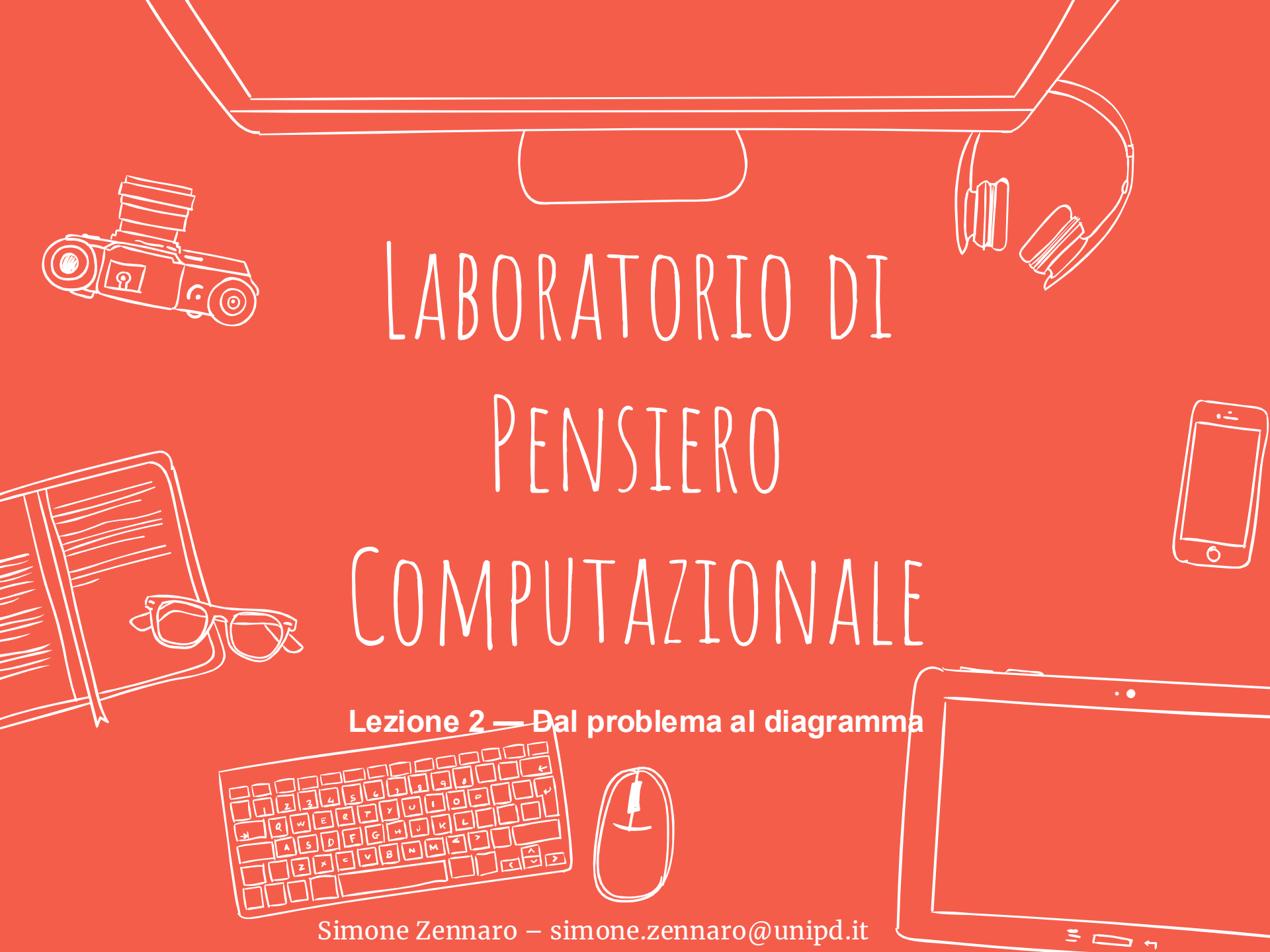
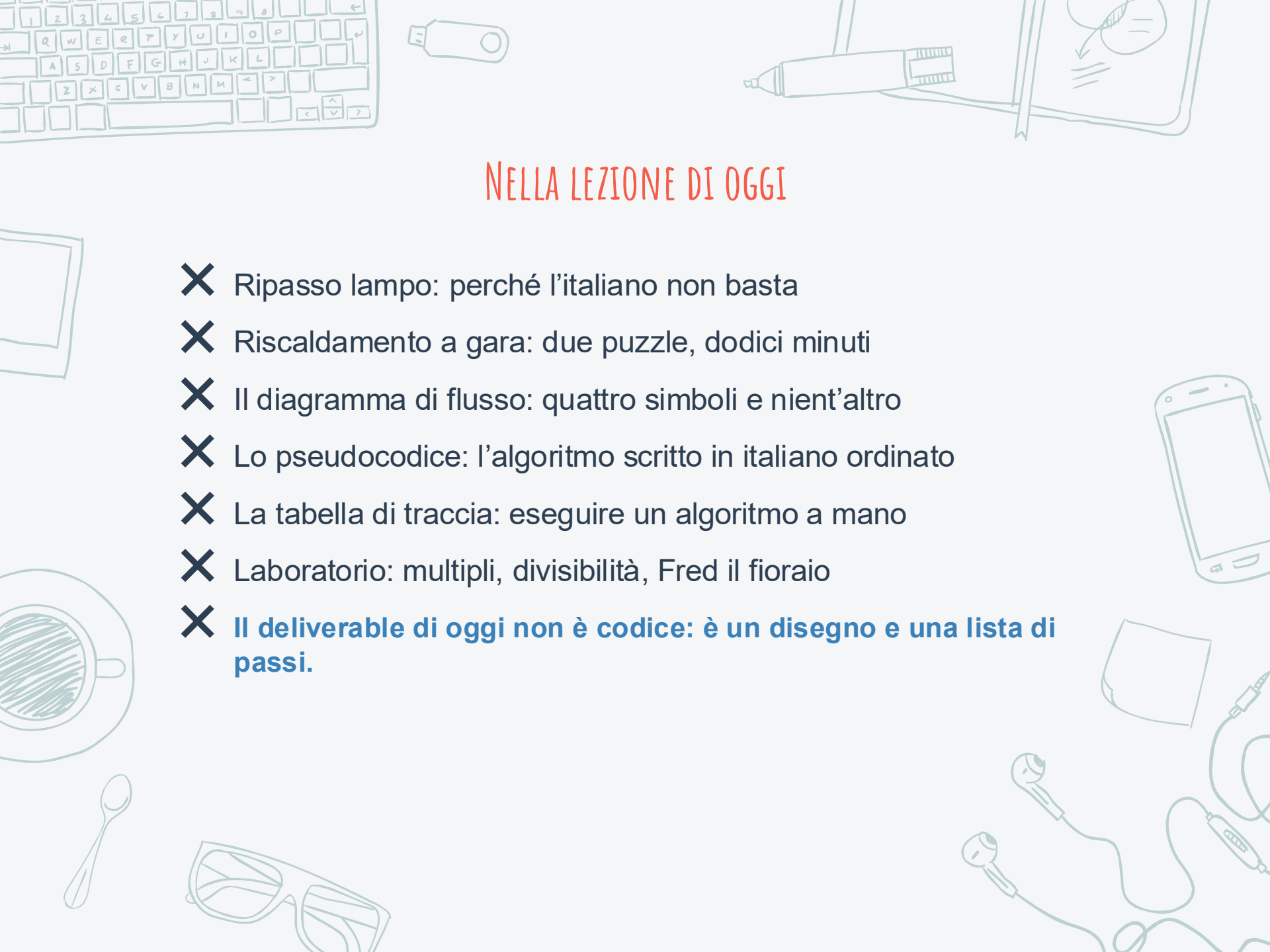




LABORATORIO DI PENSIERO COMPUTAZIONALE

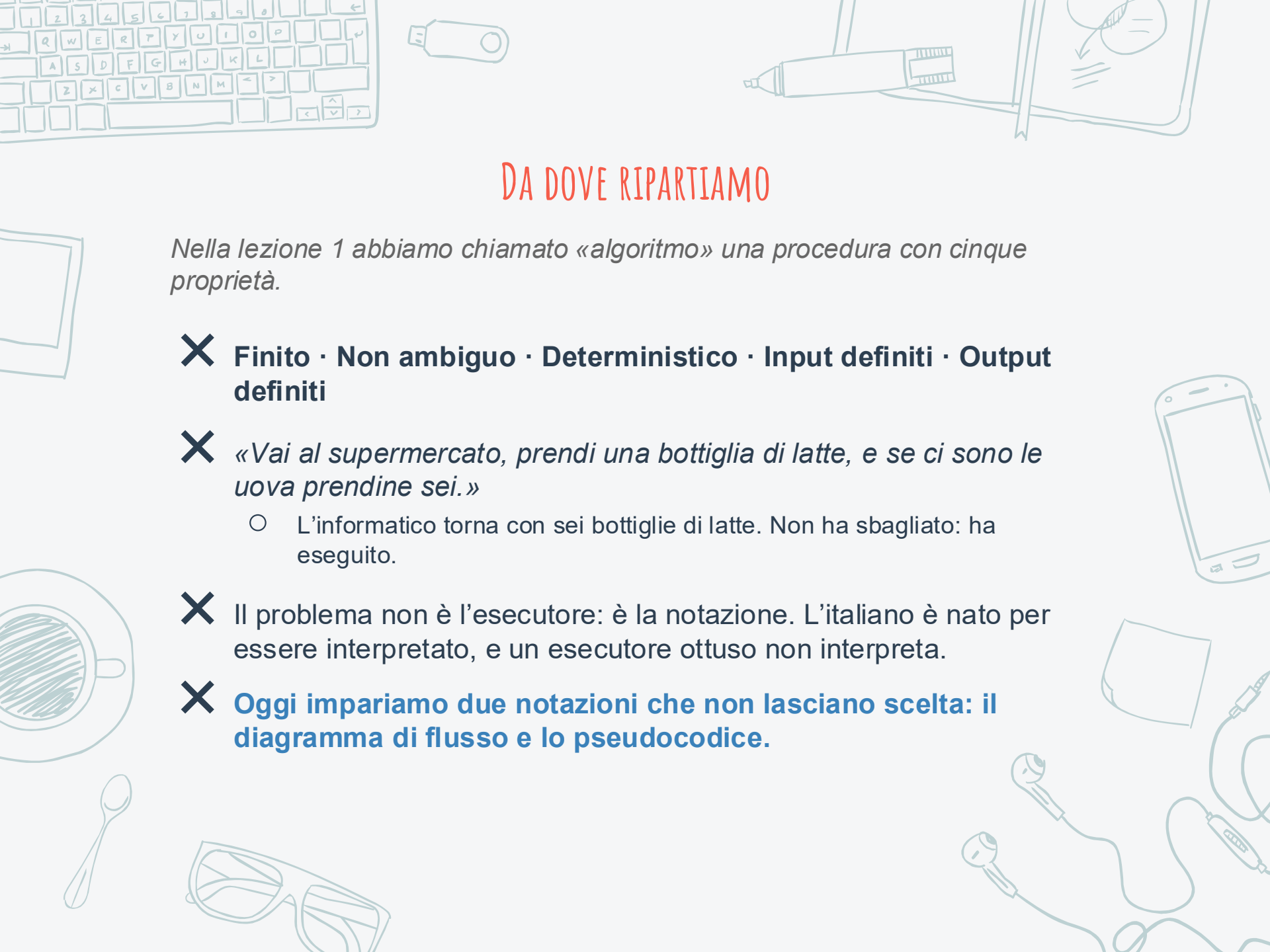
Lezione 2 — Dal problema al diagramma

Simone Zennaro – simone.zennaro@unipd.it



NELLA LEZIONE DI OGGI

- ✗ Ripasso lampo: perché l'italiano non basta
- ✗ Riscaldamento a gara: due puzzle, dodici minuti
- ✗ Il diagramma di flusso: quattro simboli e nient'altro
- ✗ Lo pseudocodice: l'algoritmo scritto in italiano ordinato
- ✗ La tabella di traccia: eseguire un algoritmo a mano
- ✗ Laboratorio: multipli, divisibilità, Fred il fioraio
- ✗ **Il deliverable di oggi non è codice: è un disegno e una lista di passi.**



DA DOVE RIPARTIAMO

Nella lezione 1 abbiamo chiamato «algoritmo» una procedura con cinque proprietà.

- ✗ **Finito · Non ambiguo · Deterministico · Input definiti · Output definiti**
- ✗ *«Vai al supermercato, prendi una bottiglia di latte, e se ci sono le uova prendine sei.»*
 - L'informatico torna con sei bottiglie di latte. Non ha sbagliato: ha eseguito.
- ✗ Il problema non è l'esecutore: è la notazione. L'italiano è nato per essere interpretato, e un esecutore ottuso non interpreta.
- ✗ **Oggi impariamo due notazioni che non lasciano scelta: il diagramma di flusso e lo pseudocodice.**

RISCALDAMENTO





CODE.ORG

✗ La difficoltà non è capire il problema, è scrivere la sequenza esatta.

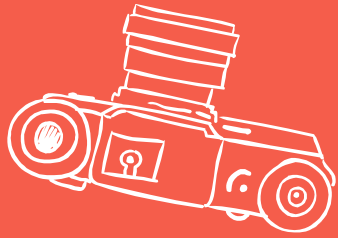
✗ Round 1 — [Angry Birds: programmazione a sequenze](#)

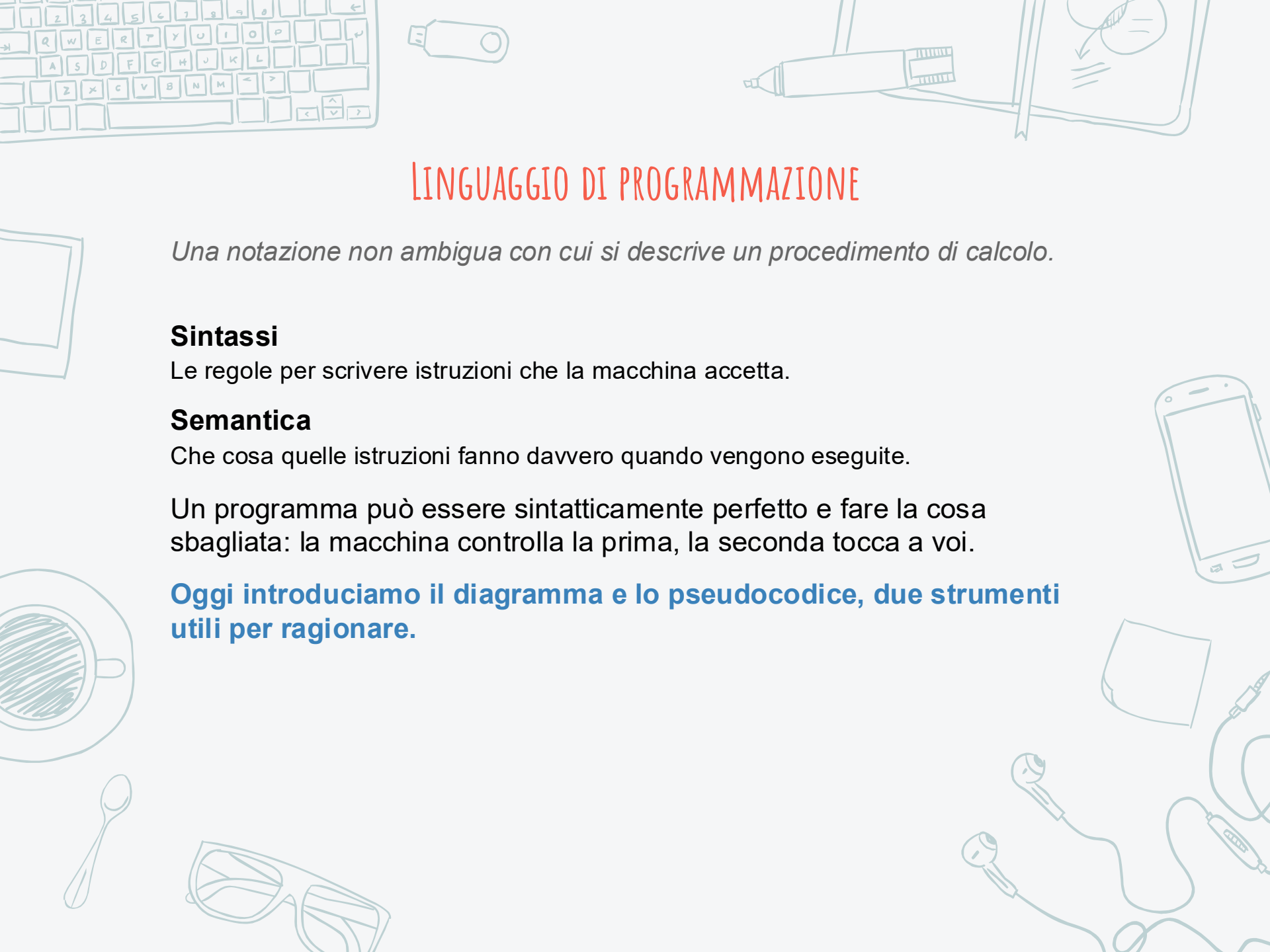
- [studio.code.org](#) → Express Course → «Programming with Angry Birds»
- 10 minuti

✗ Round 2 — [If/Else with Bee](#)

- [studio.code.org](#) → Express Course → «If/Else with Bee»
- 10 minuti

LA LINGUA DEI DIAGRAMMI





LINGUAGGIO DI PROGRAMMAZIONE

Una notazione non ambigua con cui si descrive un procedimento di calcolo.

Sintassi

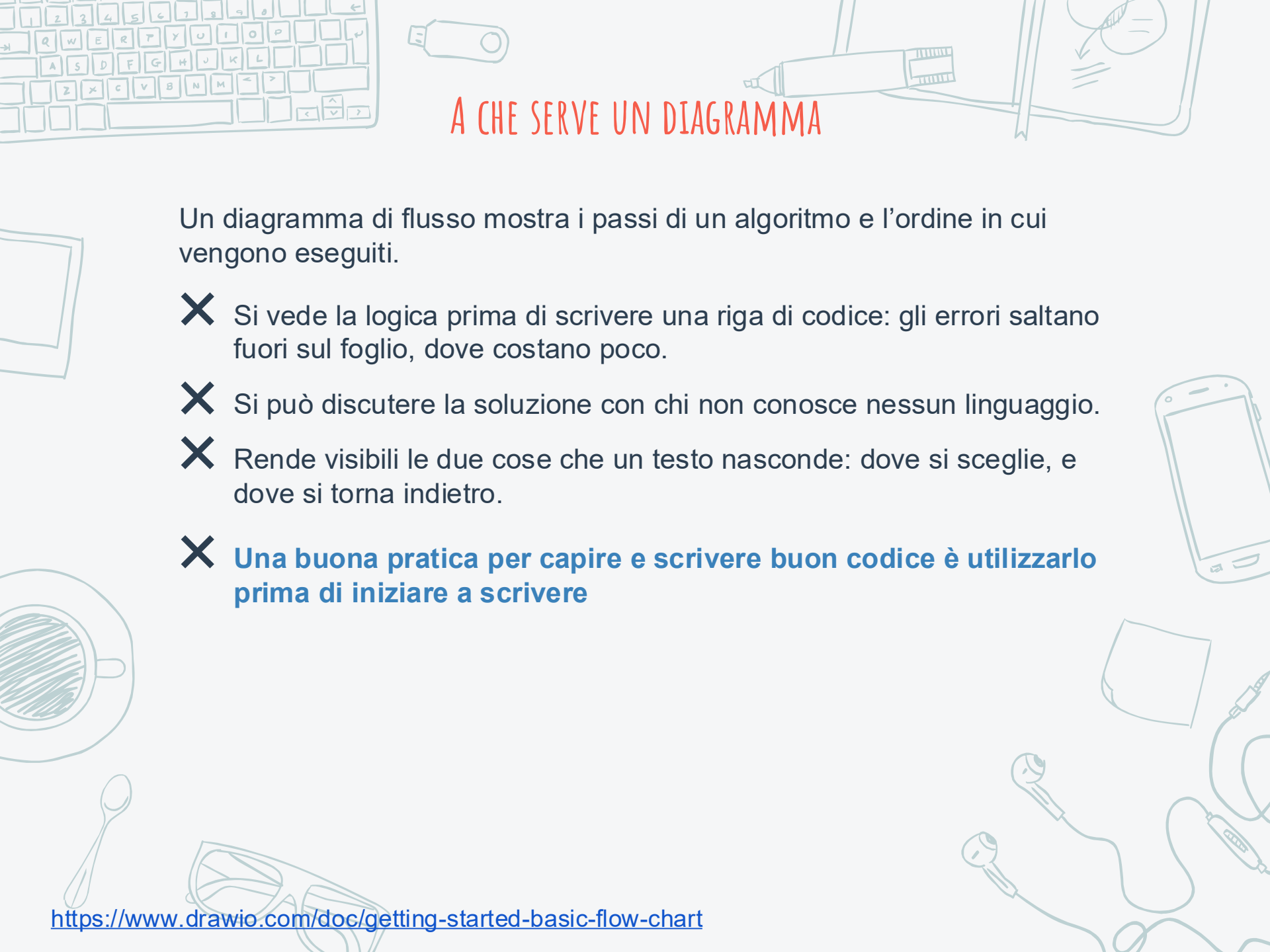
Le regole per scrivere istruzioni che la macchina accetta.

Semantica

Che cosa quelle istruzioni fanno davvero quando vengono eseguite.

Un programma può essere sintatticamente perfetto e fare la cosa sbagliata: la macchina controlla la prima, la seconda tocca a voi.

Oggi introduciamo il diagramma e lo pseudocodice, due strumenti utili per ragionare.



A CHE SERVE UN DIAGRAMMA

Un diagramma di flusso mostra i passi di un algoritmo e l'ordine in cui vengono eseguiti.

- ✗ Si vede la logica prima di scrivere una riga di codice: gli errori saltano fuori sul foglio, dove costano poco.
- ✗ Si può discutere la soluzione con chi non conosce nessun linguaggio.
- ✗ Rende visibili le due cose che un testo nasconde: dove si sceglie, e dove si torna indietro.
- ✗ **Una buona pratica per capire e scrivere buon codice è utilizzarlo prima di iniziare a scrivere**

I SIMBOLI, TUTTI QUANTI

- ✗ Un diagramma di flusso ha quattro forme. Quattro, non di più.
- ✗ **Le frecce dicono l'ordine; il rombo è l'unico punto in cui il percorso si divide.**

INIZIO

Compito
semplice

Input/
Output

FINE

Falso

Condizione

Vero

CHE COSA DICE OGNI FORMA

INIZIO

FINE

Inizio e fine del processo

Compito
semplice

Un'operazione: un calcolo,
un'assegnazione

Falso

Vero

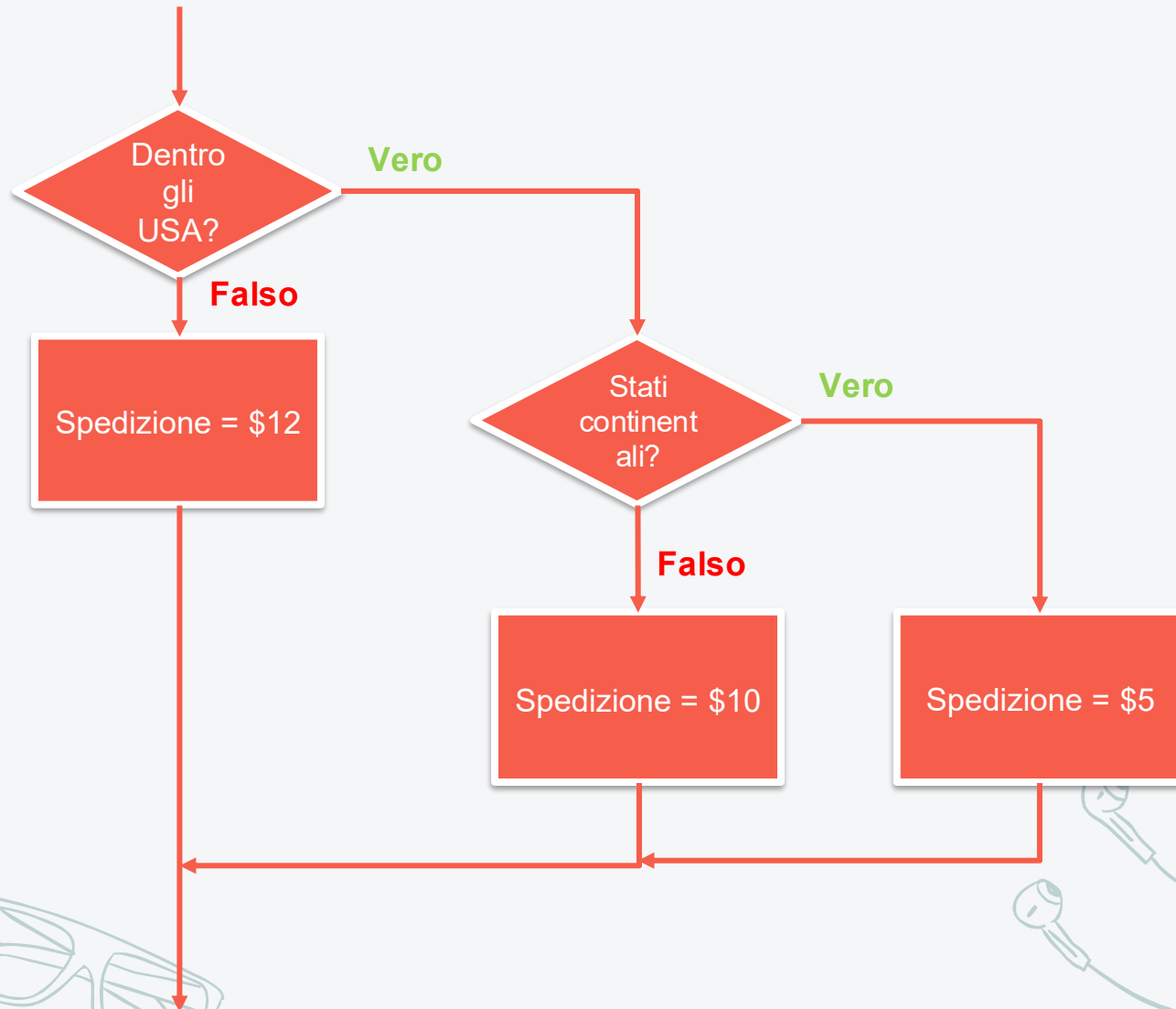
Condizione

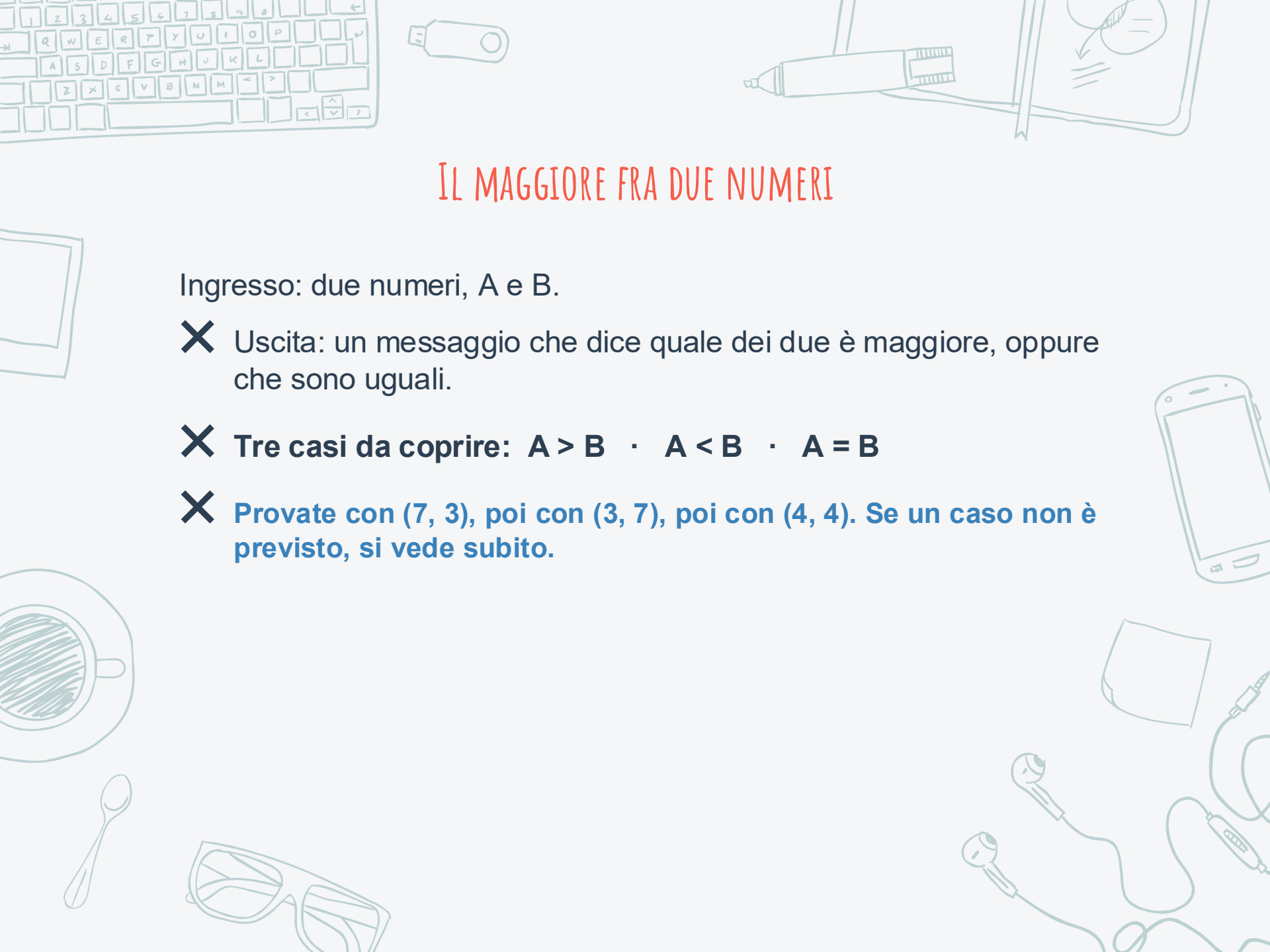
Una decisione: il percorso si
divide in Vero e Falso

Input/
Output

Ingresso o uscita di dati: leggi,
scrivi

Esempio guidato — Le spese di spedizione costano \$5 nei 48 stati continentali degli USA, \$10 verso Hawaii e Alaska, \$12 all'estero. Disegnare il diagramma che calcola il costo.





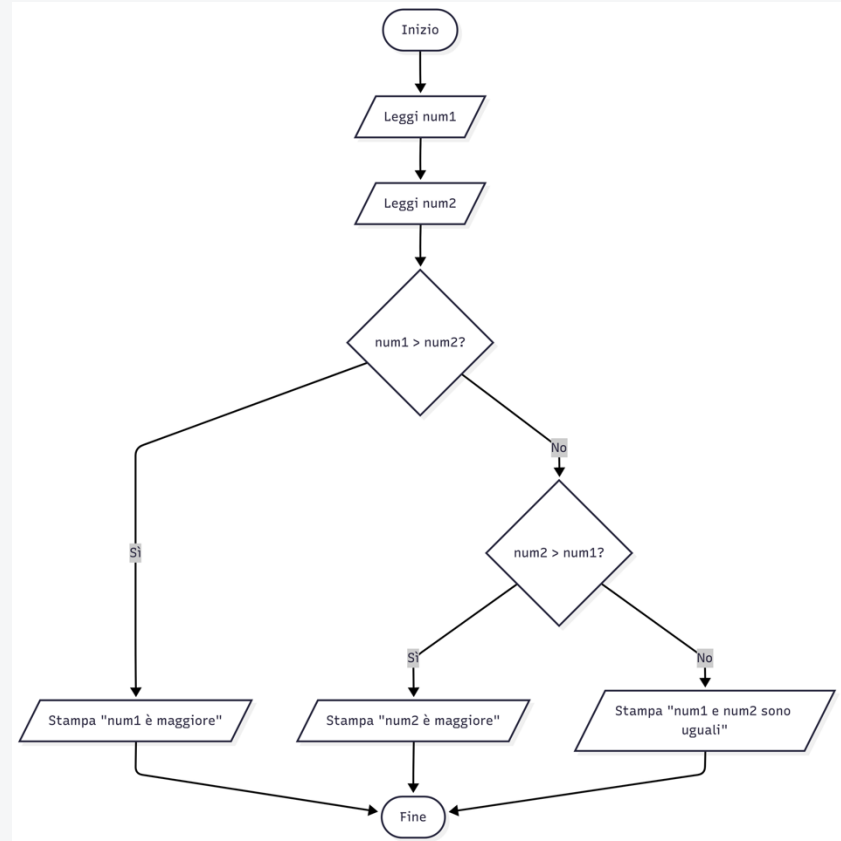
IL MAGGIORE FRA DUE NUMERI

Ingresso: due numeri, A e B.

- ✗ Uscita: un messaggio che dice quale dei due è maggiore, oppure che sono uguali.
- ✗ Tre casi da coprire: $A > B$ · $A < B$ · $A = B$
- ✗ Provate con (7, 3), poi con (3, 7), poi con (4, 4). Se un caso non è previsto, si vede subito.

IL MAGGIORE — SOLUZIONE

```
leggi num1, num2
se num1 > num2:
    scrivi "num1 è maggiore"
altrimenti se num2 > num1:
    scrivi "num2 è maggiore"
altrimenti:
    scrivi "sono uguali"
```



LO PSEUDOCODICE

Descrive un algoritmo in italiano, ma con una struttura fissa.

- ✗ Poche parole chiave, sempre le stesse: leggi, scrivi, se / altrimenti, fintanto che.
- ✗ L'indentazione dice quali istruzioni stanno dentro un blocco. È l'unica punteggiatura che conta.
- ✗ Nessuna macchina lo esegue, quindi nessuna macchina vi dirà che è sbagliato: il controllo tocca a voi.
- ✗ **Diagramma e pseudocodice dicono la stessa cosa: il primo si guarda, il secondo si traduce.**

Pseudocodice

```
1  Scrivi: "Inserisci la tua età: "  
2  Se l'età è uguale o maggiore di 18:  
3      Scrivi "Sei maggiorenne"  
4  Altrimenti:  
5      Scrivi "Non sei maggiorenne"
```

Python

```
1  age=input("Inserisci la tua età: ")  
2  if int(age)>=18:  
3      print("Sei maggiorenne")  
4  else:  
5      print("Non sei maggiorenne")
```

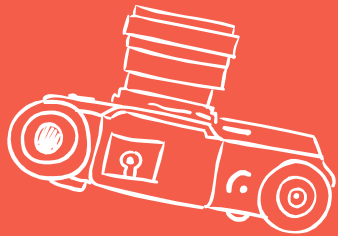
IL MASSIMO, IN PSEUDOCODICE

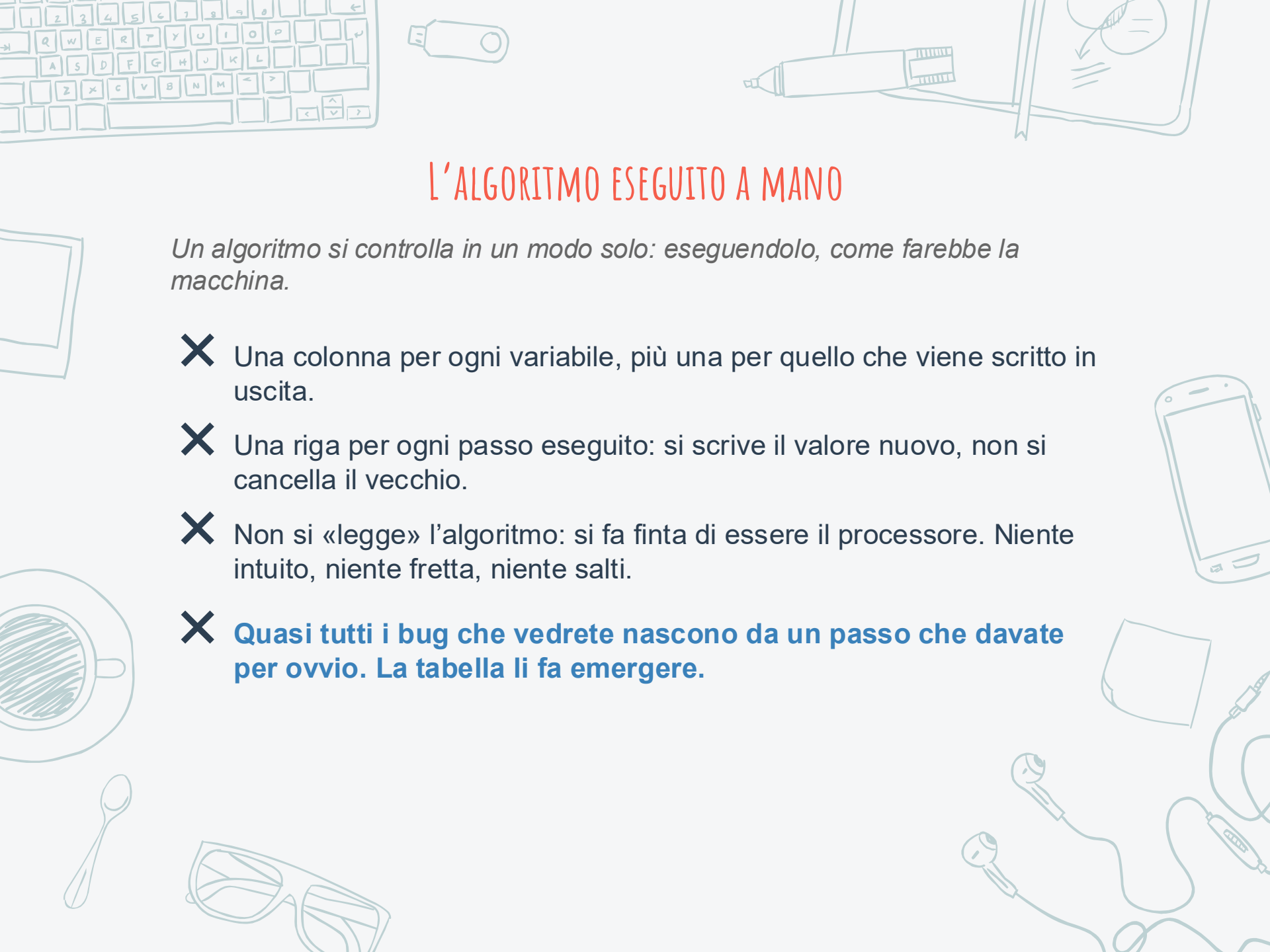
- ✗ Problema: dati due numeri A e B, trovare il maggiore.
- ✗ A parole: il massimo è il più grande dei due; se sono uguali, è quel valore.

```
se A > B:  
    max = A  
altrimenti:  
    max = B  
scrivi max
```

La trappola: perché non si comincia con «max = 0»?
Provate con A = -5 e B = -3.

LA TABELLA DI TRACCIA





L'ALGORITMO ESEGUITO A MANO

Un algoritmo si controlla in un modo solo: eseguendolo, come farebbe la macchina.

- ✗ Una colonna per ogni variabile, più una per quello che viene scritto in uscita.
- ✗ Una riga per ogni passo eseguito: si scrive il valore nuovo, non si cancella il vecchio.
- ✗ Non si «legge» l'algoritmo: si fa finta di essere il processore. Niente intuito, niente fretta, niente salti.
- ✗ **Quasi tutti i bug che vedrete nascono da un passo che davate per ovvio. La tabella li fa emergere.**

TRACCIAMO INSIEME I MULTIPLI

```
leggi n, k  
i = 1  
fintanto che  $i \leq n$ :  
    scrivi  $i \times k$   
     $i = i + 1$ 
```

Con $n = 3$ e $k = 4$.

Ogni riga è un passaggio dal rombo: prima si controlla la condizione, poi si esegue il corpo del ciclo. Il valore nuovo si scrive accanto al vecchio, non al posto suo.

Passo	i	$i \leq n$?	Scrive
inizio	1	—	
1	1	Vero	4
2	2	Vero	8
3	3	Vero	12
4	4	Falso	esce dal ciclo

PROVATE VOI: 13 DIVISO 4

leggi n , k
 $m = n$
fintanto che $m \geq k$:
 $m = m - k$
se $m = 0$:
 scrivi "Sì"
altrimenti:
 scrivi "No"

$n = 13, k = 4$.

Compilate la tabella fino alla fine, poi dite che cosa viene scritto in uscita. Aggiungete righe se ne servono: quante siano lo decide l'algoritmo, non il foglio.

Passo	m	$m \geq k$?
inizio	13	—
1		
2		
3		
4		

Poi rifatela con $n = 12$ e $k = 4$: cambia una riga sola, e cambia la risposta. È lì che sta tutta la divisibilità.