

Εισαγωγή στην Επιστημονική χρήση Υπολογιστών

ΦΥΣ 140

Φθινόπωρο 2023

Διδάσκοντες: **Φώτης Πτωχός**
Αλέξανδρος Αττίκης

e-mail: ptochos.fotios@ucy.ac.cy, attikis.x.alexandros@ucy.ac.cy

Τηλ: 22.89.2837/2834

Γραφείο: B235 & B232– ΘΕΕ02

Γενικές Πληροφορίες

➡ Διαλέξεις ⬅

Δευτέρα: 08:30 – 10:30

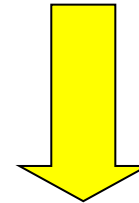
Αίθουσα: ΧΩΔ02 - 008

➡ Εργαστήρια ⬅

Πέμπτη 09:00 – 11:00 (1^ο group)

Παρασκευή 15:00 – 17:00 (2^ο group)

ΥΠΟΧΡΕΩΤΙΚΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗ



Πάνω από μια απουσία στα εργαστήρια
χωρίς σημαντικό λόγο ισοδυναμεί με
αυτόματη αποτυχία στο μάθημα

➡ Φροντιστήρια ⬅

Τρίτη 15:00 – 17:00

Αίθουσα: ΧΩΔ02 – B107

Απορίες

- ❑ Διακόπτεται για απορίες κατά την διάρκεια των διαλέξεων.
- ❑ **Περάστε από τα γραφεία μας:** Τρίτη & Παρασκευή πρωί
Ωστόσο μπορείτε να έρθετε οποιαδήποτε ώρα δείτε φως...
- ❑ Πείτε μου αν κάτι στο μάθημα δεν δουλεύει για σας και πως μπορεί να αλλάξει ώστε να σας βοηθήσει περισσότερο
- ❑ Δύο μεταπτυχιακοί συνάδελφοί σας σε κάθε εργαστήριο θα είναι παρόντες μαζί με μας και θα βοηθήσουν σε οποιαδήποτε προβλήματα

Βιβλιογραφία

Δεν θα ακολουθήσω κάποιο συγκεκριμένο βιβλίο.

Οι διαλέξεις/σημειώσεις θα είναι στο web.

Οι ασκήσεις/οδηγίες των εργαστηρίων όπως και οι **εργασίες** που θα δουλεύετε σπίτι θα τις βρίσκετε επίσης στο web.

(<https://ptohos.github.io/phy140/phy140.html>)

Χρήσιμη βιβλιογραφία/παραδείγματα μπορούν να βρεθούν στο web. Προσπαθήστε και μόνοι σας τις πρώτες δύο βδομάδες να ασχοληθείτε και να εξοικειωθείτε με

Λειτουργικό σύστημα: **Linux**

Γλώσσα προγραμματισμού: **Python v3**

Λογισμικό Γραφικών αναπαραστάσεων: **μέσα στην Python**

Όλοι έχετε accounts στο τμήμα της Φυσικής και μπορείτε να χρησιμοποιείται τους υπολογιστές του τμήματος για πρακτική εξάσκηση πέρα από τις ώρες εργαστηρίου.

➤ Στην ιστοσελίδα του μαθήματος ακολουθώντας το button [Ανακοινώσεις](#) θα βρείτε οδηγίες ώστε να εγκαταστήσετε το λειτουργικό σύστημα Linux σε περιβάλλον Windows με τη χρήση virtual box (VB).

Βιβλιογραφία

- Για Linux θα μπορούσατε να χρησιμοποιήσετε τα ακόλουθα:
 - [Linux in a nutshell](#) από Ellen Siever, Stephen Spainhour, Stephen Figgins and Jessica P. Hekman , ed. O'Reilly
 - [Running Linux](#) από Matt Welsh, Matthias Kalle Dalheimer, and Lar Kaufman, ed. O'Reilly
- Πολλά μπορούν να βρεθούν επίσης χρησιμοποιώντας την μηχανή αναζήτησης στο web [google](#)
- Για το κύριο μέρος του μαθήματος: **Python και Φυσική**
 - **Programming in Python 3: “A complete introduction to the python language”** από O'Reilly .
 - **Learning Python** by David Ascher and Mark Lutz
 - **Python Crash course** από E. Matthes
 - **Learning Scientific programming with Python**, C. Hill, Cambridge University Press
 - **A student's guide to python for physical modeling**, από J. Kinder και P. Nelson, Princeton University Press
 - **Essential Python for Physicists**, από G. Moruzzi, Springer

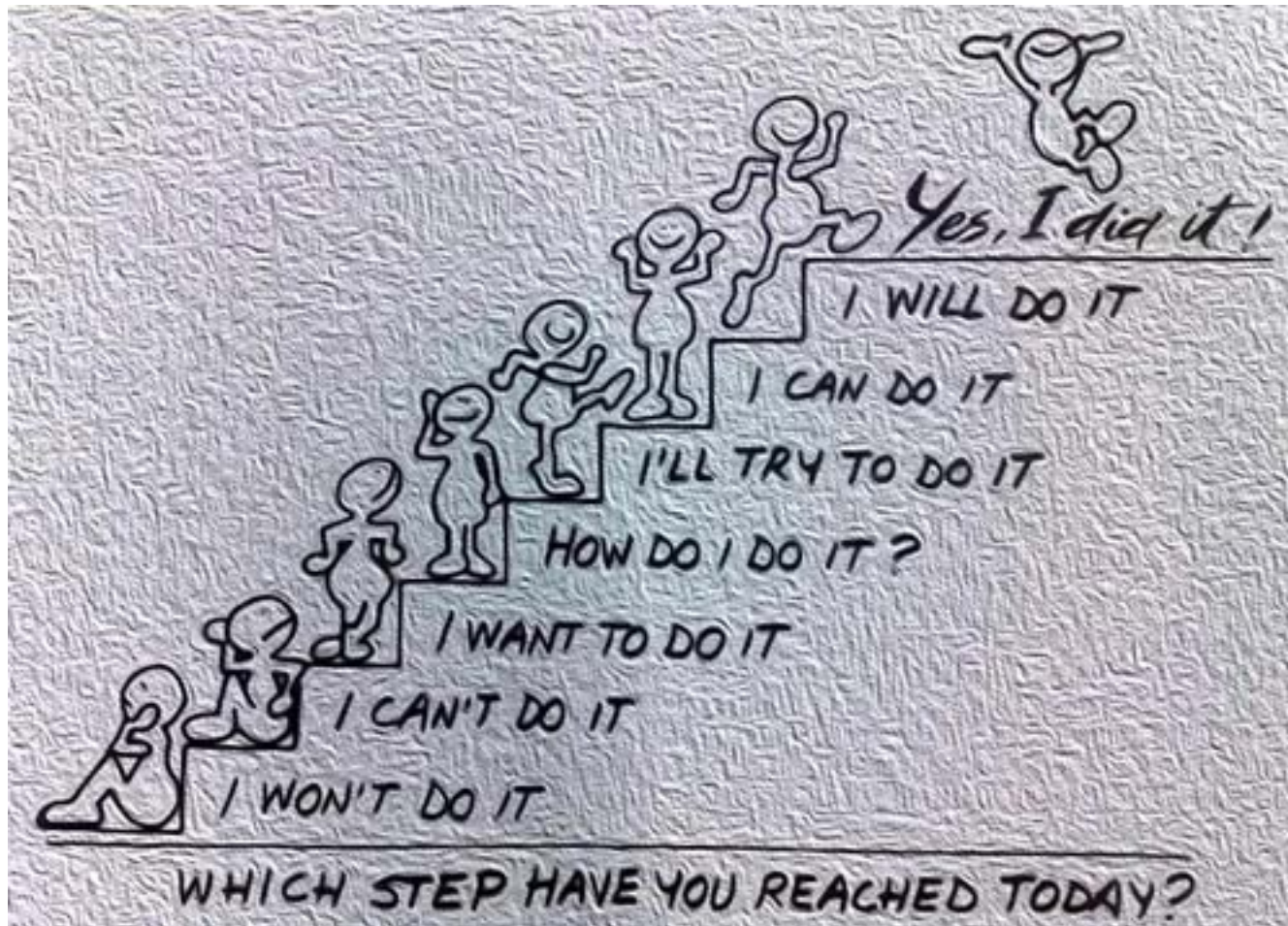
Βαθμολογία

- Η βαθμολογία θα βασιστεί στα ακόλουθα:
 - **10%** εβδομαδιαία quizzes
 - **15%** εργασίες στο σπίτι
 - **30%** 1 εξέταση προόδου (Τετάρτη 18 Οκτώβρη,
10:00 – 12:30 group1,
13:00 – 15:30 group2)
 - **45%** 4-ωρη τελική εξέταση (Δεκέμβρη – σε 2 groups)
 - **Αλλά:** Αν ο βαθμός της τελικής εξέτασης είναι καλύτερος από τον βαθμό της προόδου, η τελική εξέταση μετρά 75%

Εργασίες στο σπίτι

- ☐ Εργασίες για το σπίτι θα περιέχουν 4-5 ασκήσεις.
- ☐ Θα πρέπει να είναι δικιές σας και όχι αντιγραφή λύσεων άλλων συναδέλφων σας. Μπορείτε να συνεργάζεστε αλλά όχι ακριβώς ίδιες.

Καμπύλη μάθησης ...



Περιεχόμενο του Μαθήματος

- Βασικές αρχές προγραμματισμού σε Python, βασικές εντολές λειτουργικού συστήματος, εισαγωγή και βασικές λειτουργίες του [Editor/Emacs](#). Λογισμικό πακέτο γραφικών παραστάσεων δεδομένων.
- Κατασκευή γραφικών παραστάσεων.
- Επίλυση απλών προβλημάτων φυσικής
- Ανάλυση πειραματικών δεδομένων.
- Χρήση τυχαίων αριθμών για προσομοίωση διαφόρων φαινομένων

Βασικές εντολές PYTHON

❑ PYTHON:

- Ξεκίνησε σαν γλώσσα προγραμματισμού στις αρχές της δεκαετίας του 90. Βρήκε ευρεία εφαρμογή εξαιτίας της απλότητάς της. Τα τελευταία 10 χρόνια χρησιμοποιείται η έκδοση Python v3
- Η γλώσσα Python έγινε ιδιαίτερα δημοφιλής ως γλώσσα στις αρχές της δεκαετίας 2000 και αποτελεί την πιο δημοφιλή γλώσσα.
- Τα χρόνια της παρουσίας της διάφοροι χρήστες έγραψαν βιβλιοθήκες από χρήσιμα προγράμματα που χρησιμοποιούνται από χιλιάδες άλλους χρήστες κατά τρόπο παρόμοιο με αυτό του δανεισμού βιβλίων από βιβλιοθήκες.
- Η γλώσσα χρησιμοποιείται σε μια σειρά εφαρμογών: Ανάπτυξη internet και web, επιστημονική χρήση

Πρόγραμμα

- ❑ Πρόγραμμα είναι μια σειρά από οδηγίες (εντολές προγράμματος) συνταγμένες με τέτοιο τρόπο ώστε να επιτρέψει έναν ηλεκτρονικό υπολογιστή να λύσει κάποιο πρόβλημα. Το πρόβλημα προς λύση είναι σπασμένο σε πολύ μικρότερα κομμάτια. Τα κομμάτια αυτά θα πρέπει να σχηματίζουν μια καλώς ορισμένη δομή, όπου το πιο περίπλοκο και μεγάλο πρόβλημα στην κορυφή και το πιο απλό και εύκολα επιλύσιμο στη βάση.

➤ Από κει και ο ορισμός: top down programming:
προγραμματισμός από πάνω προς τα κάτω

- ❑ Για να πετύχετε τη λύση ενός προβλήματος με την χρήση μιας γλώσσας προγραμματισμού καλό είναι να ακολουθήσετε τα παρακάτω βήματα:
 - ❑ Εκφράστε το πρόβλημα στα Αγγλικά
 - ❑ Εξετάστε το πρόβλημα και σπάστε το σε μικρότερα προβλήματα
 - ❑ Εξετάστε τα επιμέρους τμήματα και επεξεργαστείτε τα σε μικρότερα
 - ❑ Σχεδιάστε γραφικά το σχέδιο δράσης
 - ❑ Δοκιμάστε το πρόγραμμα

Σχέση γλώσσας προγραμματισμού, προγράμματος, Η/Υ

Προγραμματιστής	Γλώσσα	Πρόγραμμα	Υπολογιστής
Chef	Ελληνικά	Κρέπες	Μάγειρας, σκεύη
Beethoven	Μουσική	Πιάνο Σονάτα	Πιανίστας, πιάνο
Εσείς	PYTHON	υπολογισμός	Ο υπολογιστής σας

- ❑ Δηλαδή προγραμματισμός είναι η διεργασία του γραψίματος κάποιων εντολών που μπορούν να εκτελεστούν πολλές φορές στο μέλλον
- ❑ Προσέξτε ότι και στα 3 παραδείγματα το πρόγραμμα είναι το μέσο για την μεταφορά εντολών ή οδηγιών μεταξύ του προγραμματιστή (αυτού που ανακαλύπτει τις οδηγίες) και του υπολογιστή (αυτού που εκτελεί τις οδηγίες)
- ❑ Η γλώσσα θέτει τους όρους κάτω από τους οποίους θα σχηματιστεί το πρόγραμμα. Για να γίνει κατανοητό ένα πρόγραμμα από τον υπολογιστή θα πρέπει να γραφτεί σύμφωνα με τους όρους αυτούς της γλώσσας.
- ❑ Το πρώτο στάδιο της διεργασίας μάθησης του προγραμματισμού είναι να μάθετε πρώτα τι μπορεί να κάνει ο υπολογιστής και πως μπορείτε να γράψετε τις οδηγίες ώστε να κάνει αυτά τα πράγματα. Το υπόλοιπο της διεργασίας είναι να μάθετε να γράφετε ωραία προγράμματα.

Χρήση Python ως υπολογιστική

Η python μπορεί να χρησιμοποιηθεί για υπολογισμό αλγεβρικών πράξεων:

Πληκτρολογώντας την εντολή **python3** εισέρχεστε σε περιβάλλον της python

```
bash-3.2$ python3
Python 3.7.6 (default, Dec 22 2019, 01:09:06)
[Clang 11.0.0 (clang-1100.0.33.12)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Μπορούμε να κάνουμε ανάθεση τιμών σε διάφορες μεταβλητές:

```
>>> x = 100
>>> █
```

Και να δούμε την τιμή αυτή χρησιμοποιώντας την εντολή **print()**:

```
>>> print(x)
100
>>> █
```

Μπορούμε να κάνουμε διάφορες αριθμητικές πράξεις:

```
>>> x+10  Πρόσθεση
110
>>> x-10  Αφαίρεση
90
>>> x*10  Πολλαπλασιασμός
1000
>>> x/10  Διάρθρωση
10.0
>>> x**2  Εκθέτης **
10000
>>> x%3   Υπόλοιπο διαίρεσης %
1
>>> x//49 Ακέραιο μέρος διαίρεσης και αποκοπή δεκαδικών ψηφίων //
2
```

Σημασία του $a=a+1$

Για να επικοινωνήσουμε με τον Η/Υ χρειάζεται να χρησιμοποιήσουμε μια γλώσσα κατανοητή στον Η/Υ. Κάποιες από τις εντολές μπορεί να μοιάζουν περίεργες

```
>>>  
>>> a = 100  
>>> a = a+1  
>>> print(a)  
101  
>>> 
```

← Μαθηματικά δεν έχει νόημα

Ωστόσο σημαίνει: (1) Ανάθεσε σε μια μεταβλητή a την τιμή 100 και (2) πρόσθεσε στην μεταβλητή a μία μονάδα και το αποτέλεσμα ανάθεσέ το στην ίδια μεταβλητή

Οι πράξεις στον Η/Υ γίνονται πάντοτε στο δεξί μέλος της εξίσωσης.

Το $=$ σημαίνει ανάθεση του αποτελέσματος του δεξιού μέλους σε μια μεταβλητή που κρατά μια θέση στη μνήμη του υπολογιστή αφού πρώτα διαγράψει ότι υπήρχε στη θέση αυτή της μνήμης

- Το σύμβολο $=$ δίνει οδηγία στη Python να αλλάξει την κατάστασή της.
- **Προσοχή:** $a+1 = a$ δεν αποτελεί μια σωστή εντολή για οποιαδήποτε γλώσσα προγραμματισμού παρόλο που μαθηματικά είναι σωστή

Ανάγκη ελέγχου ισότητας

Πολλές φορές χρειάζεται να εξετάσουμε αν μια μεταβλητή έχει συγκεκριμένη τιμή

Για αποφυγή συγχύσεων μεταξύ ανάθεσης και ελέγχου ισότητας οι περισσότερες γλώσσες χρησιμοποιούν το σύμβολο της **διπλής ισότητας == για έλεγχο ισότητας**

```
>>>
>>> a=1
>>> a==0
False
>>> ■
```

Θα πρέπει να αποφεύγονται μικτές αναθέσεις της μορφής:

```
>>>
>>> b = (a==1)
>>> print(b)
True
>>> ■
```

Έλεγχος αν $a == 1$ και ανάθεση της λογικής πράξης (**true ή false**) στη νέα μεταβλητή b

Ιδιαίτερη προσοχή για τις περιπτώσεις που ξεκινώντας το περιβάλλον Python υπάρχουν εντολές της μορφής $b = a^2 - a$

Είναι μαθηματικά σωστή αλλά δεν έχει γίνει ανάθεση τιμής στην μεταβλητή a

```
>>> b = a**2 - a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>> ■
```



Συστατικά του περιβάλλοντος python

PYTHON: η γλώσσα προγραμματισμού. Πως θα περιγράψουμε έναν αλγόριθμο στον ηλεκτρονικό υπολογιστή

python (ή python3): μια εφαρμογή που επιτρέπει να χρησιμοποιούμε τη γλώσσα άμεσα στο terminal

NumPy: βιβλιοθήκη που δίνει αριθμητικούς πίνακες και μαθηματικές συναρτήσεις

PyPlot: βιβλιοθήκη που δίνει διάφορα εργαλεία για απεικόνιση

SciPy: βιβλιοθήκη που δίνει διάφορα εργαλεία για επιστημονικές υπολογιστικές μεθόδους

Ξεκινώντας την python

python: Πληκτρολογώντας python σε terminal ξεκινάτε το περιβάλλον python και μπορείτε να αρχίσετε να πληκτρολογείτε εντολές


Υπάρχουν περιπτώσεις που η python φαίνεται να μην αντιδρά στις εντολές που δίνονται. Αυτό οφείλεται κυρίως στο γεγονός ότι κάποια (, [, { δεν έχει την αντίστοιχη),], } για να κλείσει και η Python διαβάζει περισσότερες εντολές μέχρι να βρει την),], } που θα κλείνει αυτή που είναι ακόμη ανοικτή

Αν δεν βρίσκεται ποιο block (, [, { είναι ανοικτό μπορείτε να βάλετε <ESC>

Η ανάθεση μιας τιμής σε μεταβλητή δεν εμφανίζεται στην οθόνη. Για να δείτε την τιμή θα πρέπει να δώσετε την εντολή:

`print(variable)` ή `print(variable1, variable2,...,variableN)`

Μπορείτε να κάνετε επίσης πολλαπλές αναθέσεις με μια μόνο εντολή:

```
>>> a =1; b=2; c=3
>>> A, B, C = 5, 10, 15
>>> print(a,b,c)
1 2 3
>>> print(A,B,C)
5 10 15
>>> 
```


Help για τις εντολές

Μπορεί να μην ξέρετε αν υπάρχει κάποια εντολή στην python. Μπορείτε να δείτε αν υπάρχει η εντολή δίνοντας την εντολή:

```
help(round)
```

```
-----
```

```
round(number, ndigits=None)
```

```
    Round a number to a given precision in decimal digits.
```

```
    The return value is an integer if ndigits is omitted or None. Otherwise  
    the return value has the same type as the number. ndigits may be negative.
```

```
(END)
```

Αν πληκτρολογήσετε το γράμμα q βγαίνετε από το help menu

Κύρια συστατικά ενός προγράμματος

Σταθερές

Μεταβλητές

Αριθμητικές πράξεις

Ανάθεση τιμών - ισότητες

Έλεγχος αποτελέσματος - επικοινωνία με το πρόγραμμα

Σταθερές (**constants**)

- ❑ Είναι οι ποσότητες το μέγεθος των οποίων δεν αλλάζει κατά την εκτέλεση ενός προγράμματος.
- ❑ Μπορεί να είναι αριθμητικές ή να περιέχουν χαρακτήρες
- ❑ Οι συνηθέστερες μορφές σταθερών είναι:
 - **INTEGER** (ακέραιο σταθερά)
 - **REAL** (πραγματική σταθερά)
 - **DOUBLE PRECISION** (σταθερά διπλής ακρίβειας)
 - **STRING** (σταθερά χαρακτήρων)
 - **LOGICAL** (λογική σταθερά) (αργότερα)

Σταθερές

❑ Integer (ακέραια σταθερά)

- Αποτελείται από ψηφία 0 έως 9 και πιθανώς ένα + ή – που δηλώνει αν ο αριθμός είναι θετικός ή αρνητικός
- Αν ο αριθμός είναι θετικός τότε το + δεν χρειάζεται
- Αν ο αριθμός είναι αρνητικός τότε το – πρέπει υποχρεωτικά να γραφεί μπροστά από τα ψηφία

Παραδείγματα ακεραίων σταθερών είναι:

3 -7 1803 +68 0 1000 -0 +0 -999




Σταθερές

□ Float (πραγματικές σταθερές κινητής υποδιαστολής)

- Ένας πραγματικός αποτελείται από ψηφία (0 έως και 9),
 μια τελεία (υποχρεωτικά) και πιθανώς το πρόσημο + ή –
- Αν ο αριθμός είναι θετικός το + μπορεί να παραληφθεί
- Αν ο αριθμός είναι αρνητικός τότε πρέπει οπωσδήποτε να γράψουμε το – μπροστά από τα ψηφία του.

π.χ. 2.3 -5.75 .302 -7. 0. +23. 0.0 -0.9999

❖ Μια σταθερά float γράφεται και με τη μορφή: $r \times E i$  $r \times 10^i$

π.χ. 2.3 γράφεται και σαν 2.3E00 ή 23.E-1 ή 2300.E-3

-.00854 γράφεται και σαν -8.54E-3 ή -854.E-5

10^{-12} γράφεται και σαν 1.E-12 ή .1E-11

Σταθερές

□ Strings (σταθερές χαρακτήρων)

- Οι σταθερές αυτές αποτελούνται από μια γραμματοσειρά η οποία πρέπει πάντοτε να εσωκλείεται μεταξύ αποστρόφων

“CYPRUS”

Σταθερές

❑ Complex (μιγαδικές)

- Οι σταθερές αυτές χρησιμοποιούνται με την μορφή

π.χ. $1 + 2j$

- σημαίνει το πραγματικό μέρος είναι 1 και το μιγαδικό 2



Προσοχή: είναι διαφορετικό από $1 + 2*j$ όπου το σύμβολο * δηλώνει πολλαπλασιασμό

bits and bytes

- ❑ Ο υπολογιστής χρησιμοποιεί τη **κύρια μνήμη** για αποθήκευση δεδομένων
- ❑ Η μνήμη χωρίζεται σε **words** και κάθε word περιέχει τμήμα πληροφορίας
- ❑ Ο αριθμός των words σε μια κεντρική μνήμη εξαρτάται από τον Η/Υ (αρκετές χιλιάδες για ένα μικρο-processor σε εκατοντάδες εκατομμύρια για μεγάλους υπολογιστές)
- ❑ Κάθε word αποτελείται από μικρότερες μονάδες που ονομάζονται **bits**

Το bit μπορεί να έχει μόνο μια από δύο τιμές: **0 ή 1**

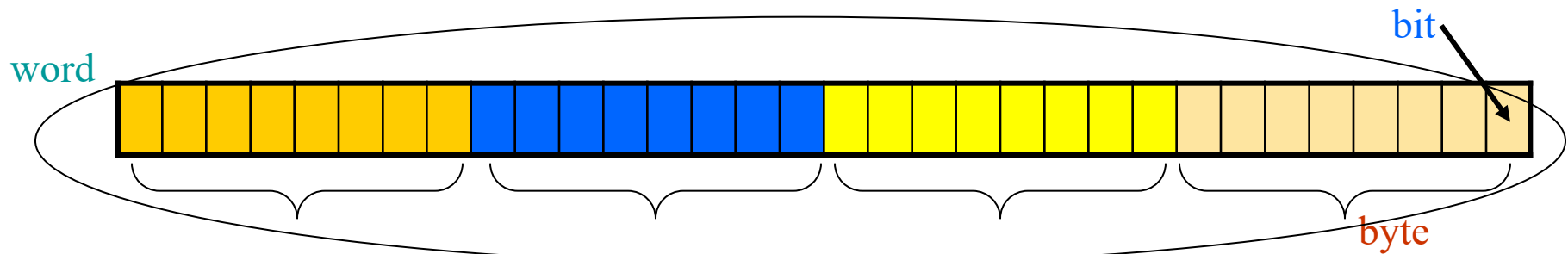
Αν το word περιέχει m bits τότε το word μπορεί να είναι σε οποιαδήποτε από 2^m διαφορετικές καταστάσεις

Η πραγματική σημασία που δίνεται σε κάποιο ιδιαίτερο συνδυασμό από 0 και 1 σε ένα word εξαρτάται από το μοντέλο του υπολογιστή

- ❑ Όλοι οι υπολογιστές (**σχεδόν**) αποθηκεύουν τους θετικούς INTEGERS με τέτοιο τρόπο ώστε κάθε bit που παίρνει τη τιμή 1 στο word αντιπροσωπεύει μια τιμή που είναι $2^{(n-1)}$ όπου n η σχετική θέση του bit στο word μετρώντας από δεξιά.

bits and bytes

- ❑ Οι περισσότεροι υπολογιστές έχουν μνήμη με 32 bits/word με ενδιάμεσο διαχωρισμό του word σε 4 ομάδες από 8 συνεχή bits
- ❑ Ένα τέτοιο γκρουπ αποτελούμενο από 8 bits ονομάζεται **byte**
- ❑ Ένα byte έχει 2^8 ή 256 διαφορετικές καταστάσεις.
Όλοι οι αριθμοί αντιπροσωπεύονται σε δυαδική μορφή



- ❑ Υπάρχει περιορισμένη ακρίβεια και επομένως προσεγγίσεις
- ❑ Το μήκος του word είναι συνήθως 32 bits ή 4 bytes
όπου κάθε byte αποτελείται από 8 bits

Μονάδες μέτρησης για την αποθήκευση αριθμών είναι:

$$1 \text{ byte} = 1\text{B} = 8 \text{ bits} = 8\text{b}$$

$$1 \text{ kbyte} = 2^{10} \text{ bytes} = 1024 \text{ bytes}$$

$$1 \text{ Mbyte} = 2^{10} \text{ kbytes} = 1024 \times 1024 \text{ bytes}$$

Γιατί υπάρχει περιορισμός στους αριθμούς

- ❑ Σχετίζεται με την αναπαράσταση αριθμών στον υπολογιστή
- ❑ Τρία συστήματα αριθμών:
 - Οι αριθμοί μέτρησης 0,1,2,...,32767
Συνδέεται με τον δείκτη των καταλόγων της μηχανής και άρα το εύρος των αριθμών είναι προσδιορισμένο.
 - Οι αριθμοί σταθερής υποδιαστολής
Οι αριθμοί αυτοί έχουν σταθερό μήκος = μήκος του word του Η/Υ ή κάποιο πολλαπλάσιο
Η διαφορά μεταξύ διαδοχικών αριθμών είναι η ίδια
Τέτοιοι είναι οι αριθμοί των υπολογισμών του χεριού (Όλοι οι πίνακες είναι συνήθως σταθερής υποδιαστολής)
 - Οι αριθμοί κινητής υποδιαστολής (floating point numbers)
Σχετίζονται με τη λεγόμενη επιστημονική σήμανση
Το σύστημα αυτό είναι σχεδιασμένο για να περιγράφει και μικρούς αλλά και μεγάλους αριθμούς


Αναπαράσταση αριθμών στον Η/Υ

- Οι αριθμοί κινητής υποδιαστολής (floating point numbers)

$$.31415927 \times 10^1$$

$$.12345678 \times 10^{-1}$$

$$-.12345678 \times 10^4$$



Mantissa: Το πρώτο τμήμα των 8 ψηφίων στους παραπάνω αριθμούς

Εκθέτης: Το τελευταίο ψηφίο (με εύρος τιμών [-38, +38])

Η mantissa και ο εκθέτης αποθηκεύονται στο ίδιο word της μηχανής

Σαν αποτέλεσμα η mantissa ενός αριθμού κινητής υποδιαστολής είναι μικρότερου μήκους του αντίστοιχου αριθμού σταθερής υποδιαστολής

Απεικόνιση αριθμών στους υπολογιστές

- Ένας ακέραιος αριθμός 2^N αντιπροσωπεύεται από N bits

Το 1^ο bit δίνει το πρόσημο

Τα υπόλοιπα $N-1$ bits χρησιμοποιούνται για τον αριθμό

Άρα 32 bits INTEGERS θα πέρνουν τιμές στο διάστημα:

$$-2147483648 < \text{Integer}_{32} < 2147483648 \quad 2^{31}$$

- Η αριθμητική με Integer αριθμούς είναι ακριβής εκτός από **υπερχείλιση** (overflow) και **υποχείλιση** (underflow)

- Οι αριθμοί **απλής ακρίβειας** (single precision ή floating - point F) χρειάζονται 4 bytes (32 bits) για την αναπαράστασή τους

Τα οποία χρησιμοποιούνται ως εξής:

{	23 bits για mantissa
	8 bits για εκθέτη
	1 bit για το πρόσημο

Η μικρότερη mantissa είναι $2^{-23} \sim 10^{-7}$

Εκθέτης:

2^a όπου $a \in [0, 255]$ (2^8 bits)

← “bias”

Απεικόνιση αριθμών

□ Για αριθμούς διπλής ακρίβειας χρειάζονται 8 bytes δηλαδή 64 bits

και χρησιμοποιούνται ως εξής:

- 52 bits για την mantissa
- 1 bit για το πρόσημο
- 11 bits για τον εκθέτη (**bias=1023**)

Η μικρότερη mantissa είναι $2^{-52} \sim 10^{-16}$

Ο εκθέτης είναι $2^{\alpha-1023}$ με $\alpha \in [0, 2048]$ (2^{11} bits)

Διάστημα

$$2.9 \times 10^{-39} \leq \text{float}_{32} \leq 3.4 \times 10^{38}$$

$$10^{-322} \leq \text{double}_{64} \leq 10^{308}$$

Ακρίβεια

float_{32} : 5-7 δεκαδικά ψηφία

double_{64} : 16 δεκαδικά ψηφία

Αριθμοί κινητής υποδιαστολής

□ Έχουν μια σειρά από ιδιαιτερότητες:

- Ας υποθέσουμε για απλότητα ότι έχουμε μια mantissa με 3 ψηφία και ο εκθέτης αποτελείται από 1 ψηφίο:

Το μηδέν, $0 = .000 \times 10^{-9}$ είναι απομονωμένο από τους υπόλοιπους αριθμούς αφού οι αμέσως επόμενοι θετικοί αριθμοί είναι $.100 \times 10^{-9}$ και $.101 \times 10^{-9}$ βρίσκονται 10^{-12} μακριά!!

Δεν υπάρχει κανένας άλλος αριθμός με mantissa να ξεκινά από 0

- Κάθε δεκάδα έχει ακριβώς τον ίδιο πλήθος αριθμών: **συνολικά 900** πηγαίνοντας από $.100 \times 10^a$ μέχρι $.999 \times 10^a$ με $a = -9, -8, \dots, 0, \dots, 8, 9$

0.999x10 ⁻¹ 0.100x10 ⁰	1x10 ⁻⁴
0.999x10 ⁰ 0.100x10 ¹	1x10 ⁻³
0.999x10 ¹ 0.100x10 ²	1x10 ⁻²

- Σε κάθε δεκάδα οι 900 αριθμοί είναι χωρισμένοι σε ίσα διαστήματα

αλλά η διαφορά μεταξύ 2 δεκάδων είναι άνιση

Υπάρχει γεωμετρικό πήδημα

Αριθμοί κινητής υποδιαστολής

- ❑ Ο αριθμός 0 και -0 (δηλαδή -0.000×10^{-9}) είναι λογικά ίδιοι
Οι περισσότεροι υπολογιστές τους έχουν ίδιους αλλά μερικοί όχι
 - Συνήθως δεν υπάρχει άπειρο που να αντιστοιχεί στο μηδέν
- ❑ Στα μαθηματικά υπάρχει ένα και μοναδικό μηδέν
- ❑ Στο προγραμματισμό υπάρχουν 2 είδη:
 - Αυτό που εμφανίζεται σε εκφράσεις όπως $\alpha \cdot 0 = 0$ που συμπεριφέρεται σαν κανονικό 0
 - Αλλά και το 0 που εμφανίζεται σε σχέσεις της μορφής $\alpha - \alpha = 0$
Αυτό το 0 μπορεί να προέρχεται από $1 - (1 - \epsilon)(1 + \epsilon) = 0$ όταν το $\epsilon < \frac{1}{2}10^{-3}$ όπου το 3 είναι ο αριθμός των δεκαδικών ψηφίων της mantissa
 - Αυτό το τελευταίο 0 εμφανίζεται κατά την αφαίρεση 2 σχεδόν ίσου μεγέθους αριθμών
 - Αποτελεί την πηγή πολλών σφαλμάτων στους υπολογισμούς με ένα ηλεκτρονικό υπολογιστή