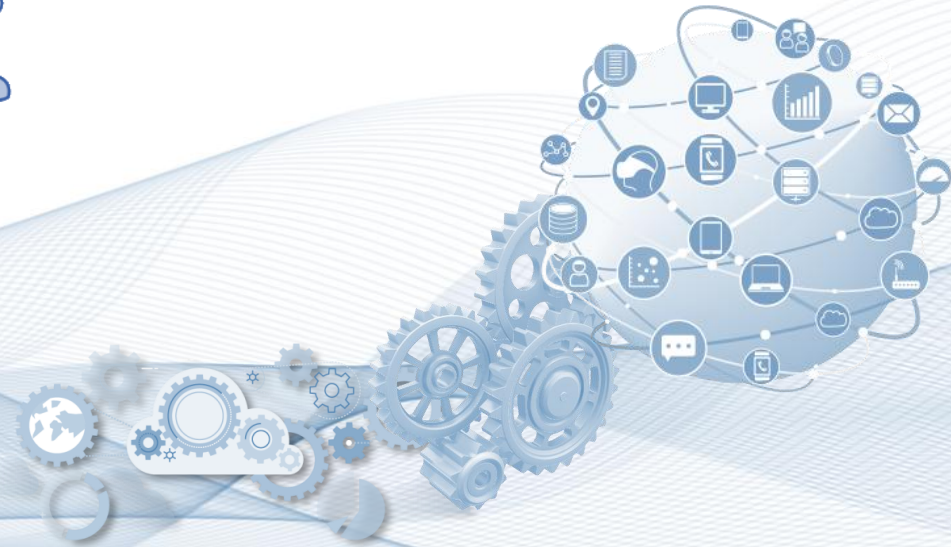


ΤΕΧΝΙΚΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΥΠΟΛΟΓΙΣΤΩΝ

Διάλεξη II

Το Τελικό Βήμα με Scratch – Μεταβλητές & Πρότυπα

Γιάννης Τζήμας, Καθηγητής



Τι θα δούμε σήμερα...

- Προχωρώντας θα δούμε πιο αναλυτικά τις έννοιες των:
 - **μεταβλητών**: όνομα, τιμή, εμφάνιση, αρχικοποίηση
 - και των **προτύπων** (patterns): σχόλια, κεντρικός κώδικας, έλεγχος, μηνύματα, μέτρηση – άθροιση – υπολογισμοί
- Όπως πάντα: οι ίδιες ιδέες σε Scratch, **JavaScript** και **Python**.



Τι είδαμε στα προηγούμενα μαθήματα και τι μας λείπει;

- Οι **τρεις** προγραμματιστικές δεξιότητες που θα πρέπει να καλλιεργήσετε όπως έχουμε πει είναι:
 - Η **αλληλουχία** των εντολών
 - Η **επιλογή** ανάμεσα σε διάφορες εναλλακτικές και
 - Η **επανάληψη**

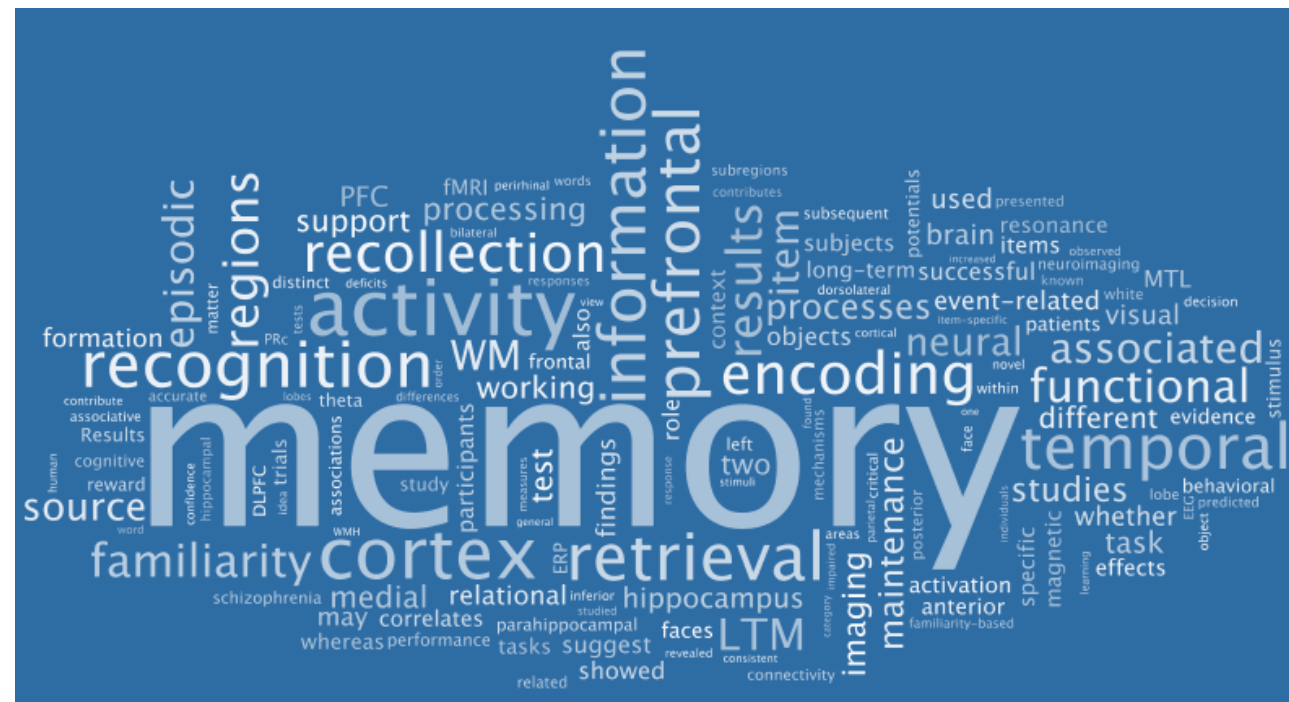
Ποιο είναι το **επόμενο βήμα**;
Στην πράξη χρειαζόμαστε επιπλέον:

- Ένα τρόπο για να **θυμόμαστε πράγματα** – “**variables – μεταβλητές**”
- Δεξιότητες για **επίλυση προβλημάτων** – “**patterns - πρότυπα**”



Μνήμη - Memory

- Όπως πιθανά ήδη ξέρετε από άλλα σας μαθήματα, οι ηλεκτρονικοί υπολογιστές (PCs, laptops, mobiles κ.λπ.) έχουν αρκετή **μνήμη** για την επεξεργασία δεδομένων.
 - 8GB,
 - 16GB,
 - 32GB,
 - Ίσως και **περισσότερο!**



Τύποι Μνήμης

- Μόνιμη

- Τα δεδομένα υπάρχουν ακόμα και αν η συσκευή κλείσει.
- π.χ. σκληροί δίσκοι, memory cards κλπ.

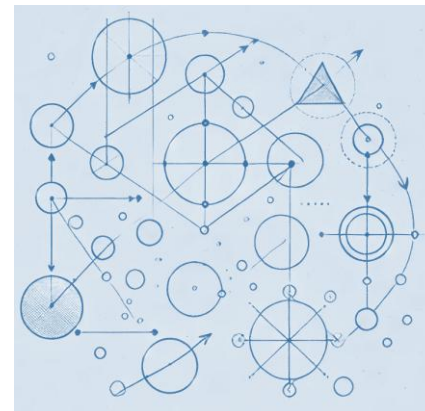
- Προσωρινή

- Τα δεδομένα αποθηκεύονται προσωρινά ώστε να γίνει η απαραίτητη επεξεργασία.
- π.χ. RAM – Random Access Memory



Μεταβλητές (Variables) = Μνήμη (Memory)

- Οι μεταβλητές είναι η **μνήμη** του **προγράμματός σας**.
 - Τις χρησιμοποιούμε για να **αποθηκεύσουμε πληροφορίες** όσο το πρόγραμμα «τρέχει».
- **Γιατί** χρειαζόμαστε μεταβλητές;
 - Μερικά **παραδείγματα...**
 - Για να κάνουμε **μαθηματικούς υπολογισμούς** – «η απόσταση έως την άκρη της σκηνής είναι...»
 - Για να **παρακολουθούμε** διάφορα πράγματα – «το σκορ του παιχνιδιού είναι...»
 - Για να **αποθηκεύουμε τιμές εισόδου** από το χρήστη – «Παρακαλώ πληκτρολογήστε το όνομά σας...»
 - Για να **κρατάμε τιμές εξόδου** που μπορεί να δείξουμε στην οθόνη – «Η σωστή απάντηση στην ερώτηση είναι...»
- **Ενδιαφέρουσα σημείωση:**
 - Όσο το πρόγραμμα τρέχει, οι μεταβλητές βρίσκονται στην **προσωρινή μνήμη** (RAM) – σε **όλες** τις γλώσσες (π.χ. JavaScript, Python), και στο Scratch.
 - Στο **Scratch** όμως, όταν **αποθηκεύουμε** το project, σώζονται στο αρχείο και οι **τρέχουσες τιμές** τους· σε JavaScript/Python χάνονται όταν τελειώσει το πρόγραμμα.



Παραδείγματα στο Scratch

- Έχουμε ήδη χρησιμοποιήσει **προκαθορισμένες μεταβλητές** στα προγράμματα που φτιάξαμε...

- x position
- y position
- direction

- pen color
- pen state (up? down?)
- mouse position
- mouse state

- volume
-

x position

say direction for 2 seconds

y position

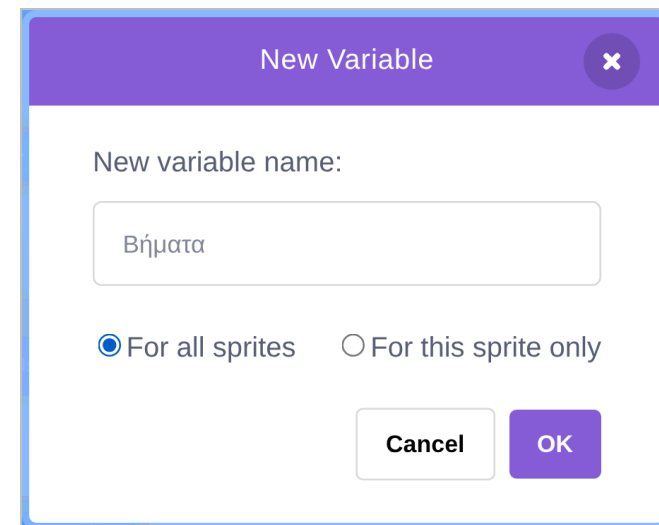
forever
if x position < 240 then
move 1 steps

direction

wait until mouse down?
set pen color to magenta
pen down
forever
go to mouse-pointer

Δημιουργώντας τις Μεταβλητές σας...

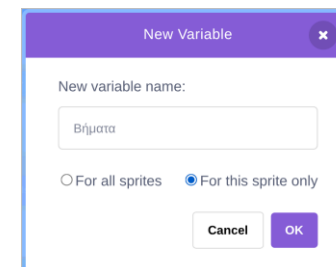
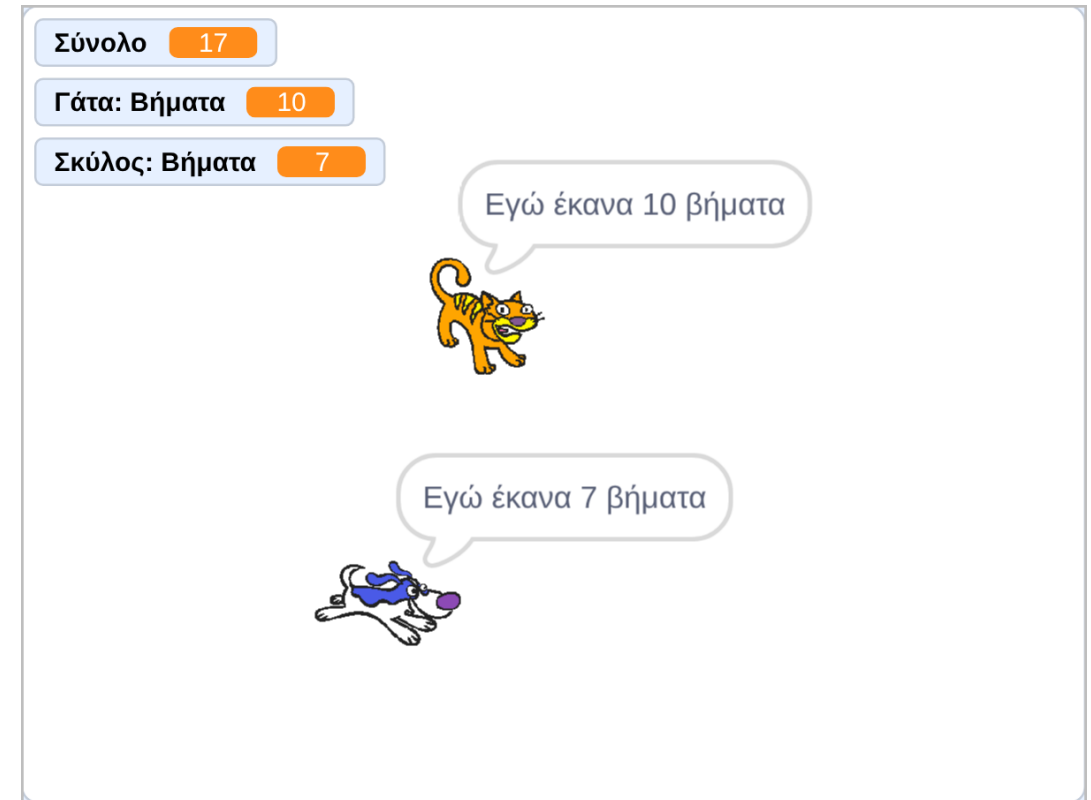
- Δημιουργήστε μία μεταβλητή όταν **θέλετε να παρακολουθήσετε κάτι...**
- Οι μεταβλητές έχουν διάφορα **χαρακτηριστικά**:
 - Κάθε μεταβλητή έχει **όνομα**.
 - Μία μεταβλητή περιέχει **μία τιμή**.
 - Το **περιεχόμενο** μίας μεταβλητής μπορεί να **αλλάξει**.
 - Μία μεταβλητή **συσχετίζεται** με **ένα** (τοπική/**local variable**), ή με **όλα** τα **sprites** (καθολική/**global variable**).
 - Μία μεταβλητή μπορεί να **φαίνεται** (π.χ. το σκορ), ή να **μη φαίνεται** στη σκηνή.



Variables → Make a Variable:
όνομα και «για όλα τα sprites»
ή «μόνο για αυτό το sprite»

Εμβέλεια: Τοπικές και Καθολικές μεταβλητές

- **Καθολική (global)** – «For all sprites»: υπάρχει **μία** για όλο το πρόγραμμα; τη βλέπουν και την αλλάζουν **όλα** τα sprites και η σκηνή (π.χ. Σύνολο).
- **Τοπική (local)** – «For this sprite only»: ανήκει σε **ένα** sprite. Δύο sprites μπορούν να έχουν τοπικές μεταβλητές με το **ίδιο όνομα** που δεν «μπερδεύονται» (Γάτα: Βήματα, Σκύλος: Βήματα).
- **Κανόνας**: κάντε μια μεταβλητή **όσο πιο τοπική** γίνεται – έτσι κανένα άλλο sprite δεν μπορεί να την αλλάξει κατά λάθος.
- Η ίδια ιδέα λέγεται **εμβέλεια (scope)** και υπάρχει και σε JavaScript και Python: μια μεταβλητή μέσα σε μια συνάρτηση είναι τοπική.

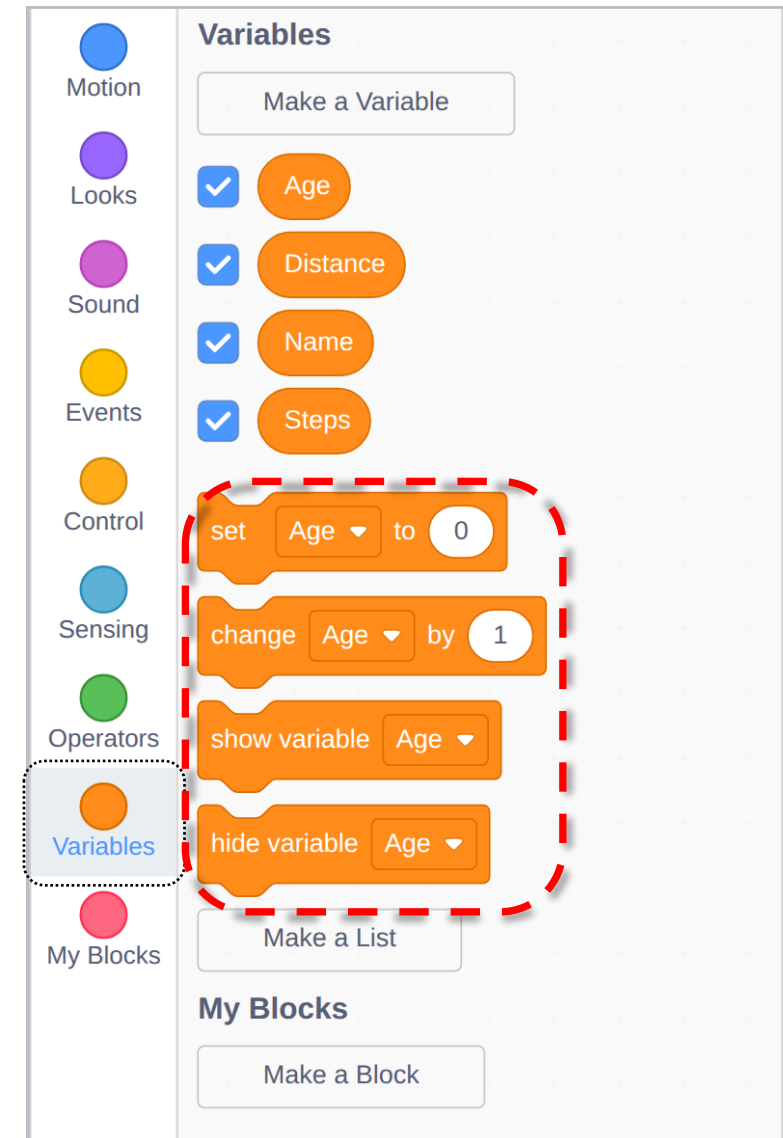


Στη σκηνή οι τοπικές φαίνονται ως «Γάτα: Βήματα». Γάτα 10 + Σκύλος 7 = Σύνολο 17.

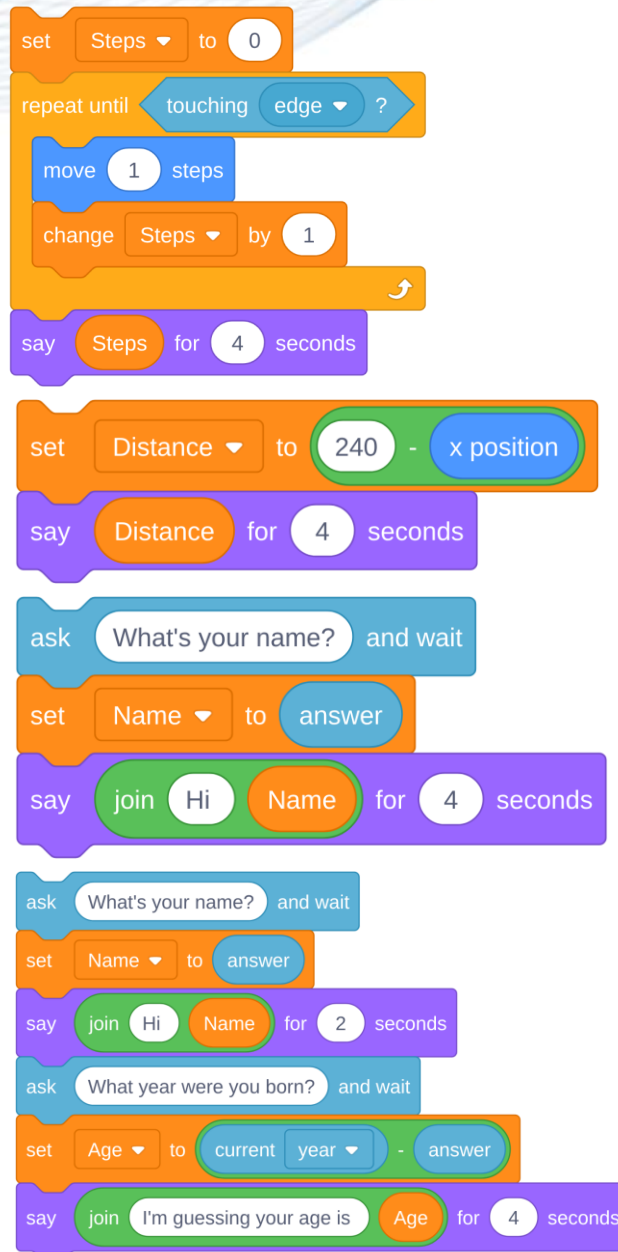
[Demos_2026-2027/lecture2_09_scope.sb3](#)

Ένα μικρό demo...

- Δημιουργώντας μεταβλητές:
 - Πόσα **βήματα** πρέπει να γίνουν έως η Scratch να αγγίξει τη γωνία;
 - Ποια είναι η **απόσταση** από τη δεξιά πλευρά;
 - Ποιο είναι το **όνομα** του χρήστη (επώνυμο, όνομα);
 - Ποιο είναι το **όνομα** και η **χρονιά** που γεννήθηκε ο χρήστης;
 - Υπολογισμός της ηλικίας



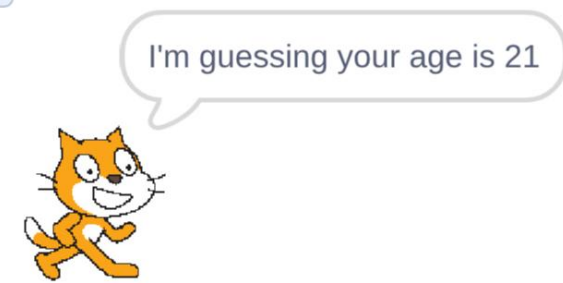
Στην πράξη...



Name Joe
Age 21
Sprite1: Steps 150
Sprite1: Distance 256.919768

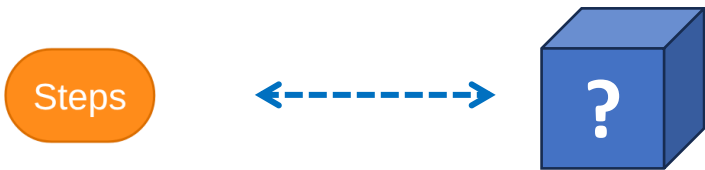


Name Joe
Age 21
Sprite1: Steps 242
Sprite1: Distance 256.919768

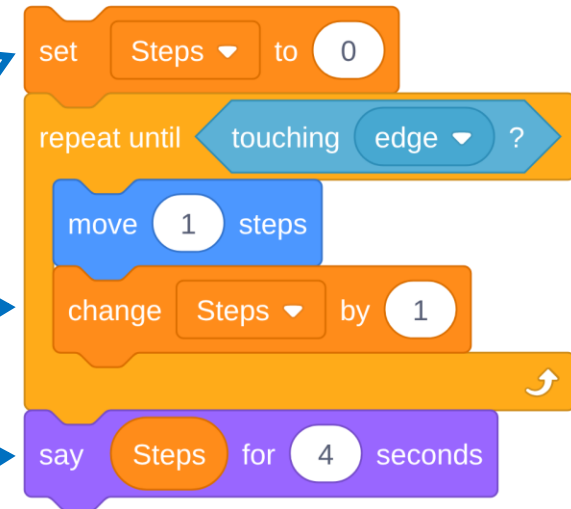


Όταν χρησιμοποιείτε λοιπόν μεταβλητές...

- Να σκέφτεστε τη μεταβλητή σαν ένα **κουτί χωρίς αρχική τιμή**:



- Θα πρέπει να **δώσετε** μία **αρχική τιμή**,
- καθώς το πρόγραμμα τρέχει να **ενημερώνετε** την τιμή αυτή
- και στο τέλος να τη **χρησιμοποιείτε**, ή να την **παρουσιάζετε**.

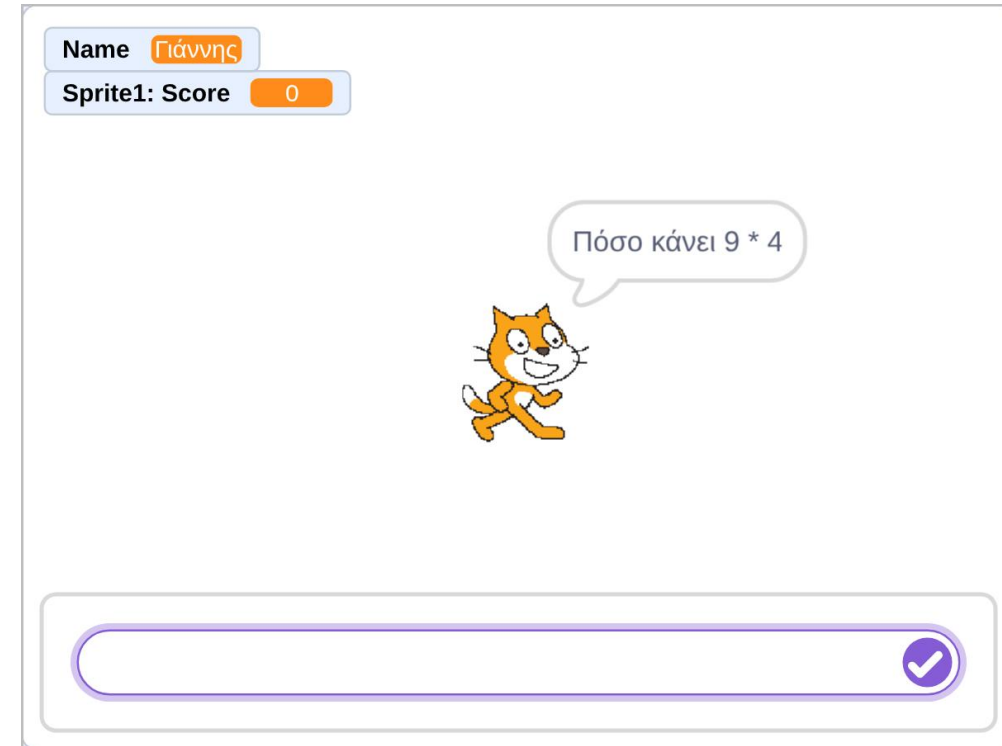


Σημ.: το «κουτί» είναι μια απλοποίηση· στην Python θα δούμε ότι μια μεταβλητή μοιάζει περισσότερο με «ετικέτα» που δείχνει σε μια τιμή.

Χρυσός Κανόνας: Χρησιμοποιείτε μία μεταβλητή για κάθε τμήμα δεδομένων που θα πρέπει να παρακολουθείτε!

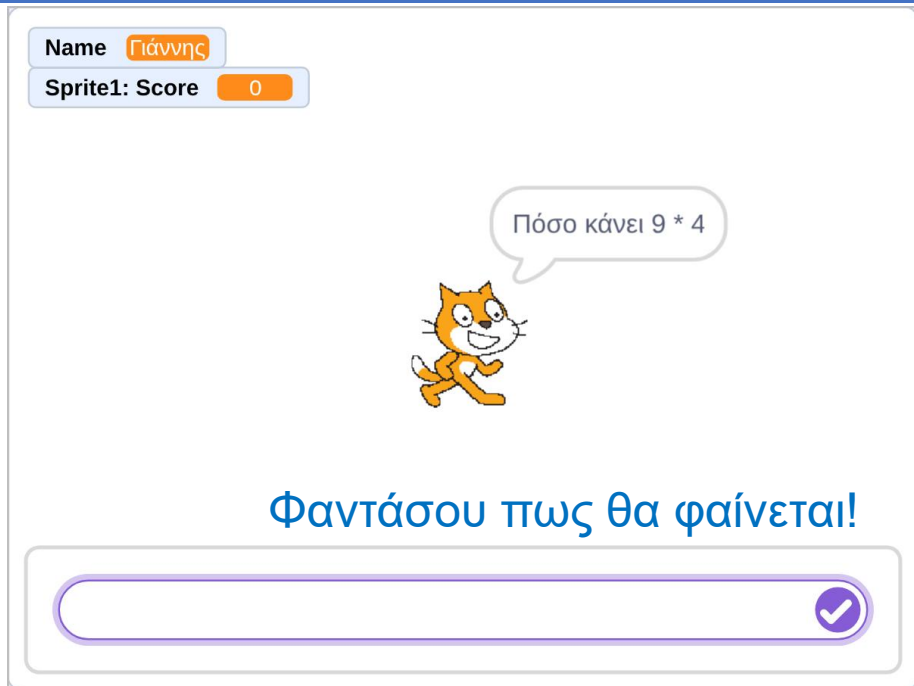
Ένα ολοκληρωμένο παράδειγμα...

- Ας χρησιμοποιήσουμε όλα όσα μάθαμε μέχρι στιγμής για να φτιάξουμε ένα απλό **παιχνίδι πολλαπλασιασμού**:
- Το πρόγραμμα θα κάνει τα ακόλουθα:
 - Επιλέγει δύο τυχαίους αριθμούς X και Y
 - Υπολογίζει το σωστό αποτέλεσμα " $X * Y$ "
 - Στη συνέχεια ρωτάει τον παίκτη:
"Πόσο κάνει $X * Y$;"
 - Συγκρίνει την απάντηση με το σωστό αποτέλεσμα
 - Αν είναι σωστή η απάντηση, τότε ο παίκτης παίρνει ένα βαθμό
 - Επανάληψη...
 - Όταν το παιχνίδι τελειώνει εμφανίζεται το σκορ

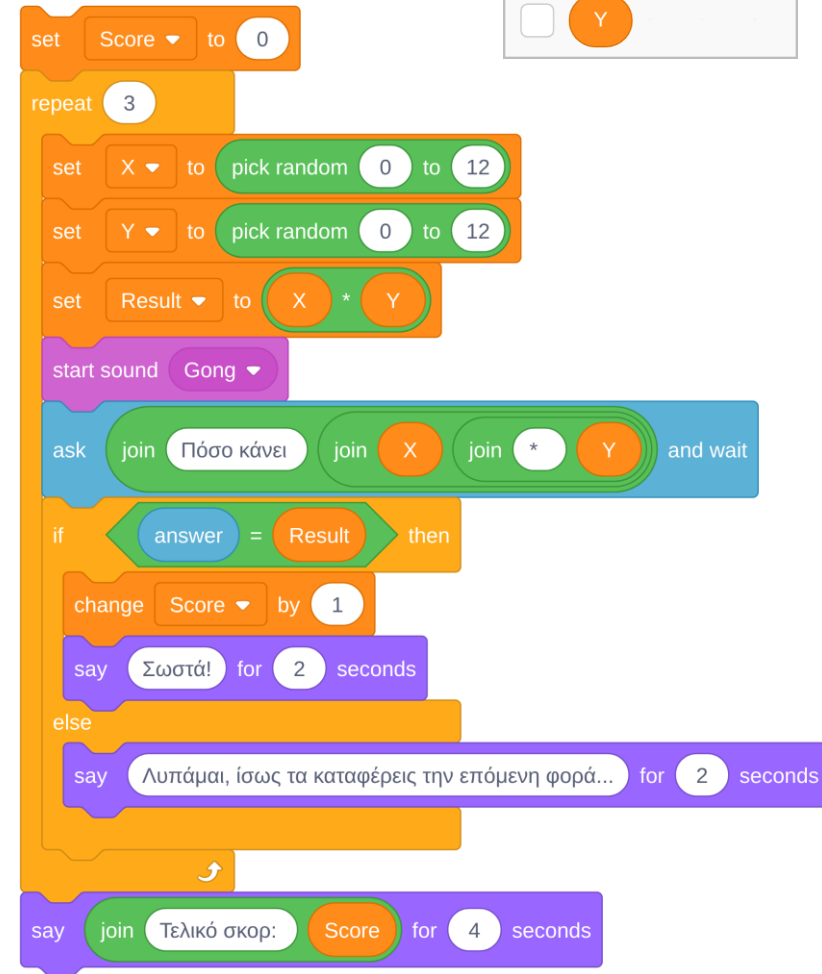


Σχεδιάζοντας τη λύση...

- Επανέλαβε 3 φορές:
 - Επέλεξε δύο τυχαίους αριθμούς, και ονόμασέ τους X και Y
 - Υπολόγισε το σωστό αποτέλεσμα " $= X * Y$ "
 - Ρώτα τον παίκτη: "Πόσο κάνει $X * Y$;"
 - Σύγκρινε την απάντηση με το σωστό αποτέλεσμα
 - Αν είναι σωστή η απάντηση, τότε ο παίκτης παίρνει ένα βαθμό
- Όταν το παιχνίδι τελιώνει εμφανίζεται το σκορ



- Πόσες μεταβλητές χρειαζόμαστε;
 - Τουλάχιστον τέσσερις
 - X, Y, Σωστό Αποτέλεσμα, Σκορ
- Η Αλληλουχία:



Συνολικά

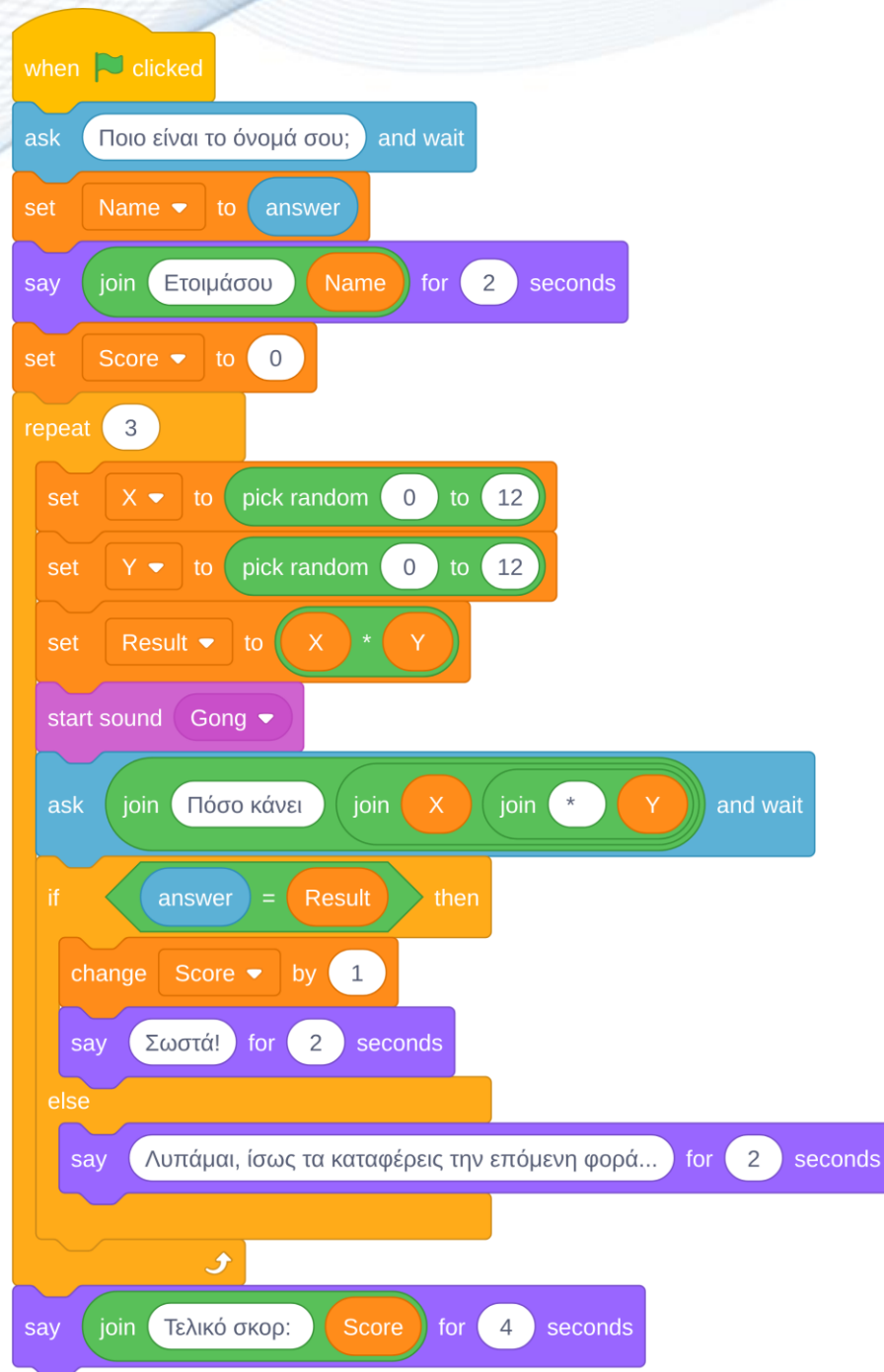
☒ Name

☐ Result

☒ Score

☐ X

☐ Y



Name Γιάννης

Sprite1: Score 0

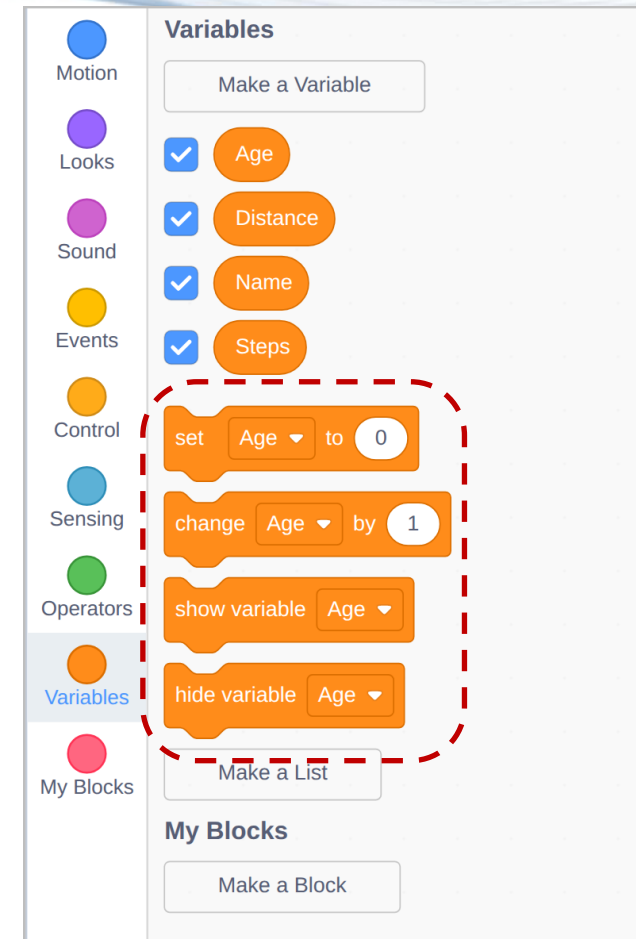
Πόσο κάνει $9 * 4$

☒

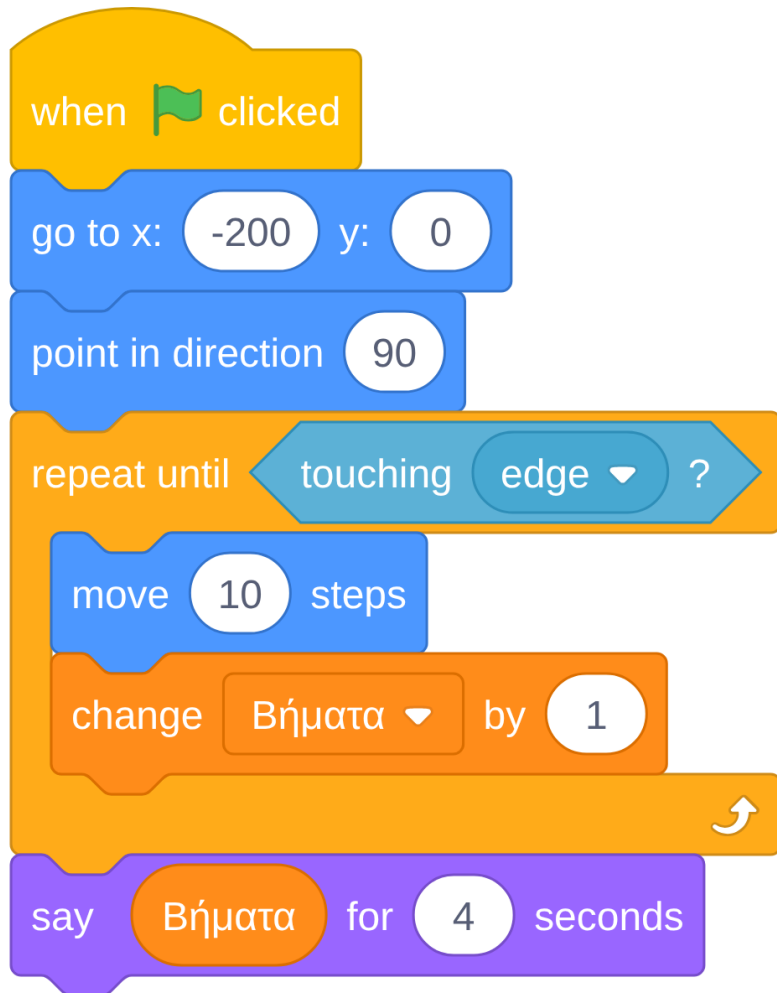
Demos_2026-2027/lecture2_02.sb3

Τι είδαμε μέχρι τώρα;

- Οι **μεταβλητές** είναι η μνήμη του προγράμματος:
 - Αποθηκεύουν πληροφορίες ενώ το πρόγραμμα «τρέχει».
 - Στο **Scratch**, η αποθήκευση του project σώζει και τις τρέχουσες τιμές τους.
- **Σημαντικός κανόνας:**
 - Χρησιμοποιείτε **μία μεταβλητή για κάθε τμήμα δεδομένων** που θα πρέπει να **παρακολουθείτε**!
- **Μην ξεχνάτε:**
 - **Αρχικοποιείτε** τις μεταβλητές πριν την πρώτη χρήση τους...



Σκεφτείτε πριν τρέξετε! – Αρχικοποίηση



Το script **δεν** δίνει αρχική τιμή στη μεταβλητή Βήματα.

Την 1η φορά η γάτα λέει **41**. Τι θα πει τη **2η** φορά που θα πατήσουμε τη σημαία;

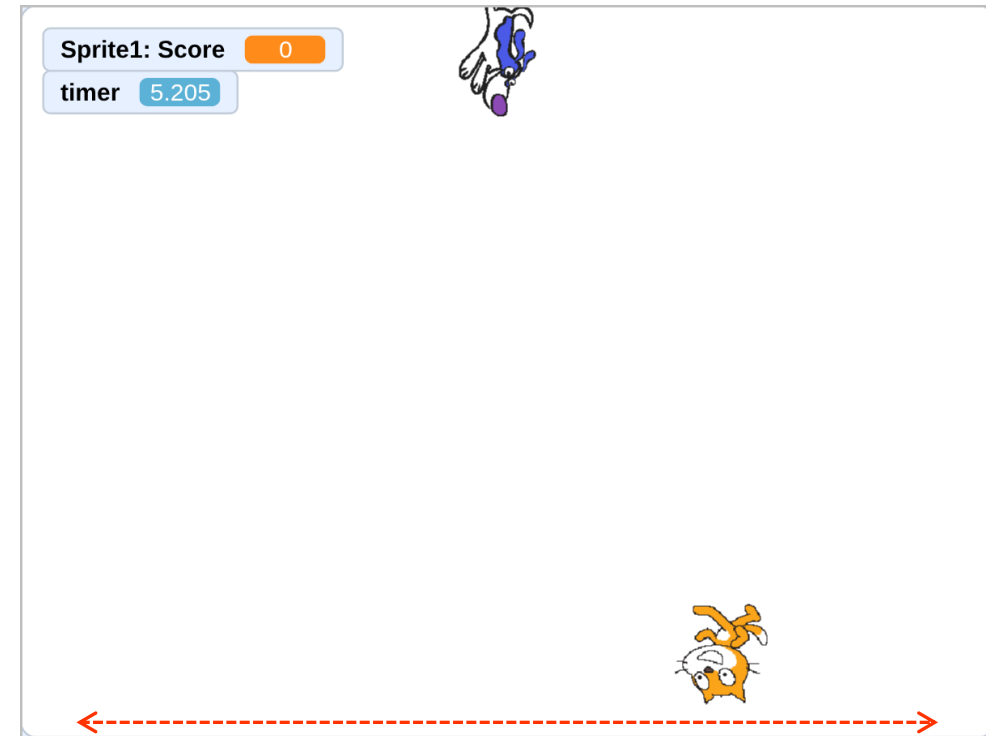
82! Η Βήματα συνεχίζει από την προηγούμενη τιμή της $(41 + 41)$. την 3η φορά θα πει 123.

Λύση: **set Βήματα to 0** στην αρχή του script.

Σε Python μια μεταβλητή χωρίς αρχική τιμή δίνει **σφάλμα** (NameError)· σε JavaScript η πράξη δίνει **NaN**.

Και λίγη δουλειά για το σπίτι... (1/2)

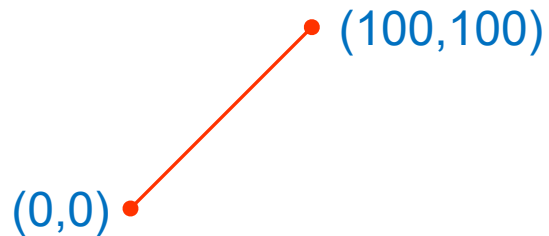
- Άσκηση 1:
 - Η πρώτη βασίζεται στην εργασία του προηγούμενου μαθήματος που φτιάξαμε ένα πρόγραμμα που έβρεχε γάτες και σκύλους.
 - Αυτή τη φορά θέλουμε η Scratch να μπορεί να μετακινηθεί αριστερά και δεξιά στο κάτω μέρος της σκηνής, προσπαθώντας να πιάσει ένα σκυλάκι το οποίο πέφτει από πάνω προς τα κάτω. Για κάθε σκυλάκι που πιάνει η Scratch θα κερδίζει ένα πόντο.



Και λίγη δουλειά για το σπίτι... (2/2)

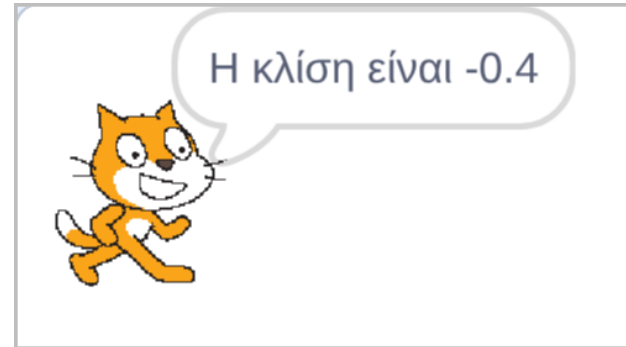
- Άσκηση 2:

- Υπολογίστε την κλίση μίας γραμμής. Η κλίση συντίθεται από δύο πράγματα:
 - Πόσο απότομη είναι και
 - Από το αν ανεβαίνει ή κατεβαίνει
- Για παράδειγμα η γραμμή αριστερά έχει κλίση 1, ενώ η γραμμή δεξιά 0.



<http://en.wikipedia.org/wiki/Slope>

- Δεδομένων δύο σημείων (x_1, y_1) και (x_2, y_2) η κλίση υπολογίζεται από τον ακόλουθο τύπο:
$$\text{slope} = (y_2 - y_1) / (x_2 - x_1)$$
- Γράψτε ένα πρόγραμμα που δέχεται σαν είσοδο τα x_1, y_1, x_2 και y_2 ζωγραφίζει μία γραμμή και υπολογίζει την κλίση της.
Προσοχή στις κάθετες γραμμές!



Προγραμματιστικές Δεξιότητες

- Σε **λογικό** επίπεδο:

- Αλληλουχία των εντολών
- Επιλογή ανάμεσα σε διάφορες εναλλακτικές
- Επανάληψη

- Σε **πρακτικό** επίπεδο:

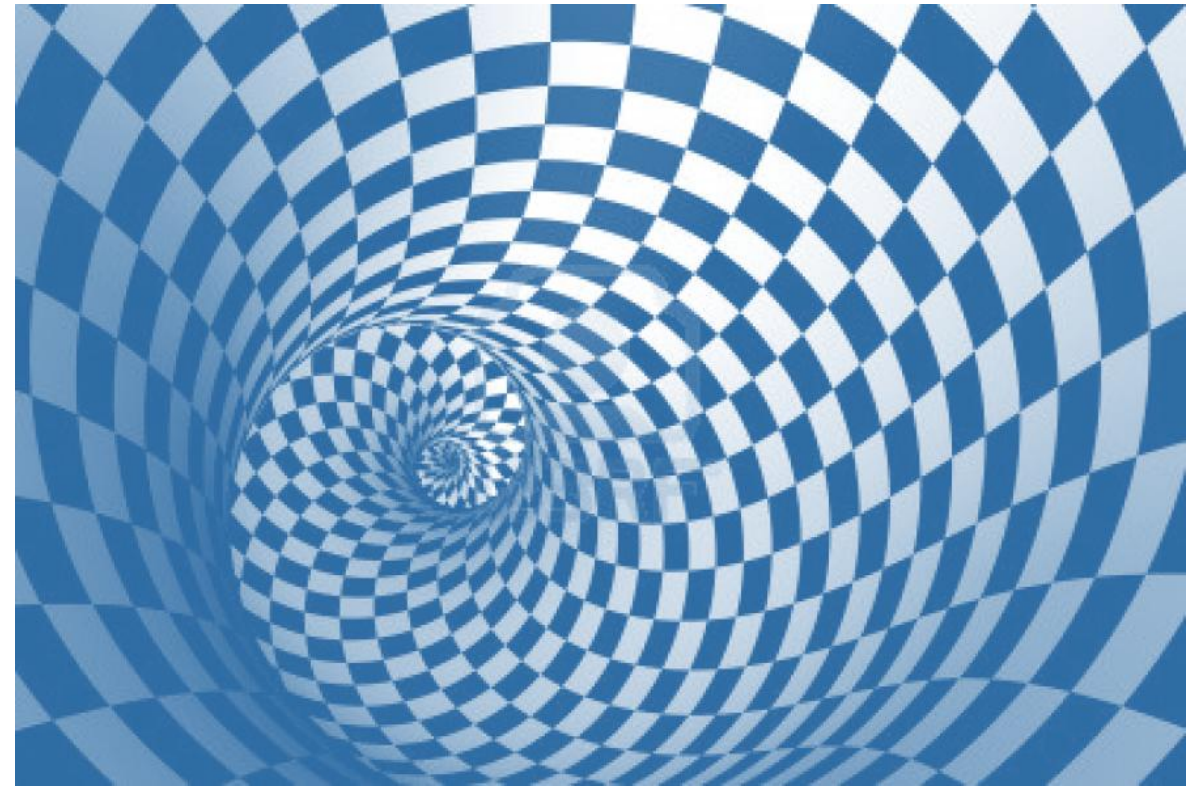
- Μεταβλητές
- Εμπειρία
- Αποφασιστικότητα
 - Οι Η/Υ μπορεί να σας δημιουργήσουν απογοήτευση, αλλά...

μην τα παρατήσετε!



Πρότυπα (patterns) = Εμπειρία (experience)

- Τα πρότυπα είναι ένας τρόπος σύλληψης της συλλογικής εμπειρίας άλλων.
 - “standing on the shoulders of giants”
- Απλά patterns:
 - commenting on your programs
 - centralized code
 - regression testing
- Μερικά πιο περίπλοκα:
 - game play
 - count, sum, and compute



#1 Σχολιασμός

- Καθώς τα προγράμματα γίνονται όλο και πιο περίπλοκα είναι πολύ δύσκολο να θυμάστε όλες τις λεπτομέρειες!
 - Μην το προσπαθήσετε καν!
- Αντί για αυτό, προσθέστε **σχόλια**
 - Τα σχόλια είναι ένας τρόπος **μόνιμης καταγραφής των σκέψεών σας**, ως μέρος του προγράμματος
 - Τα σχόλια αγνοούνται από το Scratch και από όλες τις γλώσσες προγραμματισμού – είναι φτιαγμένα **για να τα διαβάζουμε εμείς**.
 - Scratch 3: **δεξί κλικ** → Add Comment · JavaScript: **// σχόλιο** · Python: **# σχόλιο**

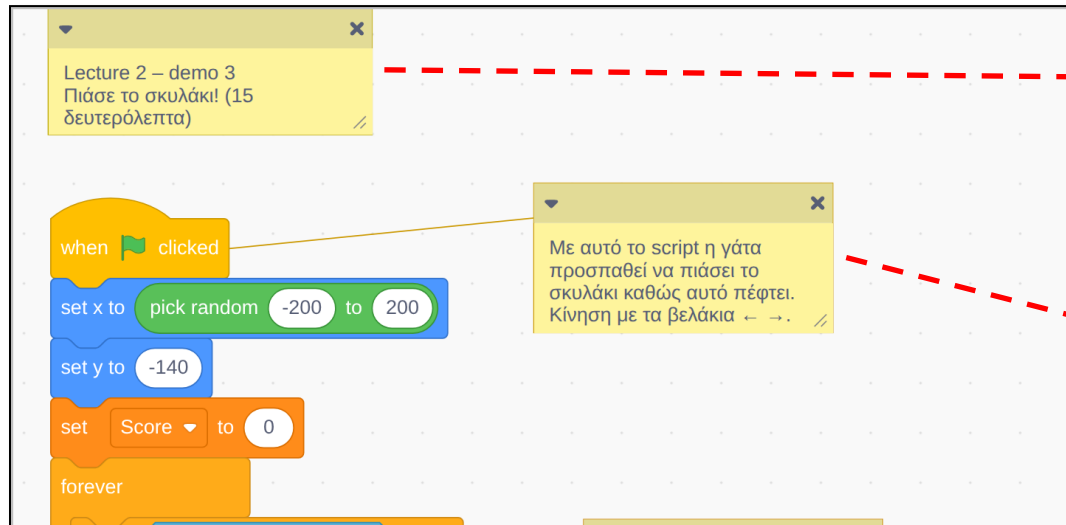
Undo

Redo

Clean up Blocks

Delete 32 Blocks

Add Comment



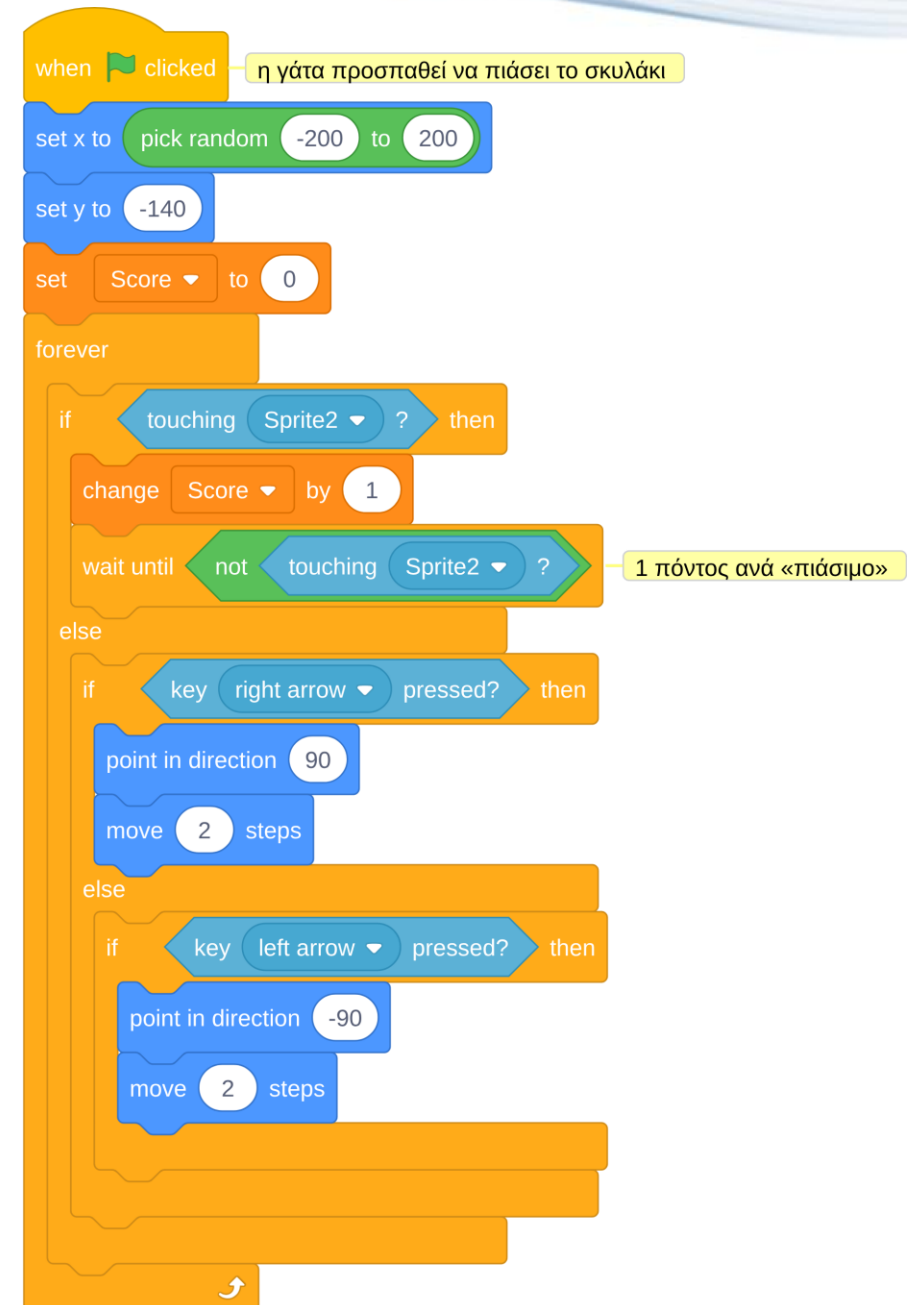
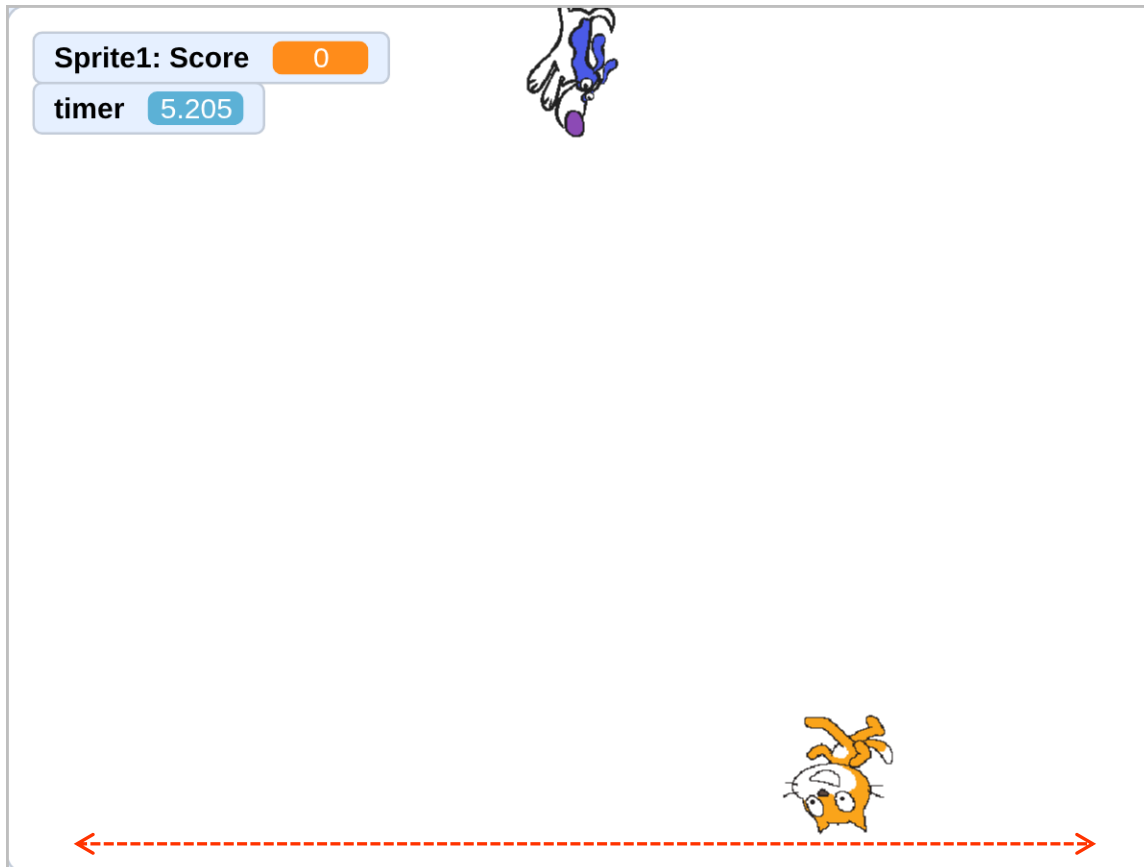
standalone

associated with a
block



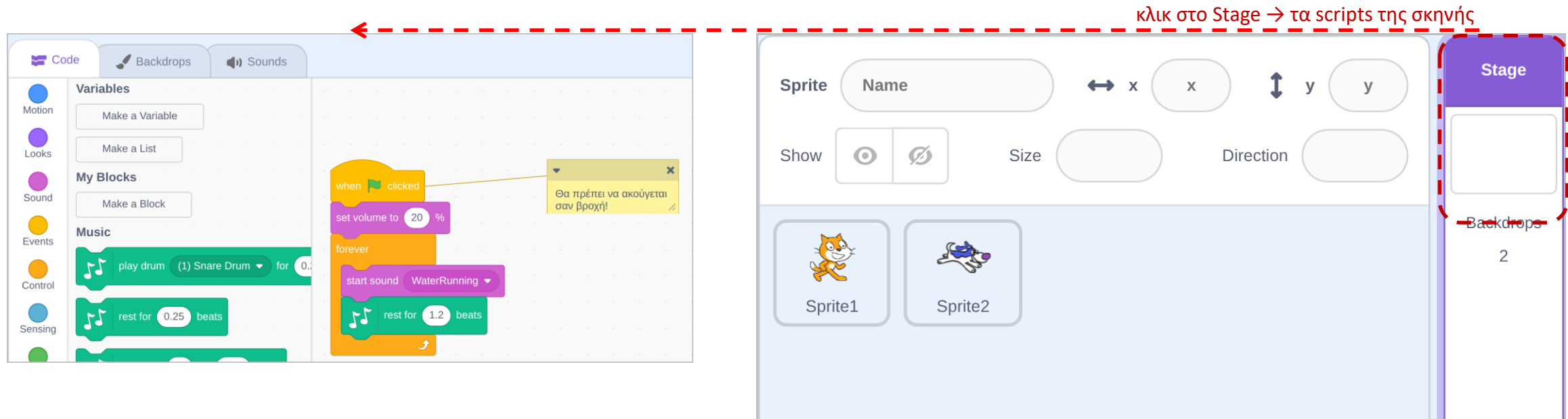
Ένα μικρό demo...

- Από την πρώτη άσκηση...



#2 Centralized Code

- Κάποιος κώδικάς μας είναι **γενικού σκοπού** – π.χ. δεν αναφέρεται σε κάποιο sprite
 - “παίξε ένα τραγούδι όσο το παιχνίδι τρέχει”
 - “παρακολουθήσε αν θα πατηθεί κάποιο κουμπί για να αλλάξει η κατάσταση του παιχνιδιού”
 - “σταμάτα το παιχνίδι μετά από Χ δευτερόλεπτα...”
- Στο Scratch, το "**stage**" (σκηνή) μπορεί επίσης να περιέχει scripts
 - Αυτό είναι ένα πολύ καλό μέρος για να μπει ο **κώδικας γενικού σκοπού**...

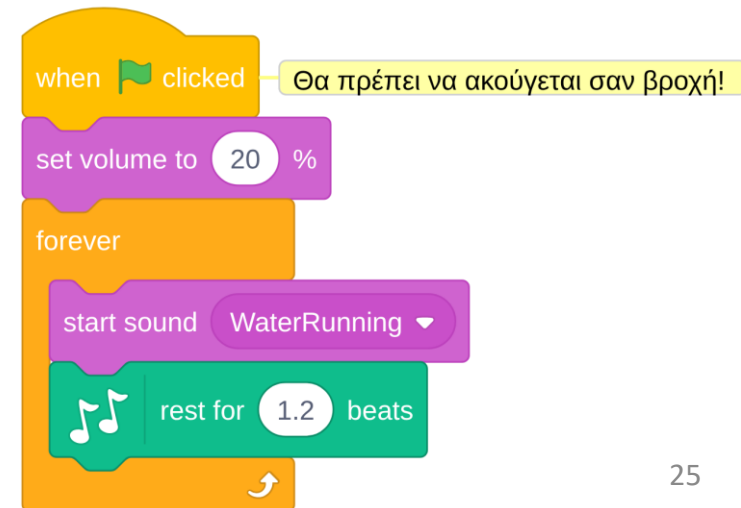
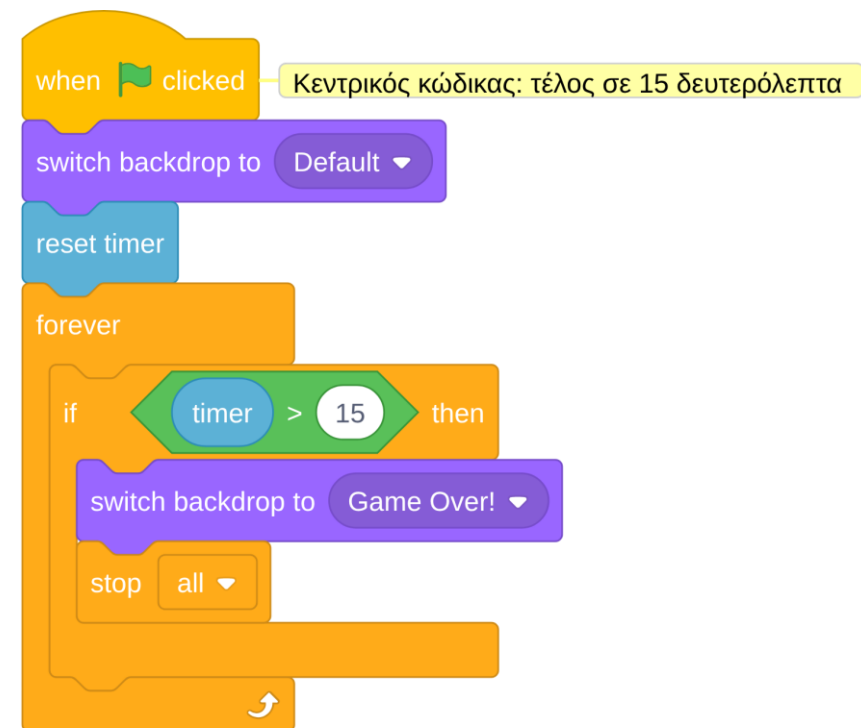
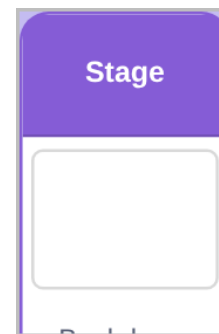


Ένα μικρό demo...

- Από την πρώτη άσκηση...
 - Ας κεντριοποιήσουμε τον κώδικα και
 - ας βάλουμε ένα χρονόμετρο.



Demos_2026-2027/lecture2_03.sb3



#3 Regression Testing

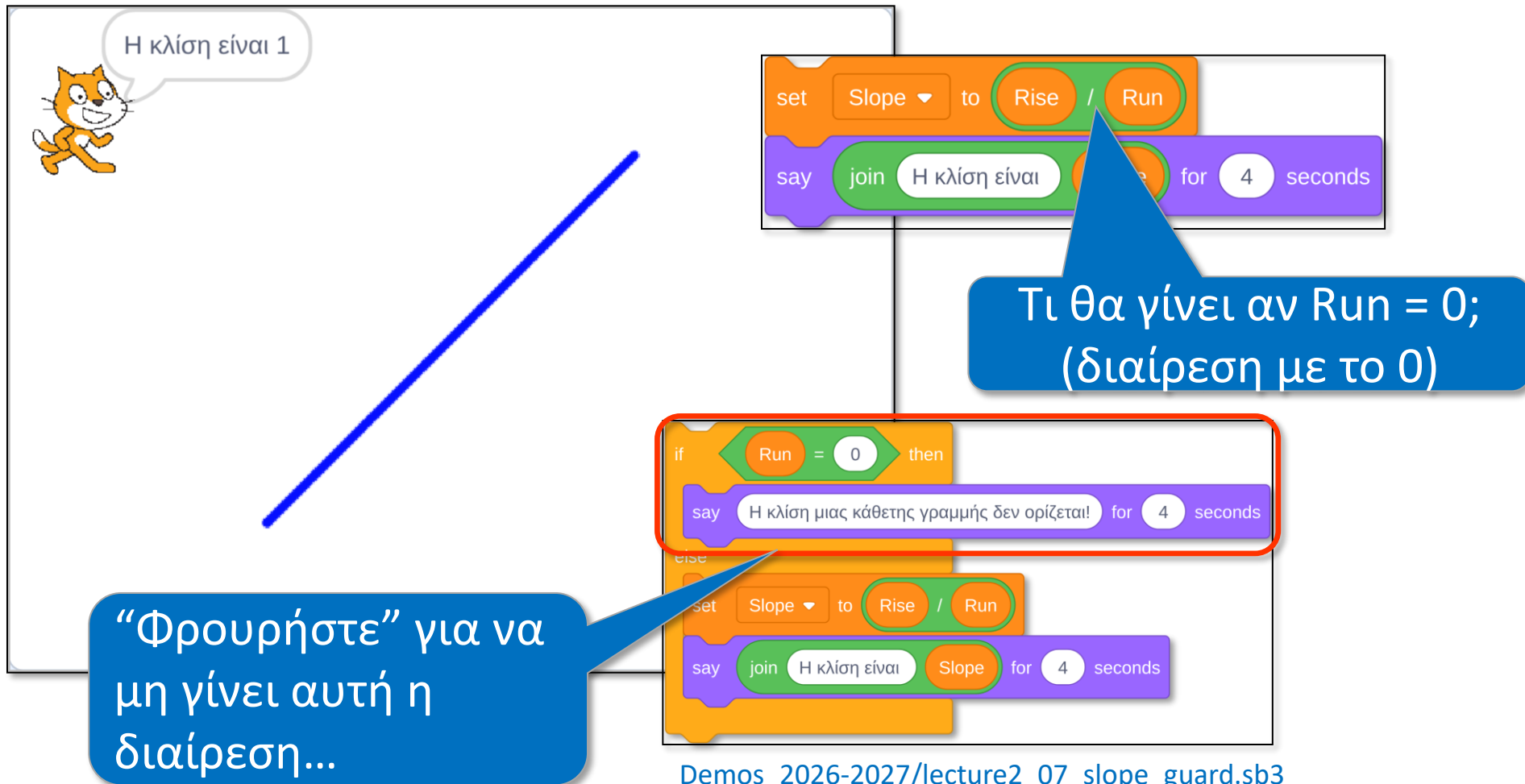
- Ο έλεγχος των προγραμμάτων είναι δύσκολος
 - Θα πρέπει να **σκεφτείτε πολύ** για όλα όσα το πρόγραμμά σας πρέπει να κάνει και να **επινοήσετε ελέγχους**, ώστε να σιγουρευτείτε ότι τα κάνει σωστά.
 - Θα πρέπει να γίνετε το ίδιο καλός/ή στο να «**σπάτε**» προγράμματα (να βρίσκετε τα λάθη τους), όσο είστε στο να τα **δημιουργείτε**.
- Έστω ότι επεξεργάζεστε και αλλάζετε το πρόγραμμά σας – τότε τι θα πρέπει να κάνετε;
 - Θα πρέπει να κάνετε **όλους τους ελέγχους ξανά**.
 - Αυτή η διαδικασία λέγεται **regression testing**.



Κανόνας: Θα πρέπει να διαθέσετε τόσο χρόνο στον έλεγχο του προγράμματος, όσο έχετε διαθέσει και στη δημιουργία του.

Ένα μικρό demo...

- Ας δούμε την άσκηση 2 με την κλίση....



The image shows a Scratch script for a "slope guard" demo. It starts with a cat character saying "Η κλίση είναι 1" (The slope is 1). The script then sets the "Slope" variable to "Rise / Run" and says "Η κλίση είναι" followed by the "Slope" variable for 4 seconds. A blue callout points to the "Run" variable in the "Slope" variable block, asking "Τι θα γίνει αν Run = 0; (διαίρεση με το 0)". Below this, an "if" block checks if "Run" is equal to 0. If true, it says "Η κλίση μιας κάθετης γραμμής δεν ορίζεται!" (The slope of a vertical line is not defined!) for 4 seconds. If false, it sets "Slope" to "Rise / Run" and says "Η κλίση είναι" followed by the "Slope" variable for 4 seconds. A blue callout points to the "if" block, saying "“Φρουρήστε” για να μη γίνει αυτή η διαίρεση..." (Guard against this division...). The file name "Demos_2026-2027/lecture2_07_slope_guard.sb3" is at the bottom.

Η κλίση είναι 1

set Slope to Rise / Run

say join Η κλίση είναι Slope for 4 seconds

Τι θα γίνει αν Run = 0;
(διαίρεση με το 0)

if Run = 0 then

say Η κλίση μιας κάθετης γραμμής δεν ορίζεται! for 4 seconds

else

set Slope to Rise / Run

say join Η κλίση είναι Slope for 4 seconds

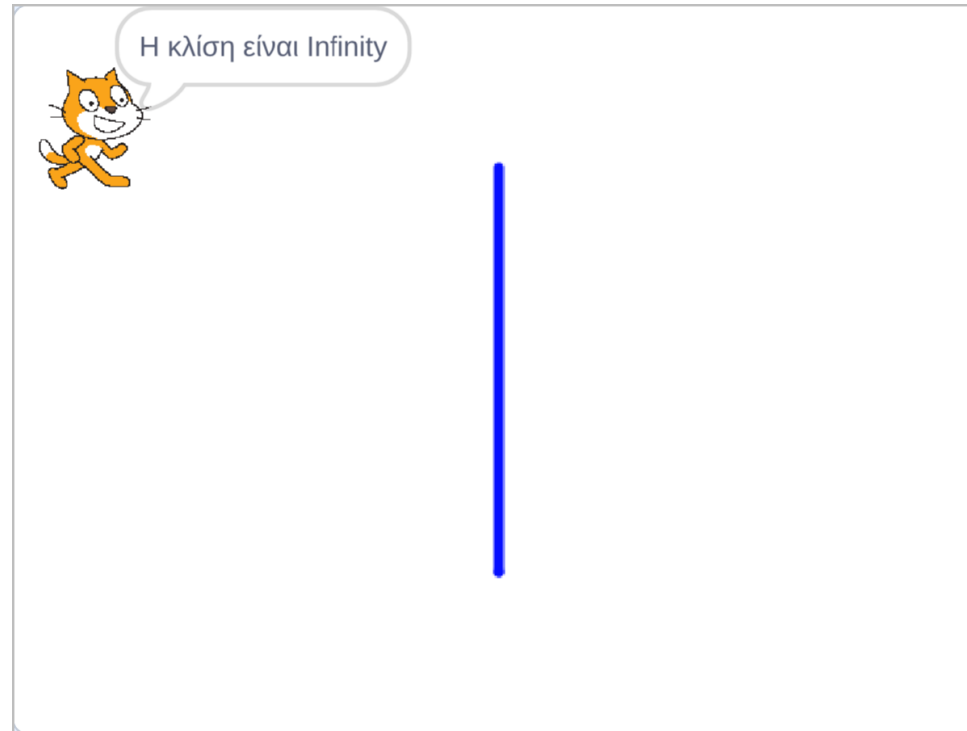
“Φρουρήστε” για να μη γίνει αυτή η διαίρεση...

Demos_2026-2027/lecture2_07_slope_guard.sb3

Αν δεν κάνουμε τον έλεγχο...



Demos_2026-2027/lecture2_04.sb3



Στο Scratch 3 η διαίρεση με το 0 **δεν σταματά** το πρόγραμμα: δίνει **Infinity** (άπειρο).
Λάθος αποτέλεσμα, χωρίς καμία προειδοποίηση!

Διαίρεση με το 0 σε τρεις γλώσσες

Scratch

100 / 0 → Infinity



JavaScript

```
console.log(100 / 0);  
// Infinity
```

```
if (run === 0) {  
    console.log("Δεν ορίζεται!");  
} else {  
    slope = rise / run;  
}
```

Python

```
print(100 / 0)  
# ZeroDivisionError:  
#   division by zero
```

```
if run == 0:  
    print("Δεν ορίζεται!")  
else:  
    slope = rise / run
```

Κάθε γλώσσα «αντιδρά» αλλιώς: η Scratch και η JavaScript δίνουν **Infinity**, η Python σταματά με σφάλμα.

Ο έλεγχος πριν από τη διαίρεση («φρουρός») δουλεύει παντού!

#4 Game Play

- Μία από τις προκλήσεις των παιχνιδιών είναι ο **συντονισμός** όλων των sprites.
 - Όταν **αλλάζουμε επίπεδα**, πώς το ξέρουν όλα τα sprites;
 - Όταν **τελειώνει το παιχνίδι**, πώς σταματάνε όλα τα sprites;
 - Πρέπει κάθε sprite να ξέρει **πώς τελειώνει το παιχνίδι**, ή μπορούμε αυτό να το κεντριοποιήσουμε;
 - Ο κώδικας που υπάρχει κεντρικά μπορεί να αλλάξει ευκολότερα στο μέλλον...



Λύση (Message Passing):

- ❖ χρησιμοποιήστε ένα **broadcast block** (στην κατηγορία *Events*) για να στείλετε ένα μήνυμα όταν κάτι συμβαίνει
- ❖ τα sprites χρησιμοποιούν ένα **when I receive block** (επίσης στην κατηγορία *Events*) για να αντιδράσουν σε ένα μήνυμα

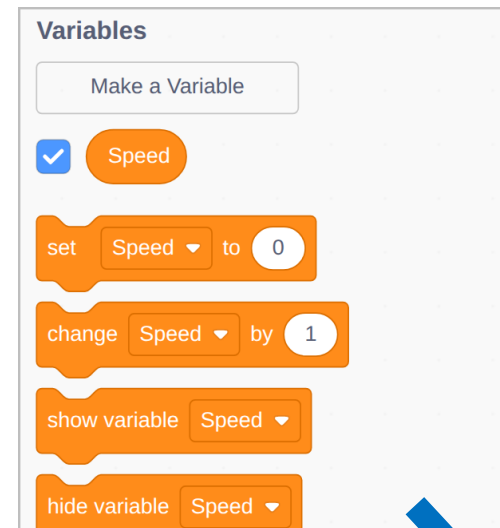
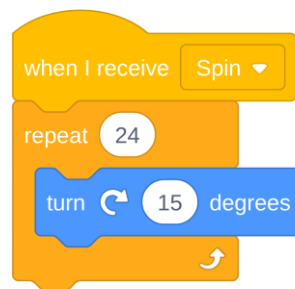
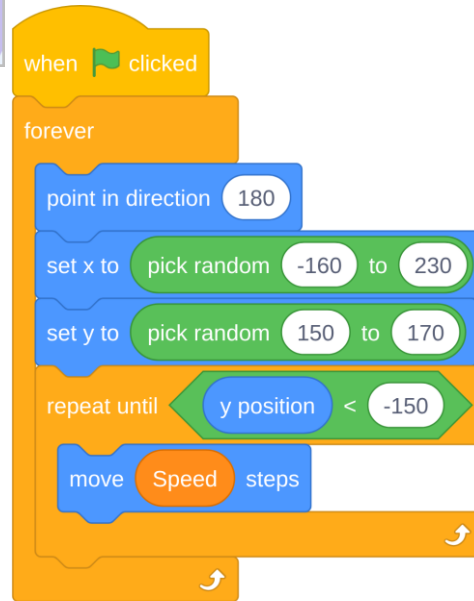
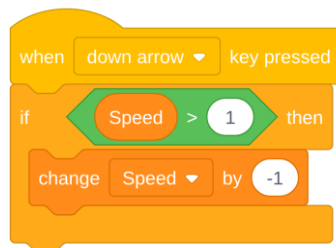
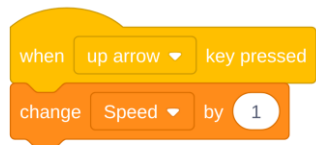
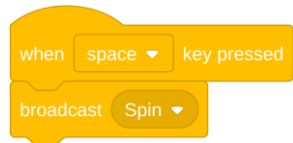
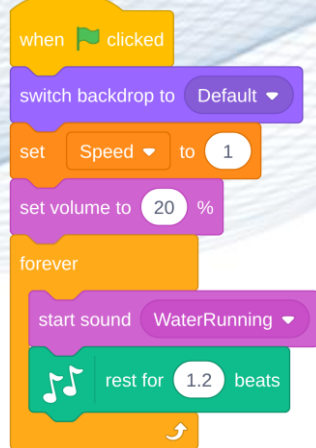
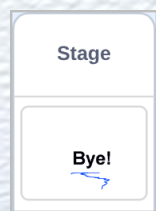
Το broadcast είναι ένα **γεγονός (event)** που το δημιουργούμε **εμείς** – όπως η πράσινη σημαία ή ένα πλήκτρο.

Ένα μικρό demo...

- Θα χρησιμοποιήσουμε το παράδειγμα με την καταιγίδα ("Raining Cats and Dogs") και θα:
 - Βάλουμε ένα "Game Over!" μήνυμα στο τέλος
 - Βάλουμε ένα "Spin" μήνυμα για να περιστρέφουμε τις γάτες και τους σκύλους
 - Βάλουμε κουμπιά αλλαγής ταχύτητας...



Στην πράξη...



Demos_2026-2027/lecture2_05.sb3



#5 Count, Sum, & Compute

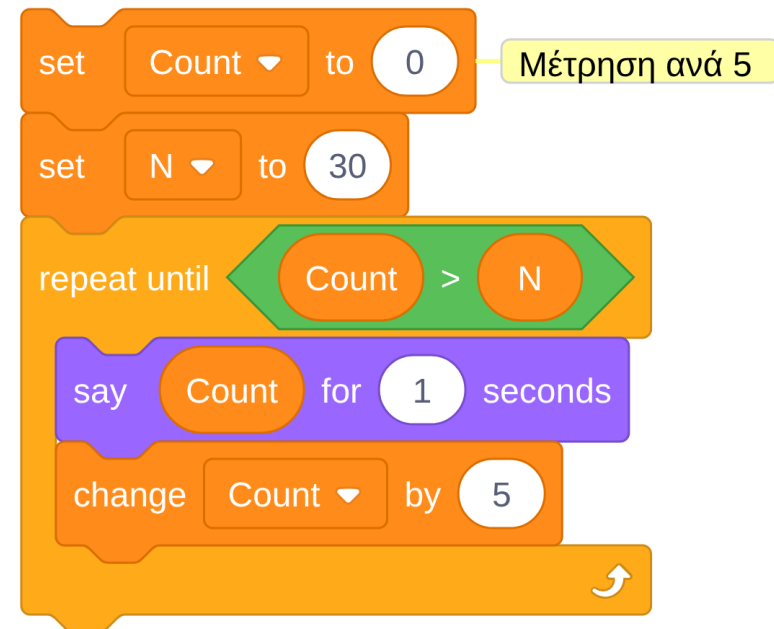
- Το Scratch είναι περισσότερο από μία γλώσσα δημιουργίας παιχνιδιών
 - Το Scratch είναι ικανό για **γενικού σκοπού υπολογισμούς**
 - Παράδειγμα: “Κλίση” το παράδειγμα που κάναμε
- **Γενικά πρότυπα υπολογισμών:**
 - **μέτρηση-counting:** 1, 2, 3, ..., N
 - **άθροιση-summing:** $1 + 2 + 3 + \dots + N$
 - **υπολογισμοί-computing:**
πρόσθεση, πολλαπλασιασμός,
τετραγωνική ρίζα, ημίτονο,
συνημίτονο, ...



Μέτρηση – Counting

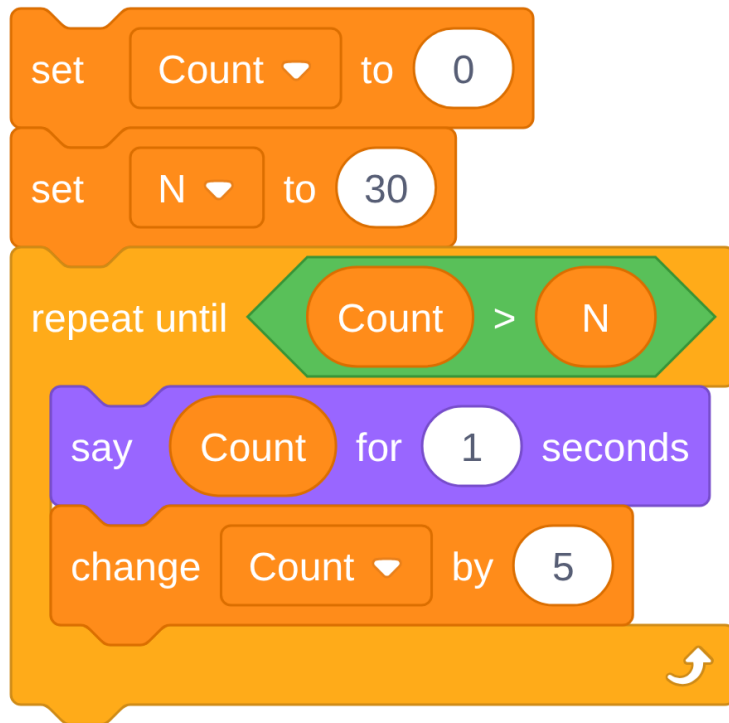
- Υποθέστε ότι θέλετε να μετρήσετε από 0 .. N ανά 5
 - 0, 5, 10, 15, 20, 25, ...

- Δημιούργησε μία μεταβλητή "Count"
- Δημιούργησε μία μεταβλητή "N"
- Θέσε την Count σε 0
- Θέσε την N στην τελευταία τιμή
- Repeat until Count > N:
 - Κάνε τους υπολογισμούς με την Count
 - Άλλαξε την Count κατά 5



Demos_2026-2027/lecture2_06.sb3

Σκεφτείτε πριν τρέξετε! – Μέτρηση



[Demos_2026-2027/lecture2_06.sb3](#)

JavaScript: `for (let c = 0; c <= 30; c += 5)`

Python: `for c in range(0, 31, 5)`
← το 31 δεν περιλαμβάνεται!

Ποιος είναι ο **τελευταίος** αριθμός που θα πει η Scratch;

Πόσους αριθμούς θα πει **συνολικά**;

Το **30** και συνολικά **7** αριθμούς: 0, 5, 10, ..., 30. Ο βρόχος σταματά μόνο όταν $\text{Count} = 35 > 30$.

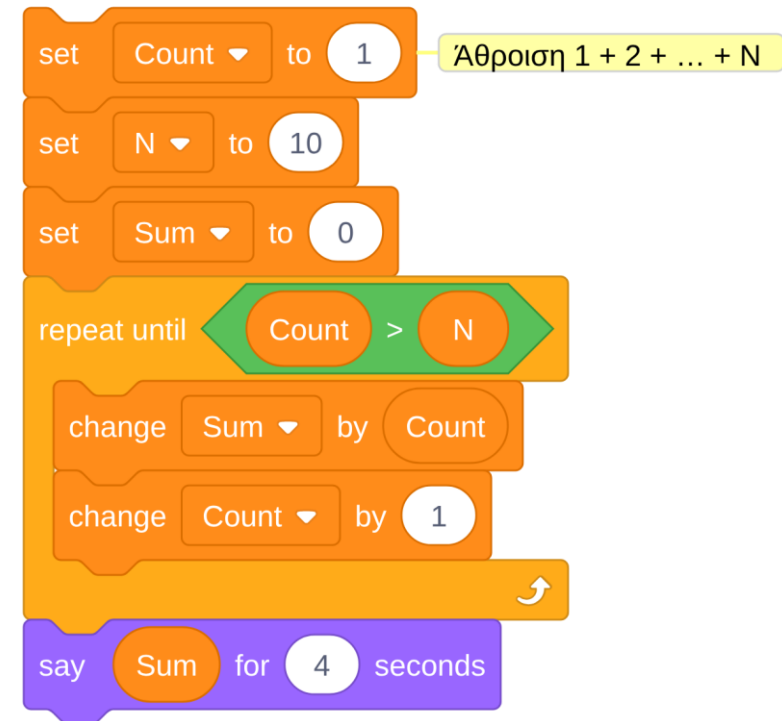
Με $\text{Count} = \text{N}$ θα σταματούσε **πριν** πει το 30!

Τα όρια ($>$ ή $\geq$, $<$ ή $\leq$) είναι από τις πιο συχνές πηγές λαθών («off-by-one»).

Άθροιση- Summing

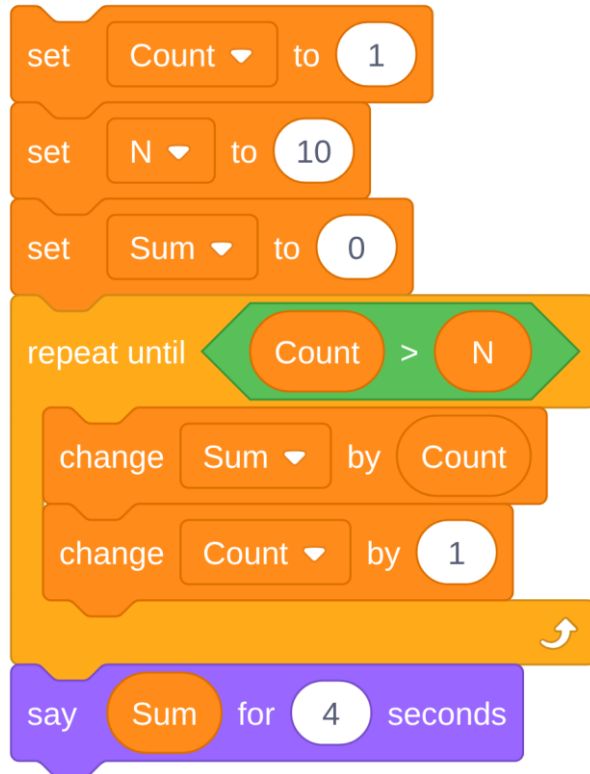
- Υποθέστε ότι θέλετε να αθροίσετε τις τιμές 1 .. N
 - $1 + 2 + 3 + 4 + 5 + \dots + N$

- Δημιούργησε μία μεταβλητή "Count"
- Δημιούργησε μία μεταβλητή "N"
- Δημιούργησε μία μεταβλητή "Sum"
- Θέσε την Count σε 1
- Θέσε την N στην τελευταία τιμή
- Θέσε την Sum σε 0
- Repeat until Count > N:
 - Άλλαξε τη Sum κατά Count
 - Άλλαξε την Count κατά 1



Άθροιση σε τρεις γλώσσες

Scratch



JavaScript

```
let count = 1;
const n = 10;
let sum = 0;
while (count <= n) {
  sum += count;
  count += 1;
}
console.log(sum); // 55
```

Python

```
count = 1
n = 10
total = 0
while count <= n:
    total += count
    count += 1
print(total) # 55
```

repeat until <(Count) > (N)> ≡ while (count <= n)

Έλεγχος: $N \cdot (N+1) / 2 = 10 \cdot 11 / 2 = 55 \checkmark$

Υπολογισμοί – Computing

- Παράδειγμα: **Νόμοι της κίνησης του Νεύτωνα**
 - Αυτή η εξίσωση υπολογίζει το ύψος H ενός αντικειμένου, όταν το πετάμε ευθεία επάνω, χωρίς να υπολογίσουμε την αντίσταση του αέρα.

$$H = V_0 * T - 1/2g * T^2$$

ύψος

χρόνος

Αρχική
ταχύτητα

Επιτάχυνση
της βαρύτητας
(9,8 m/sec²)

Ας υπολογίσουμε το ύψος που θα φτάσει μία μπάλα σε ένα ποδοσφαιρικό αγώνα.



Νόμος της Κίνησης του Νεύτωνα

- Ας εφαρμόσουμε το νόμο:
 - Ο παίκτης πετάει τη μπάλα με περίπου 27 m/s ($\approx 98 \text{ km/h}$)
 - Ας υπολογίσουμε το ύψος με $T=1, 2, 3, 4, 5$

• Δημιούργησε τις μεταβλητές που χρειαζόμαστε:

- μέτρηση: T, N
- υπολογισμός: $H, V0, g$

• Θέσε το g σε 9.8

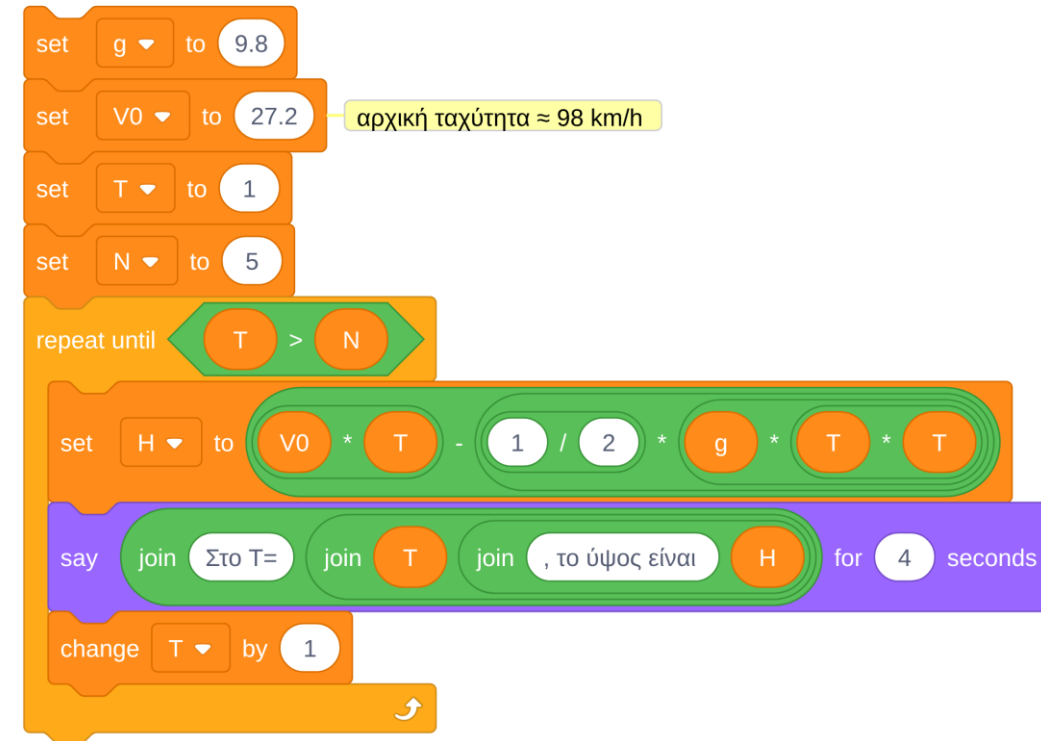
• Θέσε το $V0$ σε 27.2

• Θέσε το T σε 1

• Θέσε το N σε 5

• Repeat until $T > N$:

- Θέσε το H σε $V0 * T - \frac{1}{2} * g * T * T$
- Πες H
- Αλλάξε το T κατά 1



Η Scratch θα πει «22.299999999999997»: οι δεκαδικοί στον υπολογιστή έχουν μικρά σφάλματα στρογγυλοποίησης – το ίδιο και σε JavaScript/Python ($0.1 + 0.2 \rightarrow 0.30000000000000004$).

Demos_2026-2027/lecture2_06.sb3

Τι είδαμε μέχρι τώρα;

- Αυτό ήταν λοιπόν, **είστε έτοιμοι!** Έχετε αποκτήσει τις ακόλουθες δεξιότητες:
 - **Αλληλουχία** των εντολών
 - **Επιλογή** ανάμεσα σε διάφορες εναλλακτικές
 - Αξιοποίηση της **Επανάληψης**
 - Γνώση των **μεταβλητών** (εμβέλεια, αρχικοποίηση)
 - Έχετε δει μερικά **πρότυπα** (σχόλια, κεντρικός κώδικας, έλεγχος, μηνύματα, μέτρηση, άθροιση)
 - Απλά χρειάζεστε περισσότερη **εμπειρία** και μεγάλη **αποφασιστικότητα!!!**
- Συνεχίστε να μαθαίνετε:
 - <https://scratch.mit.edu/discuss/>
- Για περισσότερη έμπνευση:
 - <https://scratch.mit.edu/explore/projects/featured/>

Are
You
Ready?

Και λίγη δουλειά για το σπίτι...

- Υπολογίστε την πραγματική τροχιά μιας μπάλας.
- Θεωρήστε ότι ο παίκτης στέκεται στο σημείο (0,0) και πετάει τη μπάλα με θετική x και y κατεύθυνση με μία γωνία A .
- Για να υπολογιστεί η θέση της μπάλας τη χρονική στιγμή T χρησιμοποιήστε τους ακόλουθους τύπους:
 - $x_T = \cos(A) * V_0 * T$, $y_T = \sin(A) * V_0 * T - 1/2 * g * T^2$
- Ο χρήστης θα πρέπει να δώσει τη γωνία σαν είσοδο.



