Laboratorio di Pensiero Computazionale

Simone Zennaro – simone.zennaro@unipd.it

Nella lezione di oggi

- ✕ Esercizi
- ✕ Le variabili
- ✕ Le procedure e le funzioni

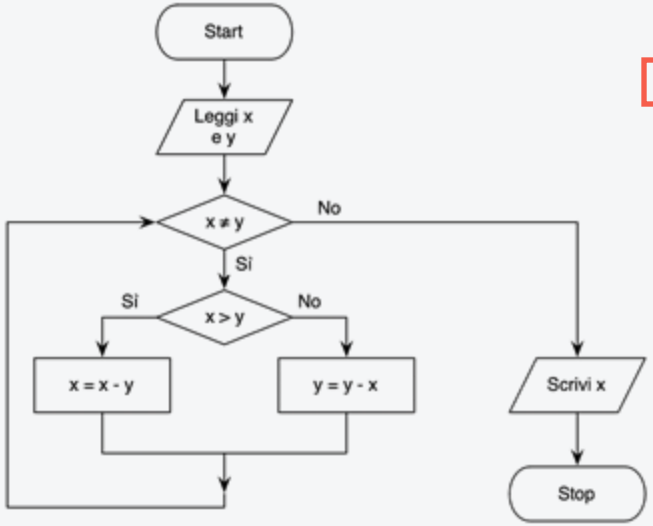
Esercizio 4

Dati due numeri interi positivi x e y calcolare il massimo comune divisore tra x e y usando somma e sottrazione.

Esercizio 4

Dati due numeri interi positivi x e y calcolare il massimo comune divisore tra x e y.

Per costruire un algoritmo elementare per calcolare il MCD (massimo comun divisore) tra x e y ancora una volta facciamo ricorso ad una divisione piuttosto elementare in modo da costruire la strategia risolutiva basandoci soltanto su operazioni aritmetiche di base. In particolare l'idea è quella di applicare una variante del cosiddetto "algoritmo di Euclide": si tratta di sottrarre ripetutamente x da y (se y > x) o y da x (se x > y) fino ad ottenere lo stesso valore per x e y; tale valore è proprio il massimo comun divisore dei due numeri originali.

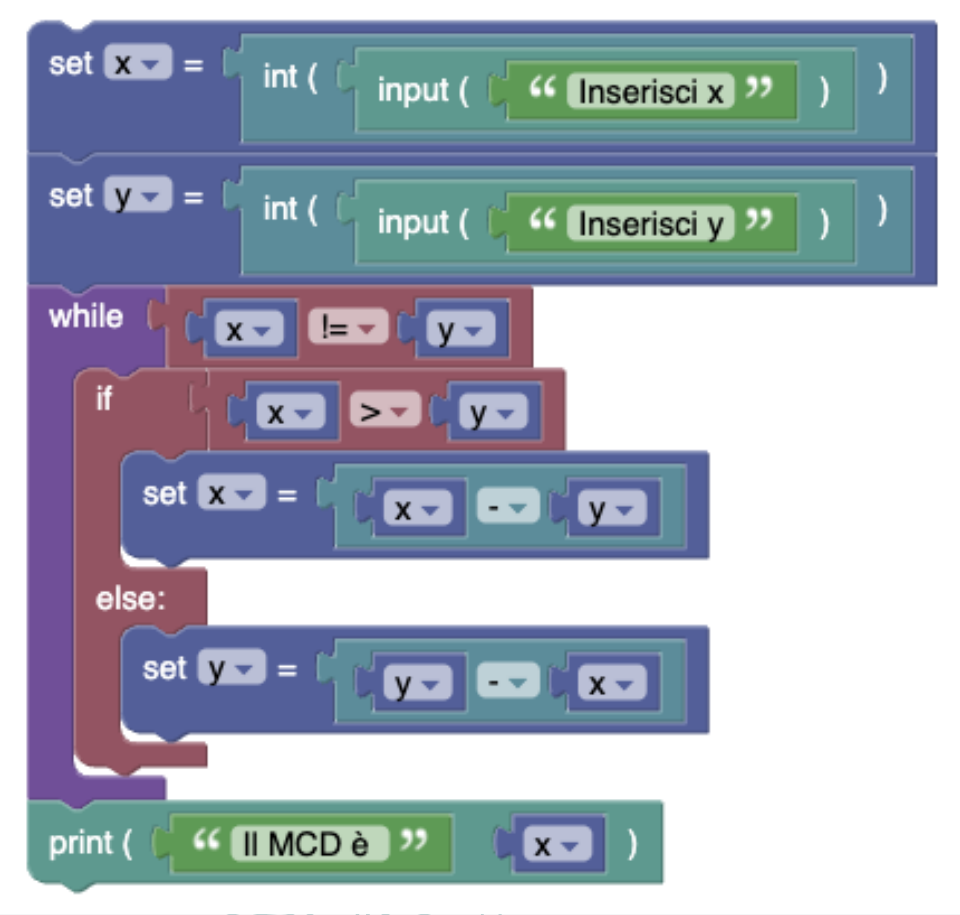



```
1: leggi x e y
3: fintanto che x≠y ripeti
4:   se x > y allora
5:     x = x – y
6:   altrimenti
7:     y = y - x
8: scrivi "il MCD è x"
9: stop
```

Esercizio 4

Dati due numeri interi positivi x e y calcolare il massimo comune divisore tra x e y.

Per costruire un algoritmo elementare per calcolare il MCD (massimo comun divisore) tra x e y ancora una volta facciamo ricorso ad una divisione piuttosto elementare in modo da costruire la strategia risolutiva basandoci soltanto su operazioni aritmetiche di base. In particolare l'idea è quella di applicare una variante del cosiddetto "algoritmo di Euclide": si tratta di sottrarre ripetutamente x da y (se y > x) o y da x (se x > y) fino ad ottenere lo stesso valore per x e y; tale valore è proprio il massimo comun divisore dei due numeri originali.



Python

```
1  x = int(input('Inserisci x: '))
2  y = int(input('Inserisci y: '))
3  while x != y:
4      if x > y:
5          x = x - y
6      else:
7          y = y - x
8  print(f"Il MCD è {x}")
9  print('Fine :)')
```

Esercitiamoci con le Variabili

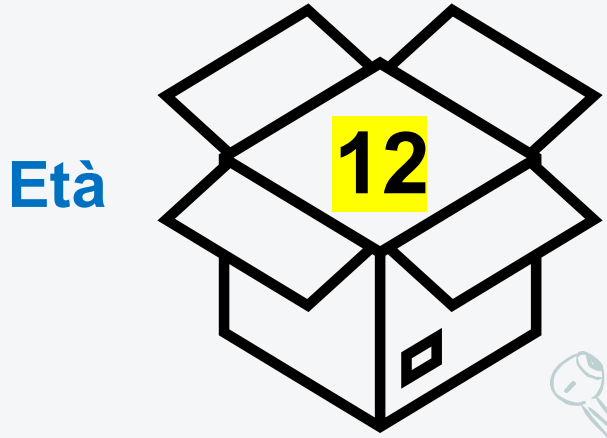
https://studio.code.org/courses/express-2025/units/1/lessons/27/levels/1

https://studio.code.org/courses/express-2025/units/1/lessons/28/levels/1

Cos'è una variabile?

Immagina una **scatola con un'etichetta**.

- L'etichetta è il **nome della variabile**.
- Dentro la scatola metti un **valore**.
- Puoi aprire la scatola, cambiare il contenuto e richiuderla.



Le variabili

Le variabili hanno:

- Un **nome** (es. età, nome, punteggio)
- Un **valore** (es. 12, “Maria”, 99)
- Possono cambiare nel tempo → **variabili = valori che variano**

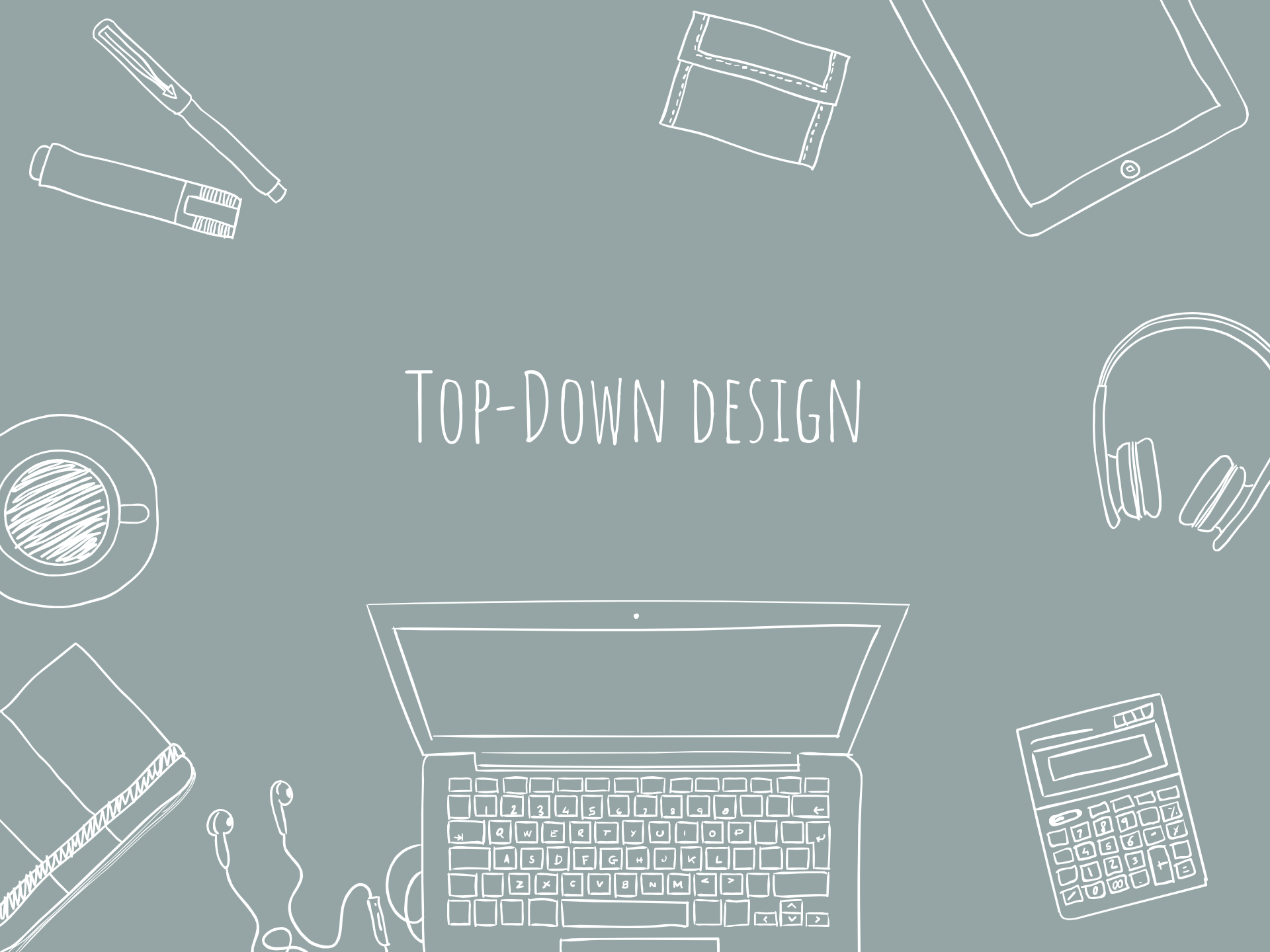
```
nome = "Giulia"
età = 12
punteggio = 0

punteggio = punteggio + 10
```

Le variabili

Perché sono utili?

- ✕ Permettono di **ricordare informazioni** durante l'esecuzione.
- ✕ Riducono gli errori → non devo riscrivere i numeri ovunque.
- ✕ Rendono i programmi più **flessibili** e **riutilizzabili**.

Top-Down design



Cos'è il Top-Down design?

Il Top-Down design è un metodo di approccio a problemi di programmazione.

Top: da una visione generale.

Down: fino al dettaglio sufficiente per programmare la soluzione.

Come?

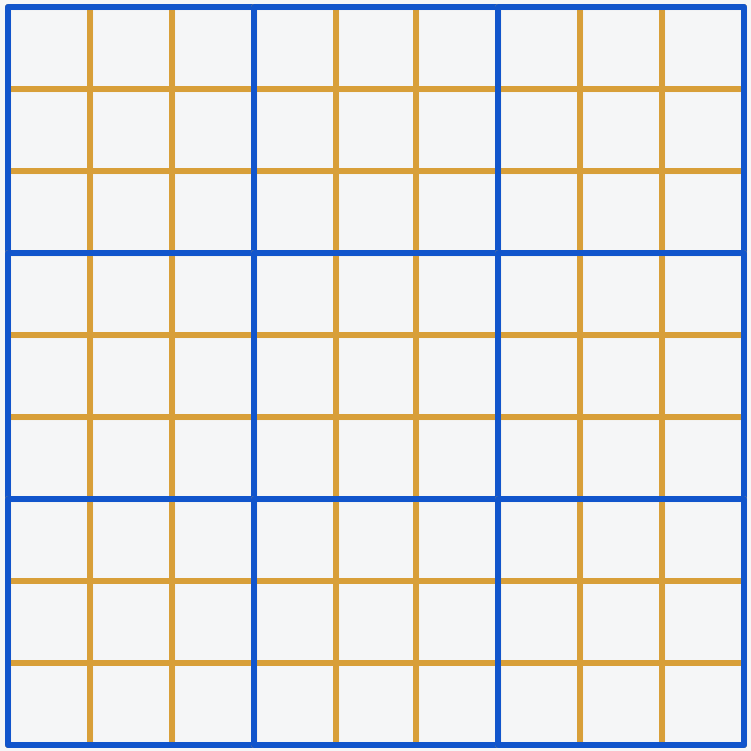
1. Dividiamo il problema in sottoproblemi
2. Dividiamo ancora i sottoproblemi
3. E ancora... e ancora... finché non arriviamo a qualcosa di programmabile in maniera diretta
4. Combiniamo i pezzi per formare la soluzione al problema originale

Esempio

Esempio in scratch

Esercizio 2

Disegnare una griglia di Sudoku come la seguente (colori compresi):



Top-down VS Bottom-up

- **Top-down** 👉 parti dal **problema grande** e lo dividi in **sottoproblemi più piccoli**, fino ad arrivare a passi semplici da risolvere.
  (Esempio: voglio fare una torta → divido in: comprare ingredienti, preparare impasto, cuocere, servire).
- **Bottom-up** 👉 parti da **piccoli pezzi già risolti** e li combini per costruire via via soluzioni più complesse.
  (Esempio: so già fare la crema e la base → li metto insieme per fare la torta).

👉 In breve:

- **Top-down = dal generale al particolare**
- **Bottom-up = dal particolare al generale**

TOP-DOWN E BOTTOM-UP SONO COMPLEMENTARI.

Se usi solo Top-Down

- ✗ Parti dal problema grande e lo spezzi in sottoproblemi → ottimo per la chiarezza.
- ✗ Ma... rischi di progettare cose che poi sono difficili da implementare se non pensi a come riutilizzare pezzi già pronti.

Se usi solo Bottom-Up

- ✗ Parti da piccoli pezzi (funzioni, moduli, librerie) → ottimo per riuso e praticità.
- ✗ Ma... rischi di costruire tante parti senza una visione globale del sistema.

TOP-DOWN E BOTTOM-UP SONO COMPLEMENTARI.

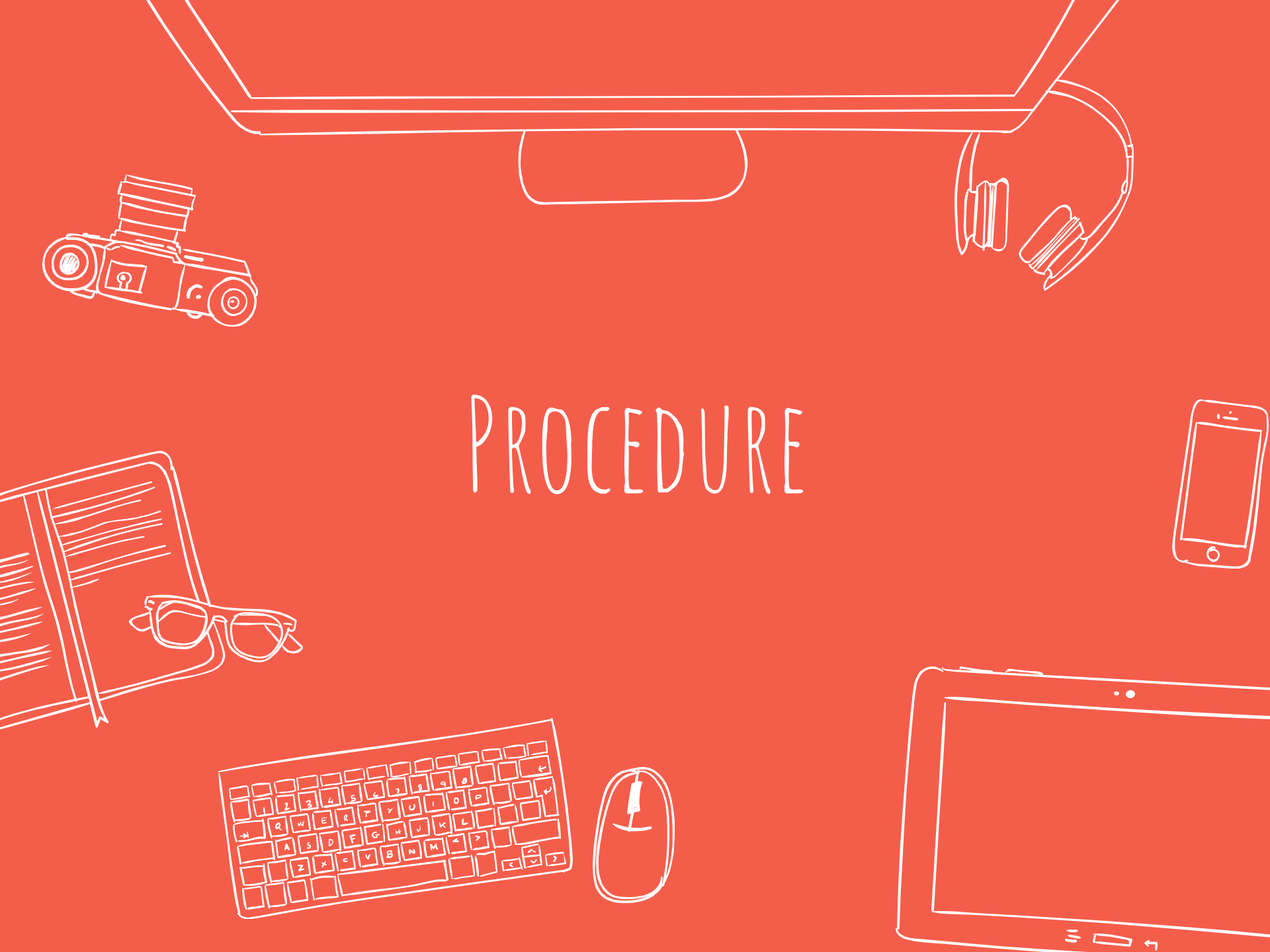
Perché si usano **in momenti diversi**:

- **Top-down** serve per capire *cosa* bisogna fare.
- **Bottom-up** serve per costruire *come* farlo, usando parti già pronte.

In pratica:

- Progetti un'app **top-down** (disegno architettura, schermate, flusso).
- La realizzi **bottom-up** (riciclo funzioni per il login, database già fatto, moduli esistenti).

PROCEDURE



Problemi

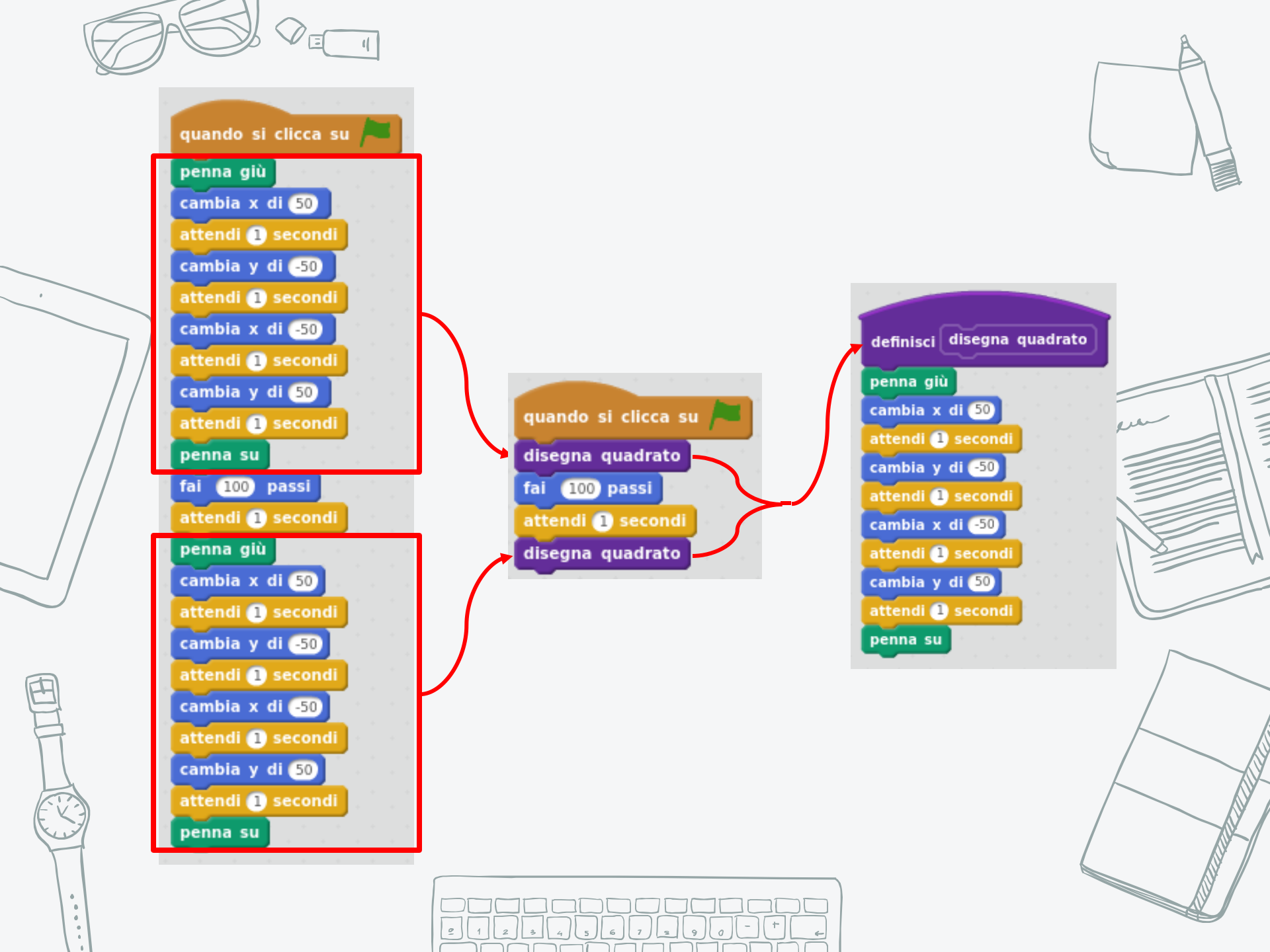
- ✕ Riscrivere la stessa sequenza di blocchi più volte
- ✕ Programmi con tanti (troppi) blocchi
- ✕ Creare programmi complessi è di per sé complesso, soprattutto se collaborare è difficile

Cos'è una procedura?

Una procedura è un blocco composto da altri blocchi e definito da noi.

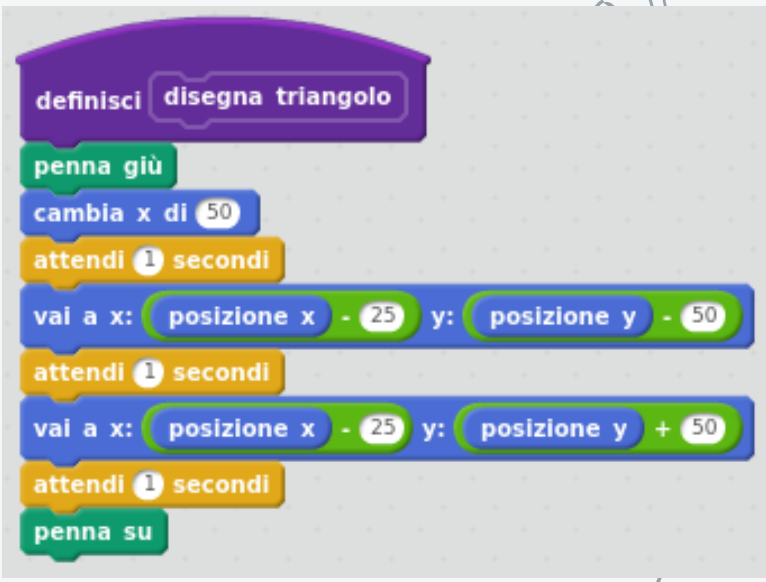
Perché è utile usare le procedura?

- Facilita la gestione della **complessità** nella composizione dei blocchi
- Permette il **riuso** dei blocchi
- Permette di creare un programma **collaborativamente**
- Riduce gli errori


quando si clicca su
penna giù
cambia x di 50
attendi 1 secondi
cambia y di -50
attendi 1 secondi
cambia x di -50
attendi 1 secondi
cambia y di 50
attendi 1 secondi
penna su
fai 100 passi
attendi 1 secondi
penna giù
cambia x di 50
attendi 1 secondi
cambia y di -50
attendi 1 secondi
cambia x di -50
attendi 1 secondi
cambia y di 50
attendi 1 secondi
penna su
quando si clicca su
disegna quadrato
fai 100 passi
attendi 1 secondi
disegna quadrato
definisci disegna quadrato
penna giù
cambia x di 50
attendi 1 secondi
cambia y di -50
attendi 1 secondi
cambia x di -50
attendi 1 secondi
cambia y di 50
attendi 1 secondi
penna su


quando si clicca su
disegna quadrato
fai 100 passi
attendi 1 secondi
disegna triangolo


definisci disegna quadrato
penna giù
cambia x di 50
attendi 1 secondi
cambia y di -50
attendi 1 secondi
cambia x di -50
attendi 1 secondi
cambia y di 50
attendi 1 secondi
penna su


definisci disegna triangolo
penna giù
cambia x di 50
attendi 1 secondi
vai a x: posizione x - 25 y: posizione y - 50
attendi 1 secondi
vai a x: posizione x - 25 y: posizione y + 50
attendi 1 secondi
penna su