ΜΑΘΗΜΑ 12
ΑΝΑΔΡΟΜΗ

ΠΕΡΙΕΧΟΜΕΝΑ:

1. Ανάδρομη
   1. Παράδειγμα: Fibonacci
   2. Παράδειγμα: ΜΚΔ με τον Αλγόριθμο του Ευκλείδη
2. Algorithm: Binary Search
3. Algorithm: Insertion Sort
4. Data Project: CRUD - Update

Αγγελική Γ.
Σταυριανή Γ.

Ασημένιος Χορηγός Μαθήματος

Όλγα Κ.

Χάλκινος Χορηγός Μαθήματος

- Έχοντας ορίσει μια συνάρτηση:
  - Μπορούμε να την καλέσουμε από οποιοδήποτε σημείο του προγράμματος (που έπεται τον ορισμό της)
  - **Άρα μπορούμε να καλέσουμε μία συνάρτηση ακόμη και μέσα σε μία συνάρτηση.**

Παράδειγμα 1: funct.call.py

```
def square(x):
   return x*x

def cube(x):
   return x**3

def f(x):
   return 3*cube(x)+4*square(x)+x+1

print("  x : ", end="")
for i in range(10):
   print("| " + str(i).center(6), end="")
print("|\nf(x): ", end="")
for i in range(10):
   print("| " + str(f(i)).center(6), end="")
print("|")
```

- Μία **Αναδρομική Συνάρτηση (Recursive Function)** είναι:
  - Μία συνάρτηση που καλεί τον εαυτό της

Παρατήρηση:
- Η αναδρομή, είναι ιδιαίτερα χρήσιμη για τον υπολογισμό ποσοτήτων που ορίζονται αναδρομικά (δηλαδή μέσω του εαυτού τους), αλλά και στην κατασκευή αλγορίθμων.

Παράδειγμα 2: factorial.py & factorial.print.py

Το παραγοντικό ορίζεται αναδρομικά ως εξής:

$$n! = \begin{cases} n \cdot (n-1)! & \text{αν } n > 1 \\ 1, & \text{αν } n = 1 \end{cases}$$

και η υλοποίηση του μέσω αναδρομικής συνάρτησης είναι η:

```
def factorial(n):
   if n == 1:
      return 1
   else:
      return n*factorial(n-1)

for i in range(1,11):
   print(f"factorial({i})={factorial(i)} ")
```

Άσκηση 1:

Ορίστε την ίδια συνάρτηση, αλλά ο κώδικας να είναι επαναληπτικός (να χρησιμοποιεί for) και όχι αναδρομικός.

Οι αριθμοί Fibonacci

Η ακολουθία Fibonacci ορίζεται ως εξής:

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{αν } n > 2 \\ 0, & \text{αν } n = 0 \\ 1, & \text{αν } n = 1 \end{cases}$$

και μπορούμε να υπολογίσουμε το n-οστό αριθμό ως εξής:

```
# fibonacci.py
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)


for i in range(11):
    print(f"fibonacci({i})={fibonacci(i)}")
```

Ιστορικό:

Το πρόβλημα που μελέτησε ο Fibonacci (το 1202) ήταν η ταχύτητα αναπαραγωγής των κουνελιών ανά μήνα:

- Μοντέλο: Ένα νεογέννητο κουνέλι θέλει ένα μήνα για να μπορεί να αναπαραχθεί. Μόλις γονιμοποιείται, θέλει ένα μήνα για να γεννήσει. Κάθε κουνέλα γεννάει αυστηρά ένα νέο ζευγάρι κουνελιών (Αρσενικό – Θηλυκό).

Μήνας	Νεογέννητα	Αναπαράγονται	Σύνολο
0	0	0	0
1	1	0	1
2	0	1	1
3	1	1	2
4	1	2	3
5	2	3	5

- [Οι αριθμοί του πίνακα μετράνε ζεύγη κουνελιών]

Άσκηση 2:

Αλλάξτε τα όρια της επανάληψης ώστε να υπολογιστούν οι 100 πρώτοι αριθμοί Fibonacci.
Τι παρατηρείτε;

Άσκηση 3:

Μελετήστε τον κώδικα Fibonacci.print.py. Μπορείτε να εντοπίσετε γιατί είναι τόσο αργός ο κώδικας;

Άσκηση 4:

Υλοποιήστε τον υπολογισμό επαναληπτικά. Παρατηρείτε βελτίωση στο χρόνο υπολογισμού;

1.2. Παράδειγμα: Αλγόριθμος του Ευκλείδη για Μ.Κ.Δ.

Ο **αλγόριθμος του Ευκλείδη** για την εύρεση του **Μέγιστου Κοινού Διαιρέτη** δύο (φυσικών) αριθμών:

- Ξεκινά με ένα ζεύγος φυσικών και σχηματίζει ένα νέο ζευγάρι με τον μικρότερο αριθμό και την διαφορά του μικρότερου από τον μεγαλύτερο αριθμό.
- Η διαδικασία επαναλαμβάνεται εωσότου οι αριθμοί γίνουν ίσοι. Ο αριθμός αυτός είναι ο ΜΚΔ των αρχικών αριθμών.

Μαθηματικά ο ΜΚΔ(a,b) όπου a,b είναι φυσικοί:

- Είναι ίσο με a, αν a=b
- Είναι ίσο με ΜΚΔ(a,b-a), αν a<b
- Είναι ίσο με ΜΚΔ(a-b,b), αλλιώς

Παράδειγμα:

A	B
255	155
100	155
100	55
45	55
45	10
35	10
25	10
15	10
5	10
5	5

<u>Άσκηση 5:</u>

Υλοποιήστε τον αλγόριθμο του Ευκλείδη.

<u>Άσκηση 6:</u>

Ορίστε την αναδρομική συνάρτηση print_list(list) η οποία παίρνει σαν όρισμα μία λίστα και τυπώνει τα περιεχόμενα της αναδρομικά Δοκιμάστε δύο παραλλαγές της:

- Πρώτα να τυπώνεται το πρώτο στοιχείο και έπειτα να γίνεται αναδρομή στα υπόλοιπα στοιχεία.
- Πρώτα να γίνεται αναδρομή στα υπόλοιπα στοιχεία και έπειτα να τυπώνεται το πρώτο στοιχείο

<u>Άσκηση 7:</u>

“Οι συνδυασμοί των n ανά k” υπολογίζονται από τον αναδρομικό τύπο:

$$C(n,k) = \begin{cases} C(\mathrm{n}-1,\mathrm{k}-1)+\mathrm{C}(\mathrm{n}-1,\mathrm{k}) & \text{αν } n>k \\ 1, & \text{αν } n=k \\ n, & \text{αν } k=1 \end{cases}$$

Κατασκευάστε πρόγραμμα που υπολογίζει το C(n,k) και υπολογίστε μέσω αυτού, την ποσότητα C(49,6)
[Η οποία, παρεπιμπτόντως, μας δίνει τους διαφορετικούς τρόπους για να συμπληρώσουμε μια στήλη του ΛΟΤΤΟ]

Το **<u>πρόβλημα της αναζήτησης στοιχείου σε πίνακα</u>** (searching):

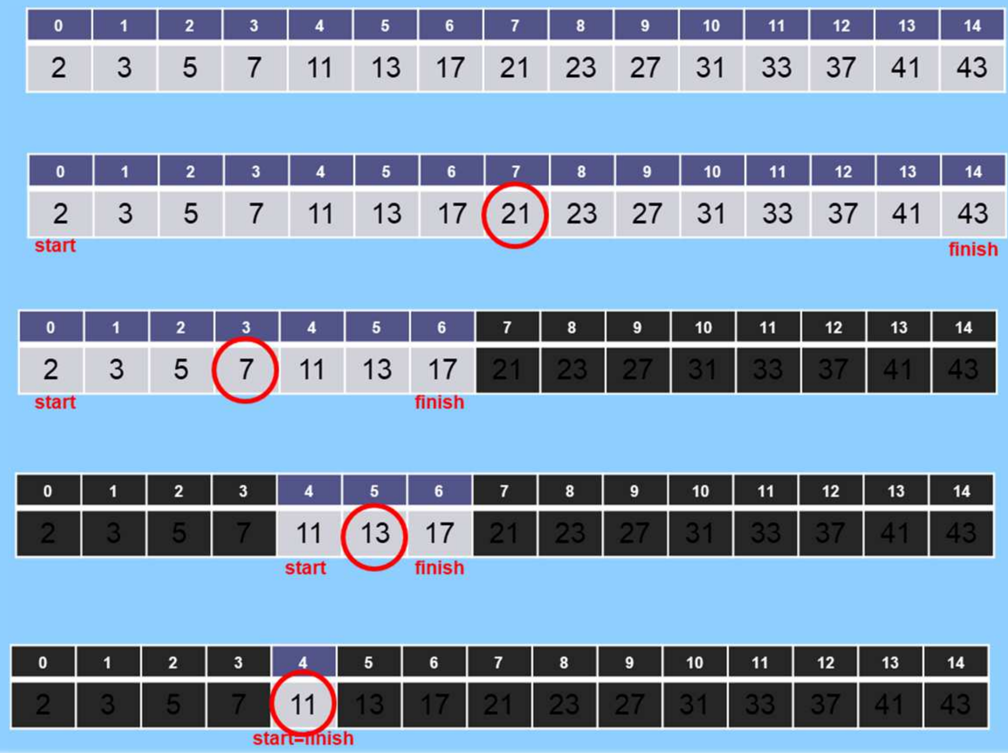
- **Είσοδος(input):** Δίνεται ένας πίνακας στοιχείων A και ένα στοιχείο x.
- **Έξοδος(output):** Η θέση του στοιχείου στον πίνακα (-1 αν δεν υπάρχει)

‘Εχουμε ήδη μελετήσει **τη σειριακή αναζήτηση** (linear search) και τώρα θα μελετήσουμε **τη δυαδική αναζήτηση** (binary search) που δουλεύει μόνο σε ήδη ταξινομημένο πίνακα.

Το **αλγοριθμικό σκεπτικό** της δυαδικής αναζήτησης είναι:

- Αν το μεσαίο στοιχείο του πίνακα είναι το x, το στοιχείο βρέθηκε! Επιστρέφουμε τη θέση του.
- Αν το x είναι μικρότερο από το μεσαίο στοιχείο τότε ψάχνουμε στο κομμάτι του πίνακα από την αρχή μέχρι το μεσαίο στοιχείο
- Αν το x είναι μεγαλύτερο από το μεσαίο στοιχείο τότε ψάχνουμε στο κομμάτι του πίνακα από το μεσαίο στοιχείο μέχρι το τέλος

Παράδειγμα Εκτέλεσης (ψάχνοντας το 11):



<u>Άσκηση 8:</u>
Υλοποιήστε τον αλγόριθμο με μία αναδρομική συνάρτηση.
[Αυξημένης δυσκολίας άσκηση – Συμβουλευθείτε το βίντεο]

<u>Άσκηση 9:</u>
Υλοποιήστε τον αλγόριθμο με μία μη αναδρομική συνάρτηση.

Algorithm Project: Insertion Sort (Ταξινόμηση με Εισαγωγή)

Το **<u>πρόβλημα της ταξινόμησης πίνακα</u>** (sorting):

- **Είσοδος(input):** Δίνεται ένας πίνακας στοιχείων.
- **Έξοδος(output):** Ο ταξινομημένος πίνακας.

Υπάρχουν πραγματικά πολλοί αλγόριθμοι για την ταξινόμηση των στοιχείων του πίνακα.

Το **σκεπτικό** της ταξινόμησης με εισαγωγή είναι:

- Έχοντας ταξινομήσει τα στοιχεία 0..i-1
- Τοποθέτησε το στοιχείο i στις θέσεις 0…i-1 κάνοντας ανταλλαγές με τα προηγούμενα, μέχρι να βρεθεί κάποιο μικρότερο στοιχείο.

<u>Unpacking και swap:</u>

- Η Python δίνει τη δυνατότητα να αναθέσουμε σε μεταβλητές τα στοιχεία ενός επαναλήπτη (iterable, π.χ. λίστα, tuple, κ.λπ.) με μία εντολή της μορφής (unpacking):

```
a, b = [0, 1]
```

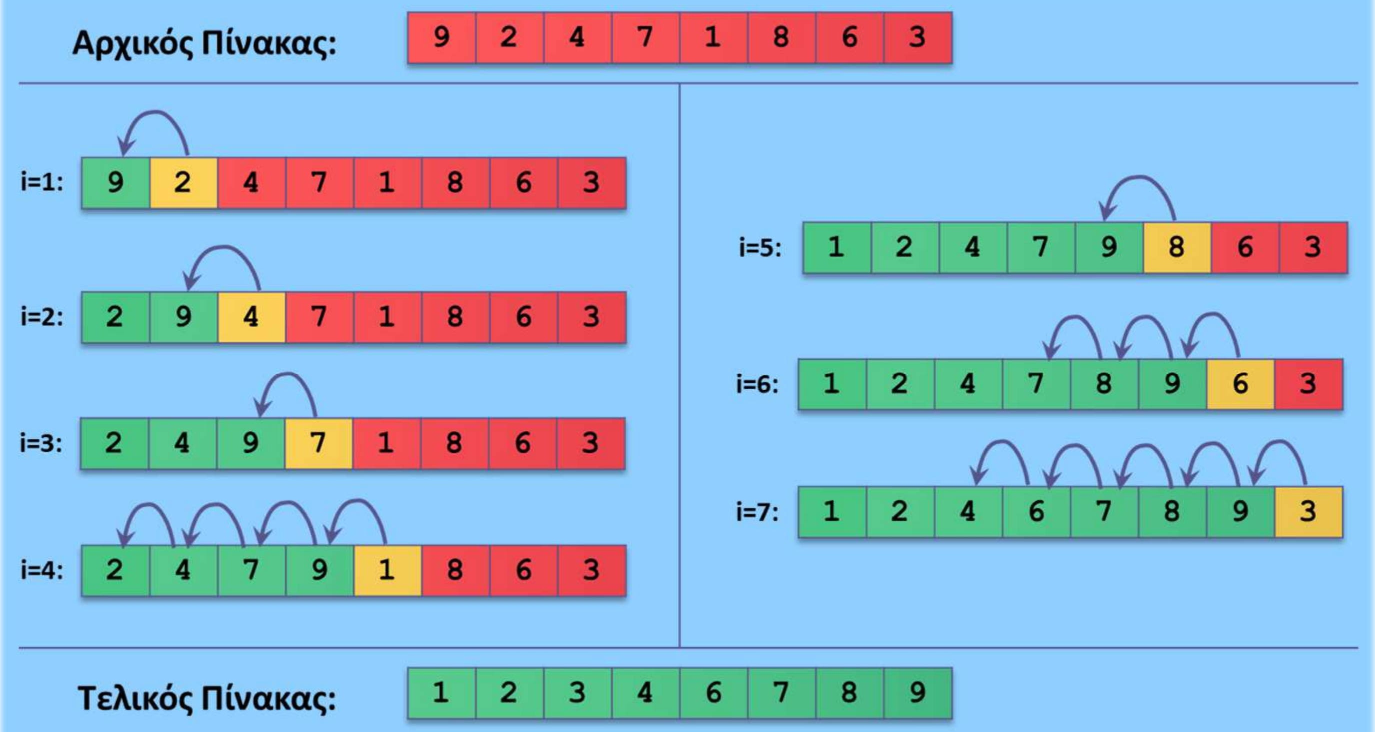
- Για να κατασκευάσουμε ένα tuple οι παρενθέσεις δεν είναι υποχρεωτικές.
- Έτσι ο παρακάτω κώδικας:

```
a, b = b, a
```

Προκαλεί ανταλλαγή των τιμών των μεταβλητών (**swap** values)

[Δες και swap.py]

Παράδειγμα Εκτέλεσης:



<u>Άσκηση 10:</u>

Υλοποιήστε τον αλγόριθμο Insertion Sort

<u>Άσκηση 11:</u>

Υλοποιήστε τον αλγόριθμο Insertion Sort ώστε να ταξινομεί τα στοιχεία σε φθίνουσα σειρά.

Άσκηση 12.1: Αναζητήσεις

Δημιουργήστε μία νέα συνάρτηση:

- **search_pupil_by_surname**(surname): Παίρνει ένα όρισμα (surname) και επιστρέφει μία λίστα με τους μαθητές, οι οποίοι έχουν το συγκεκριμένο επώνυμο.

Άσκηση 12.2: Update

Υλοποιήστε την ενημέρωση μαθητή:

- Αρχικά να προτρέπει το χρήστη να επιλέξει αν θέλει να ψάξει με επώνυμο ή με id.
- Αν ψάξει με επώνυμο και υπάρχουν παραπάνω του ενός μαθητές, τότε να τυπώνει τα πλήρη στοιχεία των μαθητών και να προτρέπει το χρήστη να επιλέξει το id του μαθητή που επιθυμει.
- Σε κάθε περίπτωση αναζήτησης αν δεν υπάρχουν μαθητές που να ταιριάζουν με τα κριτήρια αναζήτησης, τότε να τυπώνεται μήνυμα λάθους και να επιστρέφεται ο έλεγχος στο αρχικό μενου.
- Έχοντας ένα μαθητή πλέον, να γίνεται η ενημέρωση. Να ερωτάται ο χρήστης ποιο πεδίο θέλει να διορθώσει (εκτός του id) να λαμβάνεται η νέα τιμή του πεδίου και να διορθώνεται η εγγραφή.

"In the face of ambiguity, refuse the temptation to guess."

Zen of Python #12