ΠΕΡΙΕΧΟΜΕΝΑ:

1. **Δομή Επανάληψης While**
2. **Δομή Επανάληψης For**
3. **Break και Continue**
4. **Εμφωλιασμένοι Βρόχοι**

Γιάννης

Σμαραγδένιος Χορηγός Μαθήματος

Έφη

Χάλκινος Χορηγός Μαθήματος

- Με τον όρο **"Επανάληψη"** εννοούμε τις δύο εντολές της Python **(for και while)**
  - οι οποίες **μας επιτρέπουν να επαναλάβουμε πολλές φορές τον ίδιο κώδικα.**
  - (το οποίο είναι εξαιρετικά χρήσιμο και κύριο χαρακτηριστικό του προγραμματισμού)

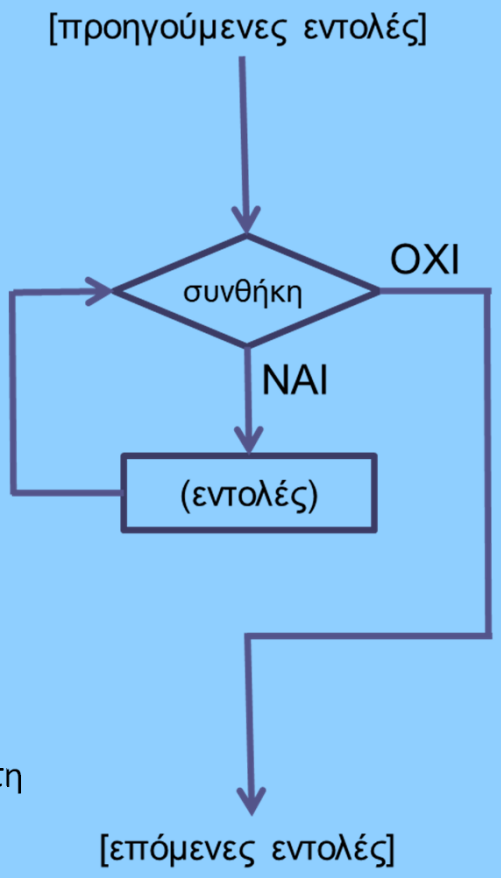
While

- Συντακτικό:

```
[προηγούμενες εντολές]

while συνθήκη:
    εντολή-1
    ...
    εντολή-N

[επόμενες εντολές]
```



- Εξήγηση:
  - Ελέγχεται η συνθήκη
  - Αν η συνθήκη είναι True
    - εκτελούνται οι εντολές
    - μεταφέρεται ο έλεγχος πάλι στη συνθήκη (αρχή)
  - Αν η συνθήκη είναι False
    - Βγαίνουμε από τη while και προχωράμε στις επόμενες εντολές

Παρατηρήσεις:

- Ισχύουν οι ίδιοι κανόνες στοίχισης όπως στη δομή ελέγχου.
- Προσοχή στην άνω κάτω τελεία.
- Συνήθως μεταφράζεται "Επανάλαβε όσο"
- Μία εντολή επανάληψης αναφέρεται συχνά και σαν **βρόχος (ή loop)**
- Ενώ οι εντολές μέσα στο βρόχο αναφέρονται και σαν **σώμα του βρόχου** και αποτελούν ένα **μπλοκ κώδικα**.

Παράδειγμα 1: while1.py

Με την εντολή while μπορούμε να επαναλάβουμε ένα τμήμα κώδικα πολλές φορές:

```
i = 1
while i <= 5:
    print("Hello!")
    i += 1
```

Παράδειγμα 2: while2.py

Η τιμή της μεταβλητής η οποία τροποποιείται σε κάθε βήμα, είναι πολύ σημαντική και "καθοδηγεί" την επανάληψη. Συνήθως μάλιστα, τη χρησιμοποιούμε στην επανάληψη:

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

Παράδειγμα 3: while3.py

Συνήθως χρησιμοποιούμε αυτήν την τιμή στους υπολογισμούς μας:

```
i = 0
while i <= 4:
    print(2*i+1)
    i += 1
```

Ορολογία:

- Εντολή Αρχικοποίησης (i=0)
- Συνθήκη (i<=4)
- Βήμα αύξησης (i+=1)

Παράδειγμα 4: while4.py

Η μεταβλητή μπορεί να τροποποιείται αναλόγως τις ανάγκες μας (π.χ. να μειώνεται αντί να αυξάνεται):

```
cnt = 9
while cnt >= 1:
    print(cnt)
    cnt -= 2
```

Παράδειγμα 5: while5.py

Από τα πιο συνηθισμένα προγραμματιστικά λάθη, είναι να πέσει το πρόγραμμα μας σε ατέρμονα βρόχο (infinite loop)!

```
cnt = 9
while cnt >= 1:
    print(cnt)
    cnt += 2
```

Παράδειγμα 6: while6.py

Μπορούμε πλέον να ελέγξουμε καλύτερα την είσοδο του χρήστη, π.χ.:

```
number = int(input("Enter a number(0-9): "))
while number < 0 or number > 9:
    number = int(input("Between 0 and 9 please: "))

print("You entered: " + str(number))
```

Παράδειγμα 7: while7.py

Και αρκετές φορές χρησιμοποιούμε μία λογική μεταβλητή για να κάνουμε τον έλεγχο της συνθήκης:

```
active = True
while active:
    user_input = input("Type string or 'quit': ")

    if user_input == "quit":
        print("Bye bye!")
        active = False
    else:
        print("Why " + user_input + "?!")
```

Άσκηση 1: Τετράγωνα Αριθμών

Κατασκευάστε ένα πρόγραμμα το οποίο:

- Θα δέχεται από την είσοδο έναν αριθμό και θα τυπώνει το τετράγωνό του
- Το παραπάνω να γίνεται επαναληπτικά, μέχρι ο χρήστης να πληκτρολογήσει “quit”, οπότε και θα τερματίζει.

Άσκηση 2: Άθροισμα αριθμών

Κατασκευάστε ένα πρόγραμμα το οποίο:

- Θα διαβάζει από την είσοδο 10 αριθμούς
- Θα τους προσθέτει και θα τυπώνει το άθροισμά τους.

Σημείωση: Ελαφρά προχωρημένη άσκηση. Συμβουλευθείτε το βίντεο.

Άσκηση 3: Άθροισμα αριθμών

Επαναλάβετε την άσκηση 2, με τη διαφορά ότι οι 10 αριθμοί θα είναι αρχικά αποθηκευμένοι σε μία λίστα.

Άσκηση 4: Μέγιστος (ξανά)

Κατασκευάστε πρόγραμμα που:

- Ορίζει μία λίστα 10 ακεραίων
- Εντοπίζει το μέγιστο από αυτούς (χρησιμοποιώντας επανάληψη)

- **Η δομή επανάληψης for:**
  - επαναλαμβάνει τον κώδικά της **για κάθε στοιχείο που ανήκει σε μια ακολουθία (sequence)**
  - **Ακολουθίες** είναι τα strings και οι λίστες, αλλά και τα εύρη (ή ranges σημερινό μάθημα) και οι πλειάδες (ή tuples επόμενο μάθημα)

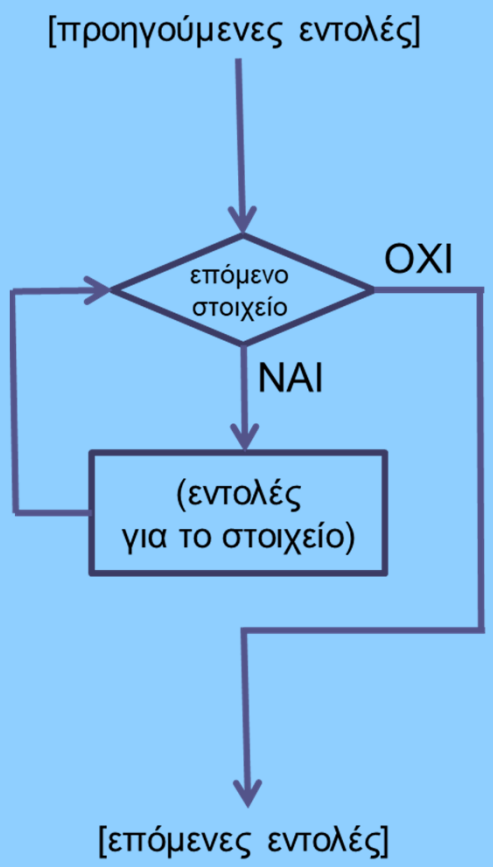
For

- Συντακτικό:

```
[προηγούμενες εντολές]

for element in sequence:
    εντολή-1
    ...
    εντολή-N

[επόμενες εντολές]
```



- Εξήγηση:
  - οι εντολές θα τρέξουν διαδοχικά για κάθε στοιχείο που ανήκει στην ακολουθία (ένα-ένα τα στοιχεία με τη σειρά που εμφανίζονται στην ακολουθία)
  - Το element αντικαθίσταται στις εντολές με το τρέχον στοιχείο.

Παρατηρήσεις:

- Ισχύουν οι ίδιοι κανόνες στοίχισης όπως στη δομή ελέγχου.
- Προσοχή στην άνω κάτω τελεία.
- Συνήθως μεταφράζεται “Για ”
- Ακολουθία σημαίνει στα μαθηματικά, ότι η σειρά έχει σημασία. Έτσι το for εφαρμόζεται σε αντικείμενα όπως τα strings και οι λίστες, όπου υπάρχει σειρά (διάταξη, είναι δηλαδή διατεταγμένα)

Παράδειγμα 8: for.list1.py

Βλέπουμε πως τρέχει η for διατρέχοντας τα στοιχεία μιας λίστας:

```
players = ["Jordan", "Giannis", "LeBron"]

for player in players:
    print(player + " plays!")
```

Παράδειγμα 9: for.list2.py

Με τα εργαλεία που έχουμε ήδη μάθει, πλέον μπορούμε να γράψουμε αρκετά περίπλοκα και χρήσιμα προγράμματα:

```
semester_grades = [4, 6, 3, 8, 10]

passed = 0
sum_grades = 0
for grade in semester_grades:
    if grade >= 5:
        passed += 1
        sum_grades += grade

print("I've succeeded in " + str(passed) + " lessons")
print("My average is " + str(sum_grades/passed))
```

Ορολογία:

- Η μεταβλητή passed επειδή **μετράει το πλήθος** κάποιων πραγμάτων, αναφέρεται ως **μετρητής (counter)**
- Η μεταβλητή sum_grades επειδή **αθροίζει κάποια πράγματα,** αναφέρεται ως **συσσωρευτής (accumulator)**

Παράδειγμα 10: for.string.py

Και το string είναι μια ακολουθία! Συγκεκριμένα, είναι μία ακολουθία χαρακτήρων:

```
string = "Once upon a time..."

for char in string:
    print(char)
```

- Η συνάρτηση range: **range(start, finish, step)**
  - Επιστρέφει μία ακολουθία ακεραίων με αρχή το **start**, τέλος το **finish-1** και βήμα **step**.
  - Το step αν παραλειφθεί, θεωρείται ίσο με 1.
- π.χ.1: range(0,10): Επιστρέφει (0,1,2,…,9)
- π.χ.2: range(3,6): Επιστρέφει (3,4,5)
- π.χ.3: range(0,10,2): Επιστρέφει (0,2,4,…,8)

Και είναι επίσης μία ακολουθία. Άρα μπορούμε να τη χρισιμοποιήσουμε με τη for για να διατρέξουμε τα στοιχεία της.

Παράδειγμα 11: range.py

Με την παρακάτω χρήση τυπώνουμε τα πολλαπλάσια του 3 από το 0 έως το 9:

```
for number in range(0,10,3):
    print(number)
```

Άσκηση 5: Παιχνίδι με τη range

Κατασκευάστε πρόγραμμα το οποίο:

- Θα τυπώνει όλους τους άρτιους αριθμούς από το 10 έως το 20
- Θα τυπώνει όλους τους περιττούς αριθμούς από το 19 έως το 11 (φθίνουσα σειρά)
- Θα τυπώνει εκείνους τους περιττούς αριθμούς από το 1 έως το 29 που είναι επίσης πολλαπλάσια του 3.

Άσκηση 6: Πρόσβαση σε συγκεκριμένα στοιχεία μιας λίστας

Κατασκευάστε πρόγραμμα το οποίο:

- Θα ορίζει μία λίστα με 5 πόλεις
- Θα τυπώνει εκείνες τις πόλεις που βρίσκονται σε άρτιες θέσεις της λίστας.

Άσκηση 7: Πρόσβαση σε συγκεκριμένα στοιχεία μιας λίστας

Κατασκευάστε πρόγραμμα το οποίο θα κατασκευάζει μία λίστα με όλους τους άρτιους αριθμούς από το 0 έως το 1000 και έπειτα την τυπώνει.

Άσκηση 8: Ταξινόμηση αριθμών

Κατασκευάστε ένα πρόγραμμα το οποίο:

- Θα δέχεται από την είσοδο έναν αριθμό που πρέπει να είναι μεταξύ του 3 και του 20, έστω N
- Θα διαβάζει από την είσοδο N ακεραίους και θα τους αποθηκεύει σε μία λίστα.
- Θα ταξινομεί τη λίστα των αριθμών
- Θα την τυπώνει

ΜΑΘΗΜΑ 5: Επανάληψη

3. Break και Continue

python 3 psounis

Break

- Αν στη διάρκεια της εκτέλεσης ενός βρόχου αποφασίσουμε ότι θέλουμε να διακόψουμε την εκτέλεση του (άρα να μην εκτελεστούν τα επόμενα βήματά του), χρησιμοποιουμε την εντολή break:
- Συνηθισμένο συντακτικό (π.χ. σε for, αλλά ισχύει και για while):

```
for element in sequence:
    ...
    if condition:
        break
    ...
```

Continue

- Αν στη διάρκεια της εκτέλεσης ενός βρόχου αποφασίσουμε ότι θέλουμε να παρακάμψουμε την τρέχουσα επανάληψη και να προχωρήσουμε στην επόμενη, χρησιμοποιουμε την εντολή continue
- Συνηθισμένο συντακτικό (π.χ. σε for, αλλά ισχύει και για while):

```
for element in sequence:
    ...
    if condition:
        continue
    ...
```

Παράδειγμα 8: break.py

```
numbers = [1, 8, 7, 4, 11, 12, 2, 9, 2, 5]
search = 2
for number in numbers:
    if search == number:
        print("Found it!")
        break
```

Το πρόγραμμα ψάχνει για ένα συγκεκριμένο στοιχείο σε μία λίστα (σειριακή αναζήτηση). Διακόπτει όταν βρεθεί το στοιχείο.

Παράδειγμα 9: continue.py

```
for number in range(0,10):
    if number % 2 == 1:
        continue
    print(number)
```

Else:

- Εφόσον ένας βρόχος φτάσει ομάλά στον τερματισμό του (χωρίς break) μπορούμε να προσθέσουμε ένα μπλοκ που θα εκτελείται στο τέλος με την else.

Π.χ.10: else.py

```
for number in range(0,10):
    print(number)
else:
    print("finished!")
```

Άσκηση 9.1: Μάντεψε τον Αριθμό

Θα κατασκευάσουμε το πρώτο μας παιχνίδι, το "μάντεψε τον αριθμό"

- Το πρόγραμμα θα "κρύβει" έναν αριθμό.
- Θα προσπαθήσουμε να τον μαντέψουμε.
- Το πρόγραμμα θα μας καθοδηγεί: π.χ. αν πληκτρολογήσουμε αριθμό που είναι μικρότερος από τον κρυμμένο αριθμο, θα εμφανίζει κατάλληλο μήνυμα.

Αρχικά ξεκινήστε την κατασκευή:

- Αποθηκεύοντας σε μία μεταβλητή τον "κρυμμένο" αριθμό
- Επαναληπτικά ο χρήστης να εισάγει έναν αριθμό και αν αυτός είναι ίδιος με τον κρυμμένο αριθμό τότε να εμφανίζει μήνυμα επιτυχίας και να τερματίζει.
- Σκεφθείτε πως μπορείτε να κάνετε την επανάληψη να τρέχει "για πάντα", ώστε το πρόγραμμα να τερματίζει μόνο εφόσον ο χρήστης έχει βρει τον αριθμό.

Άσκηση 9.2: Επέκταση με μηνύματα

Επεκτείνετε το πρόγραμμα ώστε:

- Να εμφανίζει μηνύματα καθοδήγησης για το αν ο χρήστης πληκτρολόγησε μικρότερο, ή μεγαλύτερο αριθμό από τον κρυμμένο αριθμό

Άσκηση 9.3: Επέκταση με μέγιστο αριθμό προσπαθειών

Επεκτείνετε το πρόγραμμα ώστε:

- Να ορίζει μία μεταβλητή που απεικονίζει το μέγιστο πλήθος επιτρεπτών προσπαθειών.
- Να ορίζει ακόμη μία μεταβλητή που μετράει πόσες προσπάθειες έχει κάνει ο χρήστης.
- Να σταματάει την εκτέλεση και να βγάζει μήνυμα στο χρήστη ότι έχασε, αν φτάσει στο μέγιστο πλήθος επιτρεπτών προσπαθειών.

Εμφωλιασμένοι Βρόχοι

- Οι δύο εντολές επανάληψης, είναι απλά δύο ακόμη (πιο περίπλοκες) εντολές.
- Έτσι είναι συχνό να έχουμε μια εντολή επανάληψης μέσα σε μια εντολή επανάληψης.
  - που συχνά αναφέρεται σαν **nested loops** (εμφωλιασμένοι βρόχοι)

Παράδειγμα 11: nested.loops.py

Το παρακάτω πρόγραμμα υπολογίζει την προπαίδεια των 1,2,…, 9

```
for i in range(1,10):
    for j in range(1,11):
        print(str(i) + "*" + str(j) + "=" + str(i*j))
    print("============")
```

Ο εσωτερικός βρόχος τρέχει για κάθε τιμή του i του εξωτερικού βρόχου.

Παράδειγμα 12: nested.loops2.py

Το παρακάτω πρόγραμμα εμφανίζει ένα ενδιαφέρον σχήμα!

```
for i in range(1,6):
    for j in range(1,i+1):
        print("*", end=" ")
    print("")
```

Παρατήρηση: Η print είναι μία πολυμορφική συνάρτηση. Εδώ βλέπουμε ότι αν της βάλουμε ως δεύτερο όρισμα end=“ “, δεν κάνει αλλαγή γραμμής, αλλά συνεχίζει τις εκτυπώσεις στην ίδια γραμμή.

Παράδειγμα 13: nested.loops3.py

Το παρακάτω πρόγραμμα εμφανίζει μια παραλλαγή του προηγούμενου σχήματος!

```
N = 5
for i in range(0, N):
    for j in range(0, N-i-1):
        print(" ", end=" ")
    for j in range(0, i+1):
        print("*", end=" ")
    print("")
```

Άσκηση 10: Το πάρτυ

Κατασκευάστε πρόγραμμα το οποίο:

- Θα διατηρεί σε μία λίστα τους 3 καλύτερους σας φίλους
- Θα διατηρεί σε μία δεύτερη λίστα τους 10 καλεσμένους σε ένα πάρτυ.
- Θα μετράει πόσοι από τους καλύτερους σας φίλους είναι καλεσμένοι στο πάρτυ
- Αν είναι λιγότεροι από 2, τότε θα αρνηθείτε την πρόσκληση για το πάρτυ, αλλιώς θα την αποδεχθείτε.

Άσκηση 11: Πυθαγόρειες τρίάδες

Κατασκευάστε πρόγραμμα το οποίο:

- Θα τυπώνει όλες τις τριάδες (α,β,γ) με την ιδιότητα ότι:

$$\alpha^2 + \beta^2 = \gamma^2$$

- (για ακέραιους: $0 \leq \alpha, \beta, \gamma \leq 20$)

Άσκηση 12: Ισοσκελές Τρίγωνο

Κατασκευάστε ένα πρόγραμμα το οποίο:

- Θα αποθηκεύει το πλήθος των γραμμών σε μία μεταβλητή με όνομα N.
- Θα τύπώνει ένα ισοσκελές τρίγωνο με N γραμμές.
- Π.χ. για N=5:

```
    *
   * * *
  * * * * *
 * * * * * * *
* * * * * * * * *
```

"Although never is often better than *right* now."

Zen of Python #16