματα του Πίνακα 2-3 δίνονται οι αριθμοί σε δεκαδική και δυαδική μορφή.

ΠΙΝΑΚΑΣ 2-3 | ΔΥΑΔΙΚΟΙ ΤΕΛΕΣΤΕΣ (BITWISE OPERATORS)

Τελεστής	Λειτουργία	Παράδειγμα	Αποτέλεσμα
& ΚΑΙ(AND)	Σύζευξη Αντιγραφή ενός bit αν υπάρχει και στους δύο αριθμούς	`58 & 38` `111010 & 100110`	`34` `100010`
| Ή(OR)	Διάζευξη Αντιγραφή ενός bit αν υπάρχει σε έναν από τους δύο αριθμούς	`58 | 38` `111010 | 100110`	`62` `111110`
^ XOR	Αποκλειστική διάζευξη (Exclusive OR) Αντιγραφή ενός bit αν υπάρχει σε έναν από τους δύο αριθμούς και όχι και στους δύο	`58 ^ 38` `111010 ^ 100110`	`28` `011100`
~	Αντιστροφή Αντιστροφή των bit ενός αριθμού. `~A` ισοδυναμεί με `– (A + 1)`	`15` `~1111`	`-16` `-10000`

<<	Ολίσθηση αριστερά
	`A<<n`: Ολίσθηση των bit του `A` προς τα αριστερά κατά `n` θέσεις.
	Αν `n=1` διπλασιάζει τον αριθμό. Το 2ο παράδειγμα είναι ισοδύναμο με το 2^5
		`17<<1`
		`100001`
		
		`2<<4`
		`10`
			`34`
			`100010`
			
			`32`
			`1000000`

>>	Ολίσθηση δεξιά `A>>n`: Ολίσθηση των bit του `A` προς τα δεξιά κατά `n` θέσεις. Αν `n=1` υπολογίζει την ακέραια διαίρεση του `A` με το `2 (A//2)`	`17>>1` `100001`	`8` `010000`

4. http://bit.ly/2oAGywj. Ενδιαφέρον παρουσιάζουν τα σχόλια αναγνωστών, μεταξύ άλλων και του Tim Peters, δημιουργού ενός υβριδικού αλγόριθμου ταξινόμησης με *συγχώνευση* (*timsort*) που χρησιμοποιείται εσωτερικά (για ταξινόμηση λίστας - πίνακα) εξ ορισμού στην Python και στην Java.

Ακολουθούν μερικά παραδείγματα στην κονσόλα της python

ΛΙΣΤΑ ΚΩΔΙΚΑ 2-2 | ΔΥΑΔΙΚΟΙ ΤΕΛΕΣΤΕΣ

```
01. >>> 58 & 38  #ενώ η σύζευξη γίνεται με ακέραιους αριθμούς
02. 34           #το αποτέλεσμα προκύπτει από τις δυαδικές αναπαραστάσεις
03. >>> bin(58),bin(38), bin(58&38)   # ή bin(34)
04. ('0b111010', '0b100110', '0b100010')

        #η πιο κάτω εντολή κάνει χρήση της συνάρτησης bin()για τη
        #δυαδική αναπαράσταση των δεκαδικών 58 και 38

In [1]: 58 & 38, bin(58),bin(38), bin(58&38)

        #και επιστρέφει μια πλειάδα τιμών
Out[1]: (34, '0b111010', '0b100110', '0b100010')
```

IP[y]:
IPython

Οι λειτουργίες που εκτελούν οι δυαδικοί τελεστές γίνονται σε ακέραιους μόνο αριθμούς. Όταν χρησιμοποιείται ένας δυαδικός αριθμός, ο χειρισμός των αριθμών – τελεστέων που τους συνοδεύουν γίνεται σε επίπεδο δυαδικών ψηφίων. Έτσι, προκειμένου να χρησιμοποιηθούν σωστά, θα πρέπει οι αριθμοί στους οποίους θα λειτουργήσουν να είναι ακέραιοι και σε δυαδική μορφή. Μπορεί επομένως να χρησιμοποιηθούν με δεκαδική αναπαράσταση, αλλά η λειτουργία τους, πάντα, γίνεται στον αντίστοιχο αριθμό στο δυαδικό σύστημα αρίθμησης. Υπάρχουν τομείς της επιστήμης των υπολογιστών όπου είναι απαραίτητη η επεξεργασία δεδομένων στη δυαδική αναπαράστασή τους, όπως για παράδειγμα η επεξεργασία πακέτων δεδομένων και η δρομολόγηση δεδομένων σε δίκτυα Η/Υ, στοιχεία γενικά συστημάτων επικοινωνίας, ψηφιακά κυκλώματα, σύνδεση, επικοινωνία και χειρισμός περιφερειακών συσκευών σε υπολογιστικά συστήματα, θέματα υλικού, γενικότερα, προβλήματα ασφάλειας, θέματα κρυπτογράφησης, συμπίεση αρχείων και αλγόριθμοι κωδικοποίησης εικόνας, ήχου ή βίντεο (image, sound, video processing).

1.3.3 Μοναδιαίοι τελεστές

Χρησιμοποιούνται για την αντιστροφή του πρόσημου ενός αριθμού. Σε ορισμένες βιβλιοθήκες κώδικα όπως η `Collections`, χρησιμεύει και στον χειρισμό συνόλων.

ΠΙΝΑΚΑΣ 2-4 | ΜΟΝΑΔΙΑΙΟΙ ΤΕΛΕΣΤΕΣ

Τελεστής	Λειτουργία	Παράδειγμα	Αποτέλεσμα
+	Θετική σήμανση	+12	12
-	Αντιστροφή ή αρνητική σήμανση	--12 -12	12 -12

1.4 Η έννοια της μεταβλητής

Αν όχι το πιο δύσκολο, τότε σίγουρα ένα από τα πιο συχνά προβλήματα που προκαλούν σύγχυση σε όποιον ασχολείται για πρώτη φορά με τον προγραμματισμό είναι η έννοια της μεταβλητής. Οι ομοιότητές της, αλλά κυρίως η διαφορετική φύση και χρήση της στον προγραμματισμό απ' ό,τι στον χώρο της Επιστήμης των Μαθηματικών, κάνουν τη μεταβλητή το πρώτο, πολλές φορές ανυπέρβλητο, εμπόδιο στην εκμάθηση της πρώτης γλώσσας προγραμματισμού. Η σαφής διάκριση της χρήσης της και η φιλοσοφία που «κρύβεται» πίσω από αυτήν για την κατασκευή κώδικα είναι στοιχεία που θα πρέπει από πάρα πολύ νωρίς να αποσαφηνιστούν, για να μπορέσει κάποιος να μυηθεί στον χώρο του προγραμματισμού.

Από τις πρώτες τάξεις της Μέσης Εκπαίδευσης, γίνεται γνωστό στη βασική Άλγεβρα τι είναι εξίσωση, πώς λύνονται τέτοιου είδους πραγματικά προβλήματα και πώς η επίλυσή τους ισοδυναμεί με την εύρεση μιας αριθμητικής τιμής – λύσης που θα ικανοποιεί τα δεδομένα του προβλήματος. Κύριο συστατικό αυτής της διαδικασίας είναι η μεταβλητή – ο άγνωστος, συνήθως, x – που θα αποκτήσει μια τιμή, μια λύση. Η μεταβλητή στον χώρο του προγραμματισμού αποκτά μια διαφορετική σημασία που, ενώ είναι και εδώ μια οντότητα που θα αποκτήσει κάποια τιμή (όχι πάντα αριθμητική), θα έχει όμως και μια «φυσική» υπόσταση. Με απλά λόγια:

> *Μια μεταβλητή είναι μια (αναφορά σε μια) θέση μνήμης, όπου αποθηκεύεται, ένα κάθε φορά στοιχείο ή δεδομένο.*

Αυτό το περιεχόμενο είναι δυνατόν να αλλάζει (να μεταβάλλεται), όποτε αυτό απαιτείται από τις ανάγκες του προγράμματος.

Η επόμενη μεγάλη δυσκολία έρχεται στη χρήση της μεταβλητής μέσα στον κώδικα. Η μεταβλητή καλείται να ενσωματώσει μέσα της όλα τα πιθανά δεδομένα, που κατά τον χρόνο συγγραφής του προγράμματος, δεν είναι εκ των προτέρων γνωστά στον προγραμματιστή. Είναι, τις περισσότερες φορές, άγνωστο το τι πραγματικά θα λάβει μια μεταβλητή από τον τελικό χρήστη (ή χειριστή) του προγράμματος όταν αυτό θα «τρέξει». Αυτή η αφηρημένη χρήση της, ως προς το περιεχόμενό της, είναι που θα κάνει ένα πρόγραμμα να εκτελείται όχι μόνο για μια συγκεκριμένη περίπτωση αλλά για κάθε περίπτωση του ίδιου προβλήματος.

Ένα από τα πιο απλά, εισαγωγικά, υπολογιστικά προβλήματα είναι αυτό της εύρεσης του εμβαδού ενός γεωμετρικού μεγέθους, όπως π.χ. του τετραγώνου. Το πρόγραμμα που θα κατασκευαστεί για αυτό τον υπολογισμό θα πρέπει να λειτουργεί και να υπολογίζει το αντίστοιχο εμβαδό για οποιοδήποτε τετράγωνο, με οποιοδήποτε μήκος πλευράς. Κάτι τέτοιο είναι αδύνατο χωρίς τη χρήση μεταβλητής.

Στην python (όπως και σε άλλες γλώσσες) στην πραγματικότητα μια μεταβλητή είναι απλά μια αναφορά (ή ένας *δείκτης – pointer* στη C) στη διεύθυνση της μνήμης του υπολογιστή όπου πραγματικά είναι αποθηκευμένο ένα στοιχείο, μια τιμή.

1.5 Ιδιότητες μεταβλητών

Μια μεταβλητή έχει τα εξής χαρακτηριστικά:

1.5.1 Όνομα μεταβλητής

Η ονομασία (ή ονοματοδοσία) των μεταβλητών ακολουθεί κάποιους απλούς αλλά ρητούς κανόνες.

Οι χαρακτήρες που μπορούν να χρησιμοποιηθούν για την ονομασία μιας μεταβλητής είναι:

i. Όλο το αγγλικό αλφάβητο, πεζά και κεφαλαία.
ii. Όλοι οι αριθμοί από το 0 μέχρι το 9.
iii. Ο χαρακτήρας "_" (*κάτω παύλα* ή *υπογράμμιση – underscore*). Η ιδιαίτερη σημασία όμως που δίνεται σε αυτόν τον χαρακτήρα και η χρήση του σε συγκεκριμένες περιπτώσεις κώδικα, σύμφωνα με τα πρότυπα της γλώσσας, υπαγορεύει τη χρήση του σε οποιοδήποτε σημείο του ονόματος αλλά όχι στην αρχή του. Στο Κεφάλαιο 11 (αντικείμενα) περιγράφεται η ειδική χρήση του χαρακτήρα υπογράμμισης σε ονομασίες μεθόδων κλάσεων.
iv. Από την έκδοση 3 της γλώσσας και μετά υποστηρίζεται κωδικοποίηση και στο σύστημα Unicode και είναι δυνατή η ονομασία μεταβλητών και στα ελληνικά. Κάτι τέτοιο όμως θα πρέπει να γίνεται προσεκτικά γιατί μπορεί να οδηγήσει σε λάθη, δυσανάγνωστα προγράμματα που μπορεί να μην τρέχουν σε όλα τα συστήματα.

Απαγορεύεται από την άλλη:

i. Η χρήση ειδικών συμβόλων *, #, %, +, -, / κ.λπ. τα οποία χρησιμεύουν για άλλες λειτουργίες (όπως για παράδειγμα για αριθμητικές πράξεις)
ii. Οι λέξεις του Πίνακα 2-5 οι οποίες θεωρούνται *δεσμευμένες* (*reserved words*) και αποτελούν δομικά στοιχεία της γλώσσας και επιτελούν άλλες λειτουργίες όπως π.χ. η λέξη «while» (αναφέρεται στη δομή επανάληψης, βλ. Κεφάλαιο 4).

ΠΙΝΑΚΑΣ 2-5 | ΔΕΣΜΕΥΜΕΝΕΣ ΛΕΞΕΙΣ

`False`	`class`	`finally`	`is`	`return`	`import`	`pass`
`None`	`continue`	`for`	`lambda`	`try`	`else`	`break`
`True`	`def`	`from`	`nonlocal`	`while`	`assert`	`raise`
`and`	`del`	`global`	`not`	`with`	`except`	
`as`	`elif`	`if`	`or`	`yield`	`in`	

Το όνομα κάθε μεταβλητής θα πρέπει να ξεκινάει με γράμμα (είτε πεζό είτε κεφαλαίο) ή υπογράμμιση αλλά όχι με αριθμό. Στην python (όπως και σε γλώσσες όπως C/C++, Java κ.ά.) γίνεται διάκριση μεταξύ πεζών και κεφαλαίων (*case sensitivity*), κάτι που είναι γνώριμο σε χρήστες λειτουργικών συστημάτων τύπου Linux αλλά

όχι σ' αυτούς των windows. Έτσι, οι ονομασίες `variable` και `Variable` αναφέρονται σε διαφορετικές μεταβλητές. Σύμφωνα με τα παραπάνω, ονομασίες μεταβλητών που είναι ή όχι αποδεκτές φαίνονται στον Πίνακα 2-6.

ΠΙΝΑΚΑΣ 2-6 | ΠΑΡΑΔΕΙΓΜΑΤΑ ΟΝΟΜΑΤΩΝ ΜΕΤΑΒΛΗΤΩΝ

Αποδεκτά ονόματα	Μη αποδεκτά ονόματα	
`Mikos_plevras`	`1x`	το όνομα δεν πρέπει να ξεκινά με αριθμό
`pi`	`first name`	προσοχή στο κενό (space) ανάμεσα στις λέξεις
`x1`	`x-y`	η χρήση του "-" υποδηλώνει αφαίρεση
`client_name`	`a:`	ο χαρακτήρας ":" έχει άλλη λειτουργία (έναρξη
`Net_Value_2017`		μπλοκ εντολών)
`_name`	`date.of.birth`	η χρήση της τελείας γίνεται σε αντικείμενα
`hjkjkdhfg`	`if`	είναι δεσμευμένη λέξη
`NetValue2017`		

Κάθε μη έγκυρο όνομα προκαλεί την αντίδραση του διερμηνευτή της γλώσσας, του εσωτερικού μηχανισμού δηλαδή που προσπαθεί να μετατρέψει τον πηγαίο κώδικα σε δυαδικό κώδικα (γλώσσα) μηχανής, και εμφανίζει στην οθόνη του υπολογιστή ένα μήνυμα σφάλματος, διακόπτοντας την εκτέλεση του προγράμματος. Τέτοιου είδους σφάλματα ονομάζονται *συντακτικά* (*syntax errors*).

Για παράδειγμα, η επόμενη εντολή χρησιμοποιεί τη μεταβλητή a b (με κενό ανάμεσά τους) και προκαλεί τη διακοπή εκτέλεσης του κώδικα με την εμφάνιση αντίστοιχου μηνύματος.

```
>>> a b=5 enter
  File "<ipython-input-1-a220ed9b53b0>", line 1
    a b=5
      ^
SyntaxError: invalid syntax
```

Υπάρχουν επίσης κάποιες προτεινόμενες πρακτικές που είναι καλό να ακολουθούνται αναφορικά με την ονομασία μεταβλητών. Αυτές έχουν να κάνουν με τη χρήση ή τον σκοπό των μεταβλητών. Τέτοιοι κανόνες ορίζονται στα πρότυπα της γλώσσας (PEPs – Python Enhancements Proposals και συγκεκριμένα στο PEP8[5]). Σκοπός τους είναι να βοηθήσουν στη συγγραφή προγραμμάτων με κοινή μορφή, ευανάγνωστων και εύκολα κατανοητών από άλλους προγραμματιστές.

- Όλα κεφαλαία (με ή χωρίς κάτω παύλα): για τη δημιουργία σταθερών (δηλαδή μεταβλητών που δεν αλλάζουν τιμή), π.χ. `PI, FIXED_COST, FPA`
- Όλα πεζά (με ή χωρίς κάτω παύλα): συνήθεις μεταβλητές ή συναρτήσεις (αναλύονται αργότερα), π.χ. `age, mobile_phone, displaymenu`

5. http://bit.ly/2x34tuE

- Καμηλοειδής μορφή (camel case ή, ορθότερα, πεζοκεφαλαία): ονομασία μεθόδων κλάσεων. Γίνεται χρήση κεφαλαίου γράμματος μόνο στο πρώτο γράμμα κάθε λέξης και ονομάζεται έτσι από τον γνωστό μορφότυπο του συμπαθούς θηλαστικού. Π.χ. `InventoryUpdate, dateOfBirth.`

Μερικά περιβάλλοντα εργασίας (IDEs), όπως το PyCharm και το Spyder (μέσα από τις ρυθμίσεις του), εμφανίζουν ειδοποιήσεις για τη μη τήρηση των κανόνων PEP8. Ωστόσο, καμία από αυτές τις μορφές δεν είναι υποχρεωτικό να χρησιμοποιείται. Αν όμως ο κώδικας που γράφεται γίνεται σε συνεργασία ή δοθεί σε άλλους, αν πρόκειται για επαγγελματική δουλειά ή αν δημοσιευτεί σε κάποια εργασία, στο internet ή σε κάποιο αποθετήριο κώδικα (π.χ. Github), τότε θα πρέπει να ακολουθούνται κοινές πρακτικές. Παρόλα αυτά, όποιος τρόπος και αν χρησιμοποιηθεί θα πρέπει να ακολουθείται σε ολόκληρο το πρόγραμμα ο ίδιος.

Τέλος, προτιμάται η χρήση απλών, μικρών ή και σύνθετων λέξεων αντί σκέτων γραμμάτων, που θα υποδηλώνουν το περιεχόμενο μιας μεταβλητής όπως π.χ. `mikos_plevras` αντί για ένα απλό `a`, για το μήκος μιας πλευράς ενός τετραγώνου, ή `afm` για το ΑΦΜ ενός πελάτη αντί για ένα σκέτο `x`.

1.5.2 Το περιεχόμενο μιας μεταβλητής

Μια μεταβλητή στην python ξεκινάει να υπάρχει όταν θα της δοθεί μια τιμή. Αυτή η τιμή κάποια στιγμή μπορεί να αλλάξει με κάποια άλλη ή απλά να διαγραφεί από τη μνήμη εντελώς, οπότε διαγράφεται και η μεταβλητή.

Διαφέρει επομένως, το όνομα μιας μεταβλητής από το περιεχόμενό της, την τιμή της.

Κατά τη διάρκεια εκτέλεσης ενός προγράμματος το όνομα μιας μεταβλητής δεν αλλάζει, σε αντίθεση με το περιεχόμενο, την τιμή και τον τύπο της (το είδος δεδομένων που περιέχει). Και αυτό γιατί η python, σε αντίθεση με άλλες γλώσσες, όπως η C/C++, Java κ.ά., είναι μια δυναμική γλώσσα στη χρήση μεταβλητών.

Αυτό πρακτικά σημαίνει πως μια μεταβλητή μπορεί αρχικά να περιέχει έναν αριθμό αλλά στη συνέχεια να περιέχει κείμενο, μια λίστα ή οτιδήποτε άλλο. Κάτι τέτοιο βέβαια δεν αποτελεί μια συνήθη ή καλή προγραμματιστική πρακτική.

Τι όμως μπορεί να περιέχει μια μεταβλητή; Το περιεχόμενο ή ο τύπος δεδομένων μιας μεταβλητής μπορεί να είναι απλές αριθμητικές τιμές (5, 3.14, -2323234) και χαρακτήρες (κείμενο όπως "Εγνατίας 147" – πάντα μέσα σε εισαγωγικά), μέχρι πιο σύνθετες δομές όπως λίστες και λεξικά που περιγράφονται στα Κεφάλαια 5 και 9. Στις επόμενες παραγράφους περιγράφονται οι τρεις αριθμητικές τιμές που υποστηρίζει η python: ακέραιοι (integers αλλά και λογικές τιμές τύπου ακεραίου), πραγματικοί (κινητής υποδιαστολής – floating point), μιγαδικοί (complex) αριθμοί αλλά και λογικές τιμές (Boolean: True – False, ως απλό υποσύνολο των ακέραιων αριθμών). Ο Πίνακας 2-7 περιλαμβάνει τους βασικούς τύπους δεδομένων που υποστηρίζει η python.

ΠΙΝΑΚΑΣ 2-7 | ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Τύποι	Παραδείγματα
Ακέραιοι	`3, -12, 2**300` που δίνει έναν πολύ μεγάλο ακέραιο: `203703597633448608626844568840937816105146839366593625063614044935438129976333670618339737 6`
Πραγματικοί	`3.14, 100.0`
Μιγαδικοί	`12+2j`
Λογικές τιμές	
True (ή 1)	
False (ή 0)	
Χαρακτήρες	`'Κείμενο', "Άννα", "Εγνατίας 12", "...είπαν το μεγάλο 'ΟΧΙ' !"`
Λίστες	`[1, 3.14, "a"] , [1, 2, [2, 1] ]`
Σύνολα	`{1, 2, 3, "Πέτρος", 3.14}`
Πλειάδες	`(1, 2, "a")`
Λεξικά	`{1: 'μήλα', 2: 'μπανάνες'}`

Σε αντίθεση με άλλες γλώσσες προγραμματισμού (όπως οι C/C++ και Java), που θεωρούνται *αυστηρές στην πληκτρολόγηση* (*strictly – strongly typed*), στην python δεν απαιτείται (στην ουσία δεν είναι δυνατόν) να δηλωθεί από την αρχή ο τύπος της. Έτσι, η εντολή

```
>>>a=10
```

δημιουργεί μια νέα μεταβλητή με όνομα `a` και (ακέραια) τιμή τον αριθμό `10`. Λίγο πιο μετά μπορεί να ακολουθήσει μια εντολή όπως η

```
>>>a=3.14
```

όπου η python «ξεφορτώνεται» την παλιά τιμή (το `10`) και αποθηκεύει στη μεταβλητή `a` τον (πραγματικό αυτή τη φορά) αριθμό `3.14`.

Σε φυσικό επίπεδο, μια μεταβλητή δεν είναι παρά μόνο ένα *όνομα* (*name – label*) στον *χώρο ονομάτων* (*namespace*) που διαχειρίζεται η κονσόλα της εκάστοτε υλοποίησης της γλώσσας. Αυτό το όνομα συνοδεύεται από μια διεύθυνση μνήμης όπου βρίσκεται η τιμή της. Σε ξεχωριστό χώρο της φυσικής μνήμης του υπολογιστή, φυλάσσεται η τιμή της μεταβλητής και ο τύπος της. Πιο συγκεκριμένα, επειδή στην python όλα είναι αντικείμενα, αποθηκεύεται ο τύπος του αντικειμένου στο οποίο «δείχνει» η μεταβλητή. Έτσι, ο διερμηνευτής της γλώσσας δεν χρειάζεται να μετατρέψει ή να αποκρυπτογραφεί τιμές κάθε φορά που θα συναντά το όνομα της μεταβλητής, όλες οι πληροφορίες βρίσκονται αποθηκευμένες μαζί με την τιμή της μεταβλητής.