



LABORATORIO DI PENSIERO COMPUTAZIONALE

Lezione 5 — Le funzioni

Simone Zennaro – simone.zennaro@unipd.it

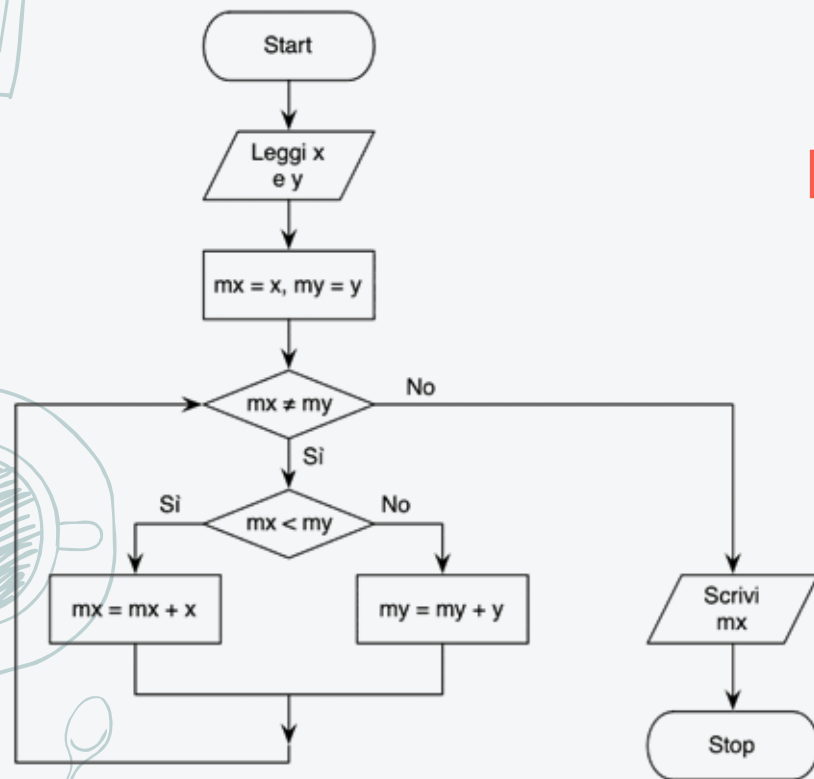


NELLA LEZIONE DI OGGI

- ✗ Ripasso dell'esercizio sul mcm
- ✗ Riscaldamento con code.org
- ✗ Che cos'è una funzione e come si scrive
- ✗ Parametri e contratto
- ✗ Tabella di traccia con una chiamata a funzione
- ✗ Esercizi durante la lezione: mcm, numero primo, conto corrente, contare i primi

IL MCM

Esercizio 5 della lezione scorsa: calcolare il minimo comune multiplo usando solo la somma. Si fa avanzare ogni volta il multiplo più piccolo, finché i due multipli non coincidono.



```
1: leggi x e y
2:  $m_x = x$ 
3:  $m_y = y$ 
4: fintanto che  $m_x \neq m_y$ 
5:   se  $m_x < m_y$  allora
6:      $m_x = m_x + x$ 
7:   altrimenti
8:      $m_y = m_y + y$ 
9: scrivi "il mcm tra x e y è  $m_x$ "
10: stop
```

UNO SCHEMA CHE SI RIPETE

Negli esercizi delle ultime due lezioni il ciclo centrale è quasi sempre lo stesso.

Esercizio	Ciclo centrale
Divisibilità	fintanto che $m \geq k$: $m = m - k$
Divisione intera	fintanto che $r \geq y$: $r = r - y$, $d = d + 1$
MCD	fintanto che $x \neq y$: sottrai il minore dal maggiore
mcm	fintanto che $mx \neq my$: aumenta il minore

In tutti e quattro si sottrae o si somma finché non si verifica una condizione. Quando uno stesso procedimento serve più volte, conviene scriverlo una volta sola e dargli un nome.

RISCALDAMENTO





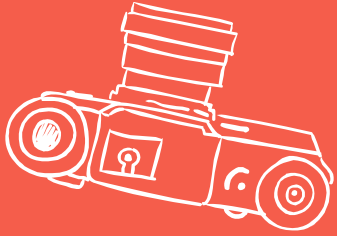
CODE.ORG: LE FUNZIONI

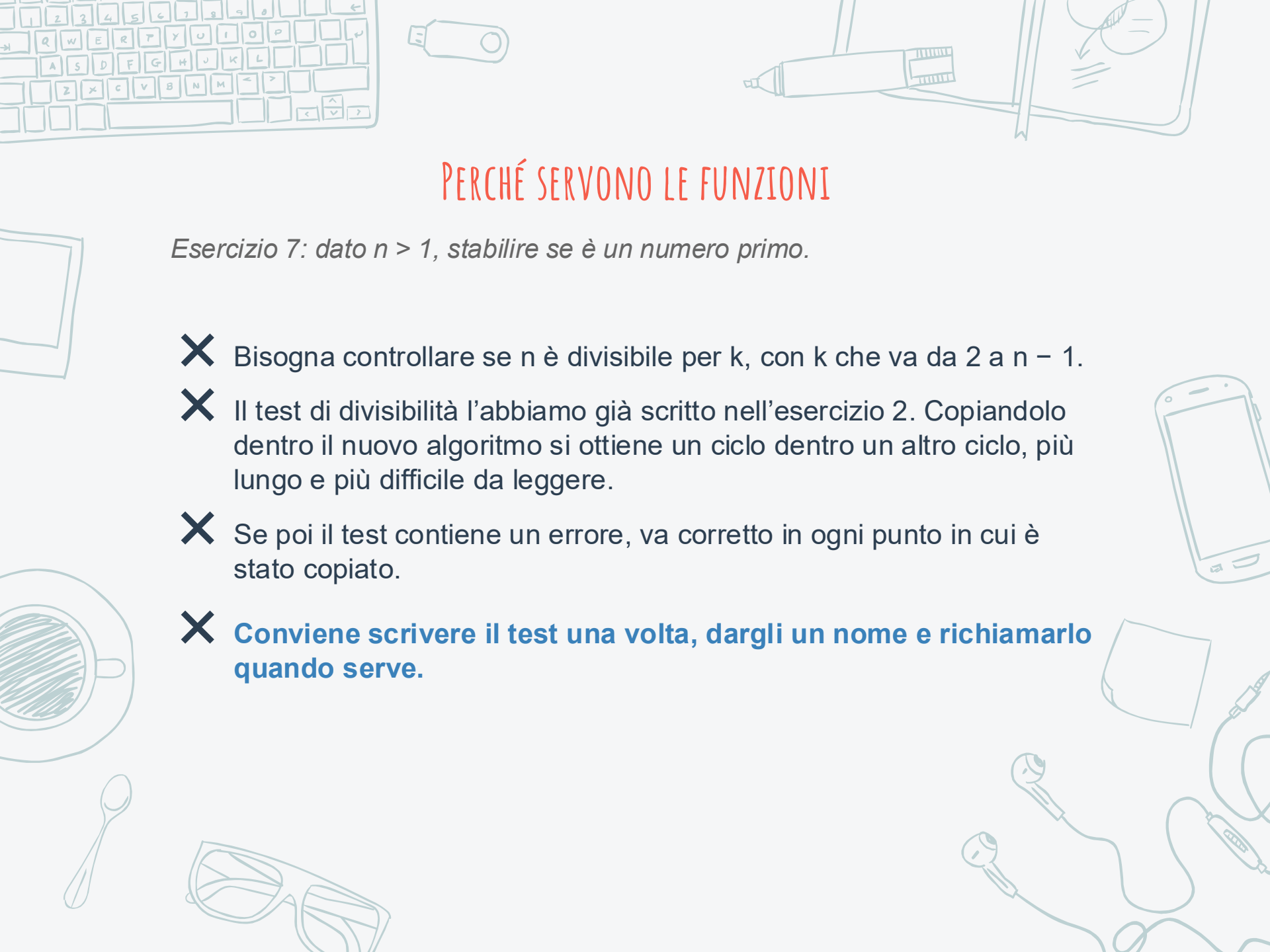
Otto minuti di gioco, poi tre minuti per confrontare i risultati in chat.

✗ Percorso sulle funzioni

- studio.code.org → Express Course → lezione 22
- ✗ Mentre giocate, contate quanti blocchi contiene la vostra funzione e quante volte la chiamate.
- ✗ Poi scrivete in chat quanti blocchi avreste usato senza la funzione.

LE FUNZIONI

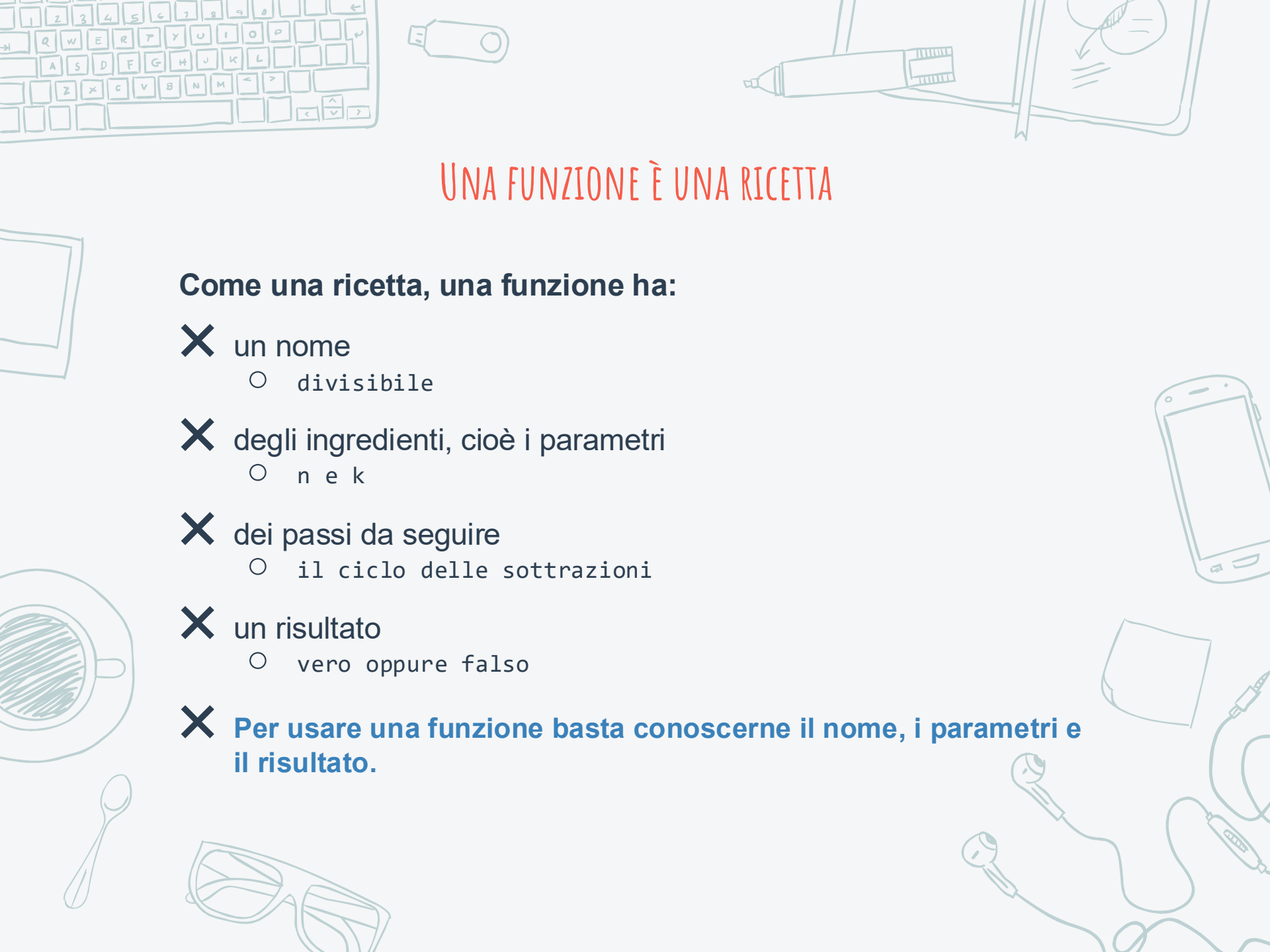




PERCHÉ SERVONO LE FUNZIONI

Esercizio 7: dato $n > 1$, stabilire se è un numero primo.

- ✗ Bisogna controllare se n è divisibile per k , con k che va da 2 a $n - 1$.
- ✗ Il test di divisibilità l'abbiamo già scritto nell'esercizio 2. Copiandolo dentro il nuovo algoritmo si ottiene un ciclo dentro un altro ciclo, più lungo e più difficile da leggere.
- ✗ Se poi il test contiene un errore, va corretto in ogni punto in cui è stato copiato.
- ✗ **Conviene scrivere il test una volta, dargli un nome e richiamarlo quando serve.**



UNA FUNZIONE È UNA RICETTA

Come una ricetta, una funzione ha:

- ✗ un nome
 - divisibile
- ✗ degli ingredienti, cioè i parametri
 - n e k
- ✗ dei passi da seguire
 - il ciclo delle sottrazioni
- ✗ un risultato
 - vero oppure falso
- ✗ **Per usare una funzione basta conoscerne il nome, i parametri e il risultato.**

LA SCATOLA NERA

Chi usa una funzione non ha bisogno di sapere come è fatta dentro.



Chi la usa deve sapere

che cosa riceve
che cosa restituisce
in quali casi funziona

Chi la usa può ignorare

come viene fatto il calcolo
quante sottrazioni servono
come si chiamano le variabili interne

Questo è un esempio di astrazione: si separa quello che la funzione fa da come lo fa. Se riscrivessimo divisibile usando la divisione, chi la usa non dovrebbe cambiare nulla.

IL CONTRATTO

Il contratto dice che cosa riceve una funzione, che cosa restituisce e a quali condizioni.

Funzione	Riceve	Restituisce	Condizioni
divisibile(n, k)	due interi	vero / falso	$n \geq 0, k > 0$
mcd(x, y)	due interi	un intero	$x > 0, y > 0$
primo(n)	un intero	vero / falso	$n > 1$

Se le condizioni non sono rispettate, la funzione può dare un risultato sbagliato o non terminare. Provate a tracciare `divisibile(12, 0)`.

Il contratto si scrive prima dei passi della funzione.

DEFINIRE E CHIAMARE

Definizione

```
funzione divisibile(n, k):  
    m = n  
    fintanto che  $m \geq k$ :  
        m = m - k  
    se m = 0 allora  
        restituisci vero  
    altrimenti  
        restituisci falso
```

Chiamata

```
leggi a, b  
se divisibile(a, b) allora  
    scrivi "sì"  
altrimenti  
    scrivi "no"
```

n e k sono i parametri. a e b sono gli argomenti, cioè i valori passati nella chiamata: n prende il valore di a e k quello di b.

«restituisci» termina la funzione e passa il risultato a chi l'ha chiamata. Nel diagramma di flusso ogni funzione ha un diagramma separato, con il suo INIZIO e la sua FINE.

ESERCIZIO — MCM IN UNA RIGA

Nell'esercizio 4 avete scritto $\text{mcd}(x, y)$. Usatela senza riscriverla.

- ✗ Vale la relazione $\text{mcm}(x, y) \times \text{mcd}(x, y) = x \times y$
 - $x = 4, y = 6$: $\text{mcm} = 12, \text{mcd} = 2, 12 \times 2 = 24 = 4 \times 6$
- ✗ Scrivete il contratto e lo pseudocodice di $\text{mcm}(x, y)$, chiamando mcd .
- ✗ Controllo: $\text{mcm}(4, 6) = 12$ e $\text{mcm}(12, 18) = 36$
- ✗ **Confrontatela con la versione a somme vista a inizio lezione: quale è più facile da controllare?**

SOLUZIONE – MCM

Contratto: $mcm(x, y)$ riceve due interi maggiori di 0 e restituisce il loro minimo comune multiplo.

```
funzione mcm(x, y):  
    restituisci  $x \times y / mcd(x, y)$ 
```

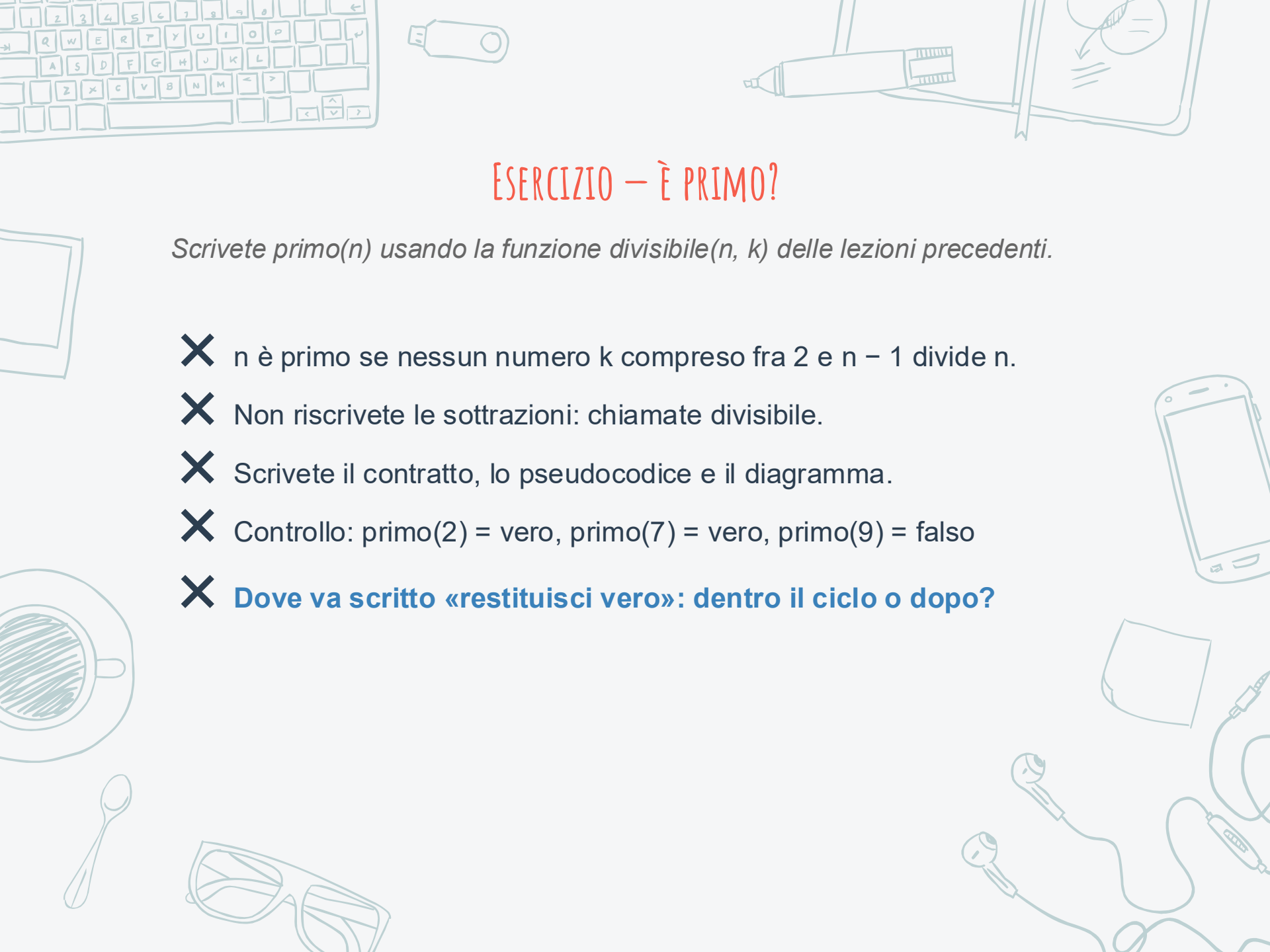
x	y	mcd(x, y)	$x \times y$	mcm(x, y)
4	6	2	24	12
12	18	6	216	36

La versione a somme dell'esercizio 5 usa un ciclo e quattro variabili. Questa è una riga sola e si appoggia a mcd, che è già stata controllata. Se mcd avesse un errore, lo si correggerebbe lì e mcm tornerebbe giusta senza modifiche.



SCRIVERE O RESTITUIRE

- ✗ Una funzione che scrive il risultato lo mostra sullo schermo.
- ✗ Una funzione che restituisce il risultato lo passa a chi l'ha chiamata, che può usarlo nei calcoli successivi.
- ✗ `primo(n)` può usare `divisibile` solo se `divisibile` restituisce il risultato.
- ✗ *Una funzione che non restituisce nulla si chiama anche procedura, come i blocchi del percorso di code.org.*
- ✗ **In questo corso le funzioni restituiscono il risultato; a scriverlo ci pensa chi le chiama.**



ESERCIZIO — È PRIMO?

Scrivete $\text{primo}(n)$ usando la funzione $\text{divisibile}(n, k)$ delle lezioni precedenti.

- ✗ n è primo se nessun numero k compreso fra 2 e $n - 1$ divide n .
- ✗ Non riscrivete le sottrazioni: chiamate divisibile .
- ✗ Scrivete il contratto, lo pseudocodice e il diagramma.
- ✗ Controllo: $\text{primo}(2) = \text{vero}$, $\text{primo}(7) = \text{vero}$, $\text{primo}(9) = \text{falso}$
- ✗ **Dove va scritto «restituisce vero»: dentro il ciclo o dopo?**

SOLUZIONE — È PRIMO?

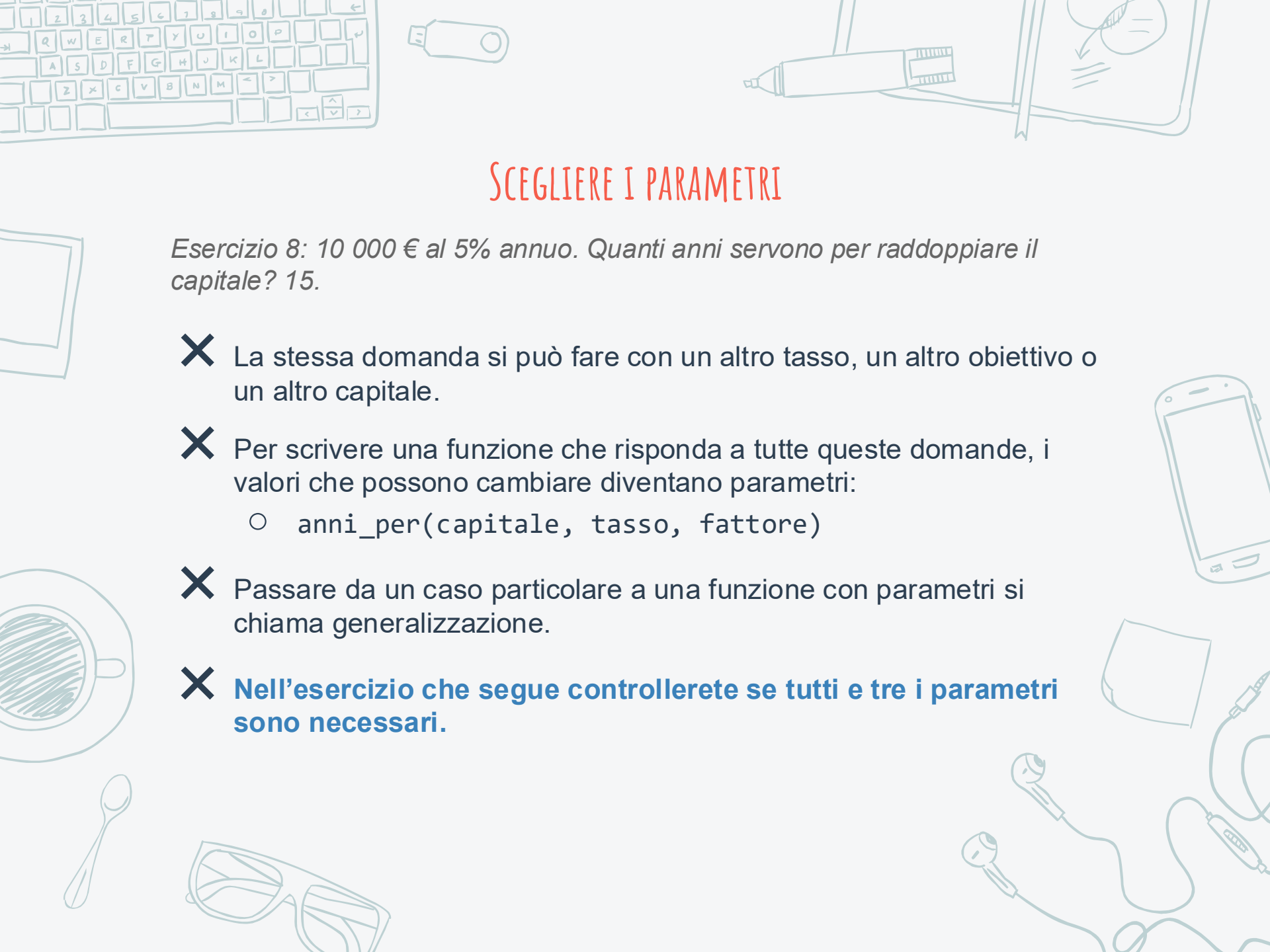
Contratto: $\text{primo}(n)$ riceve un intero $n > 1$ e restituisce vero se n è primo, falso altrimenti.

```
funzione primo(n):  
    k = 2  
    fintanto che k < n:  
        se divisibile(n, k) allora  
            restituisci falso  
        k = k + 1  
    restituisci vero
```

n	primo(n)	Perché
2	vero	il ciclo non viene eseguito
7	vero	nessun k da 2 a 6 divide 7
9	falso	divisibile(9, 3) è vero

«restituisci falso» sta dentro il ciclo, perché basta un divisore per rispondere.

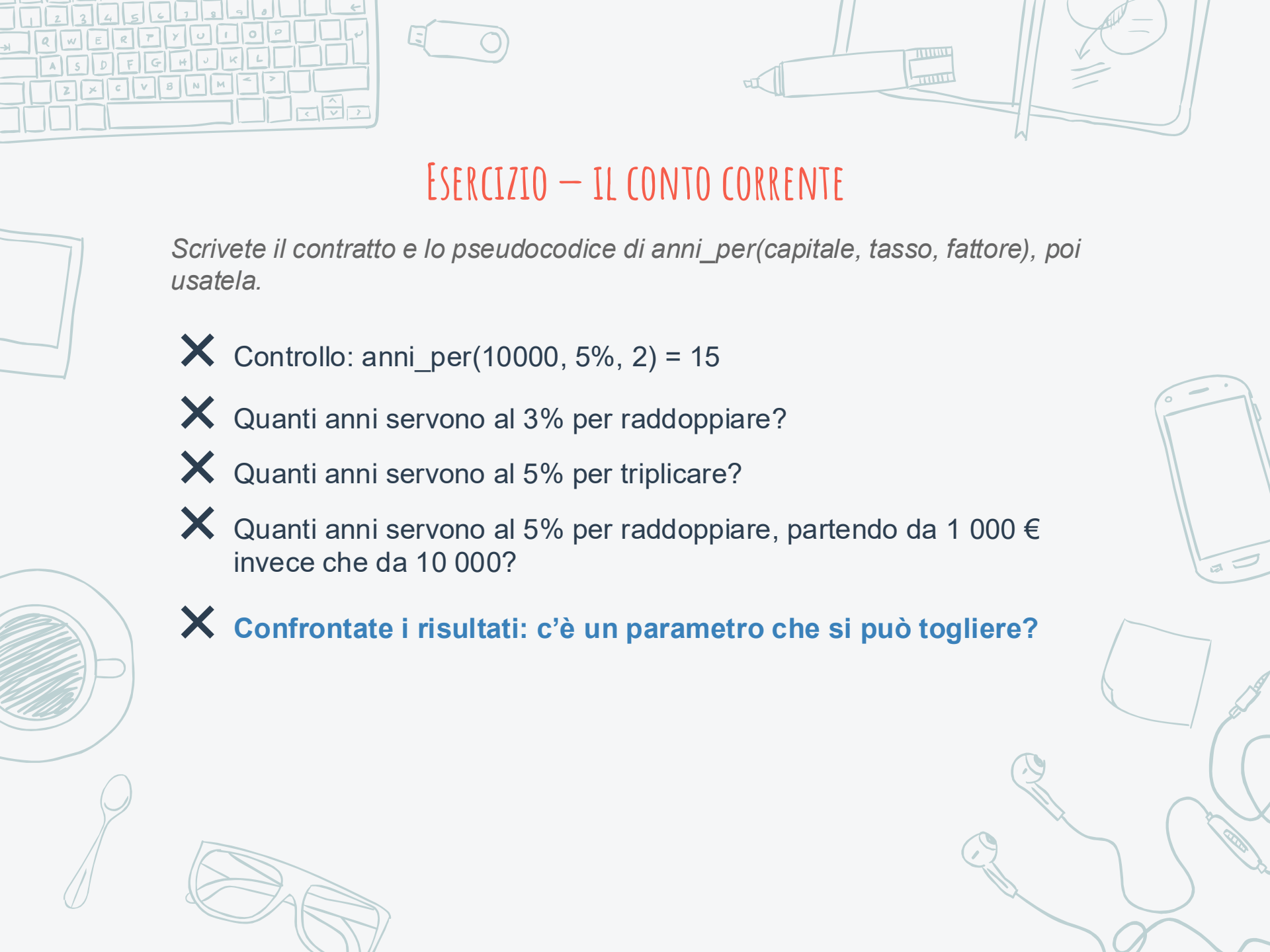
«restituisci vero» sta dopo il ciclo, perché si può dire solo dopo aver provato tutti i k.



SCEGLIERE I PARAMETRI

Esercizio 8: 10 000 € al 5% annuo. Quanti anni servono per raddoppiare il capitale? 15.

- ✗ La stessa domanda si può fare con un altro tasso, un altro obiettivo o un altro capitale.
- ✗ Per scrivere una funzione che risponda a tutte queste domande, i valori che possono cambiare diventano parametri:
 - `anni_per(capitale, tasso, fattore)`
- ✗ Passare da un caso particolare a una funzione con parametri si chiama generalizzazione.
- ✗ **Nell'esercizio che segue controllerete se tutti e tre i parametri sono necessari.**



ESERCIZIO — IL CONTO CORRENTE

Scrivete il contratto e lo pseudocodice di `anni_per(capitale, tasso, fattore)`, poi usatela.

- ✗ Controllo: `anni_per(10000, 5%, 2) = 15`
- ✗ Quanti anni servono al 3% per raddoppiare?
- ✗ Quanti anni servono al 5% per triplicare?
- ✗ Quanti anni servono al 5% per raddoppiare, partendo da 1 000 € invece che da 10 000?
- ✗ **Confrontate i risultati: c'è un parametro che si può togliere?**

SOLUZIONE – CONTO CORRENTE

Contratto: $\text{anni_per}(\text{capitale}, \text{tasso}, \text{fattore})$ riceve $\text{capitale} > 0$, $\text{tasso} > 0$ e $\text{fattore} > 1$, e restituisce gli anni necessari perché il saldo arrivi a $\text{capitale} \times \text{fattore}$.

```
funzione anni_per(capitale, tasso, fattore):  
    saldo = capitale  
    anni = 0  
    fintanto che saldo < capitale × fattore:  
        saldo = saldo × (1 + tasso)  
        anni = anni + 1  
    restituisci anni
```

Chiamata	Anni
$\text{anni_per}(10000, 5\%, 2)$	15
$\text{anni_per}(10000, 3\%, 2)$	24
$\text{anni_per}(10000, 5\%, 3)$	23
$\text{anni_per}(1000, 5\%, 2)$	15

Con 1 000 € e con 10 000 € il risultato è lo stesso: 15 anni.

IL CAPITALE NON SERVE

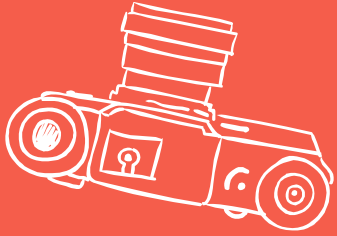
Le prime righe della tabella di traccia con due capitali diversi, al 5%.

Anni	Saldo con 1 000 €	Saldo con 10 000 €	Saldo / capitale
0	1 000,00	10 000,00	1,0000
1	1 050,00	10 500,00	1,0500
2	1 102,50	11 025,00	1,1025
3	1 157,63	11 576,25	1,1576

```
funzione anni_per(tasso, fattore):  
    valore = 1  
    anni = 0  
    fintanto che valore < fattore:  
        valore = valore × (1 + tasso)  
        anni = anni + 1  
    restituisci anni
```

Ogni anno il saldo viene moltiplicato per lo stesso numero, quindi il rapporto fra saldo e capitale non dipende dal capitale di partenza. Il parametro capitale si può togliere: basta partire da 1 e fermarsi quando si supera il fattore.

SEGUIRE UNA CHIAMATA



TRACCIAMO PRIMO(9)

```
funzione primo(n):  
  k = 2  
  fintanto che k < n:  
    se divisibile(n, k) allora  
      restituisci falso  
    k = k + 1  
  restituisci vero
```

Con $n = 9$.

È la funzione dell'esercizio precedente. Quando si arriva a $\text{divisibile}(9, k)$ non si traccia l'interno della funzione: si scrive direttamente il risultato.

k	$k < n$?	divisibile(9, k)	Che cosa succede
2	Vero	falso	$k = 3$
3	Vero	vero	restituisci falso

Bastano due righe. Con un programma più lungo si procede allo stesso modo, una funzione alla volta.

PROVATE VOI: PRIMO(7)

Stessa funzione, con $n = 7$. Completate la tabella, poi dite che cosa restituisce la funzione e quante volte ha chiamato divisibile.

k	$k < n$?	divisibile(7, k)	Che cosa succede

Se finite prima: quante chiamate a divisibile fa primo(97)? E primo(91)? Ne ripareremo nella lezione 6.

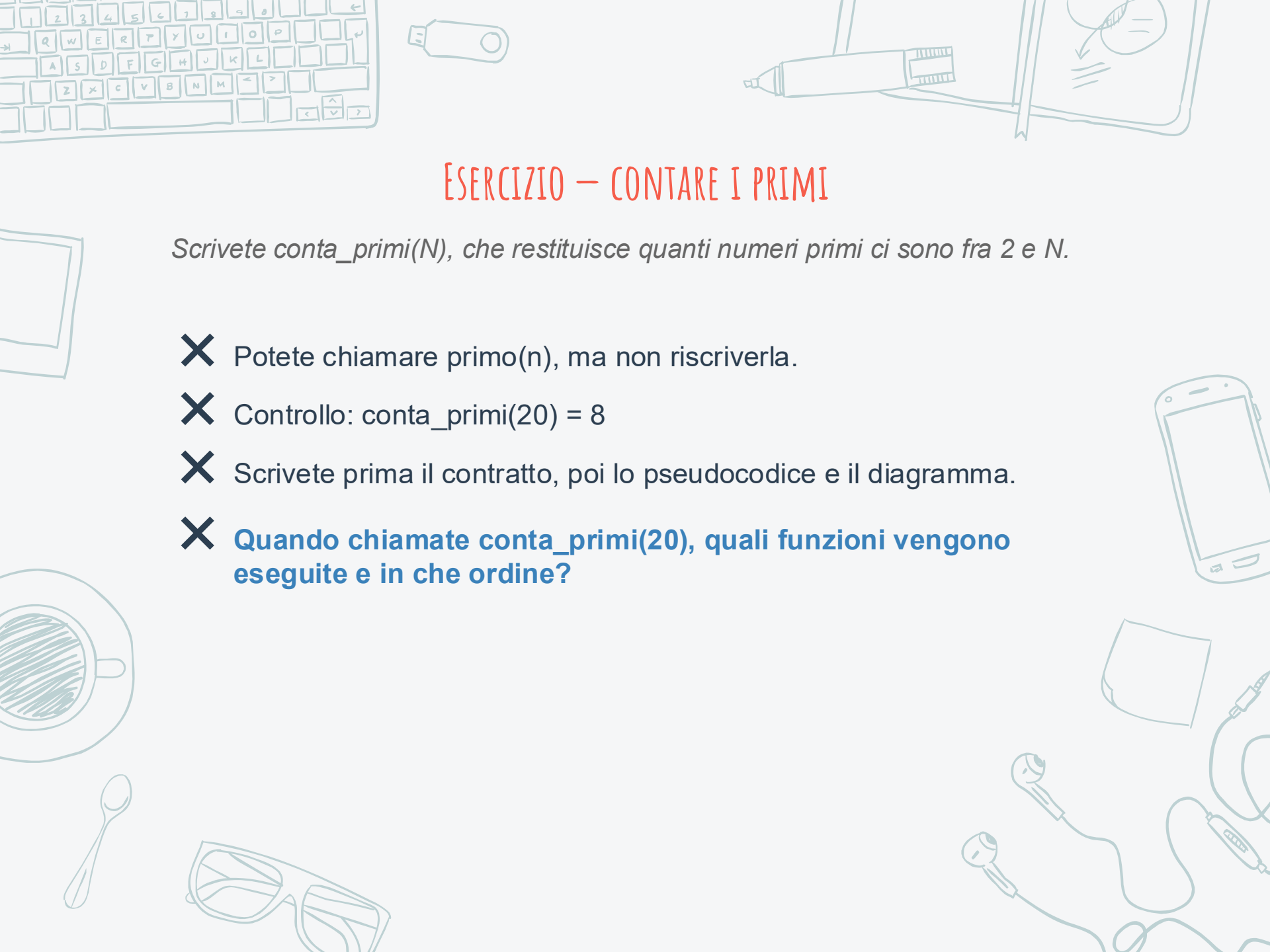
SOLUZIONE — PRIMO(7)

k	k < n ?	divisibile(7, k)	Che cosa succede
2	Vero	falso	k = 3
3	Vero	falso	k = 4
4	Vero	falso	k = 5
5	Vero	falso	k = 6
6	Vero	falso	k = 7
7	Falso	—	esce dal ciclo, restituisci vero

primo(7) restituisce vero dopo 5 chiamate a divisibile.

primo(97) fa 95 chiamate, perché 97 è primo e vanno provati tutti i k da 2 a 96.

primo(91) ne fa 6, perché $91 = 7 \times 13$ e si ferma a k = 7.



ESERCIZIO — CONTARE I PRIMI

Scrivete `conta_primi(N)`, che restituisce quanti numeri primi ci sono fra 2 e N .

- ✗ Potete chiamare `primo(n)`, ma non riscriverla.
- ✗ Controllo: `conta_primi(20) = 8`
- ✗ Scrivete prima il contratto, poi lo pseudocodice e il diagramma.
- ✗ **Quando chiamate `conta_primi(20)`, quali funzioni vengono eseguite e in che ordine?**

SOLUZIONE — CONTARE I PRIMI

Contratto: conta_primi(N) riceve un intero $N \geq 2$ e restituisce quanti numeri primi ci sono fra 2 e N.

```
funzione conta_primi(N):  
    c = 0  
    n = 2  
    fintanto che n ≤ N:  
        se primo(n) allora  
            c = c + 1  
        n = n + 1  
    restituisci c
```

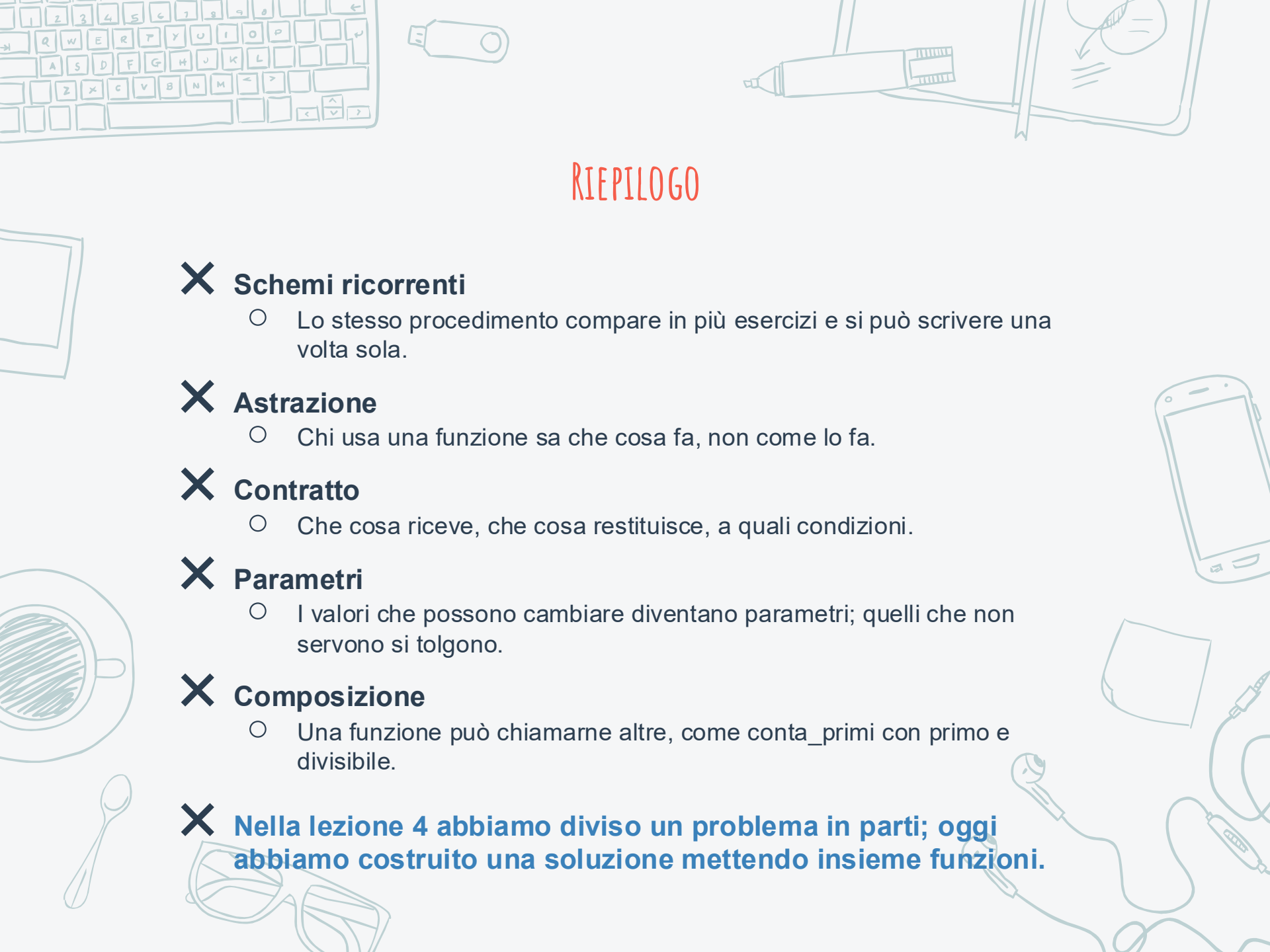
Con N = 20

primo(n) restituisce vero per
2, 3, 5, 7, 11, 13, 17, 19
quindi c = 8.

Chi viene eseguito

conta_primi chiama primo per ogni n;
primo chiama divisibile per ogni k.

È lo schema del contatore della lezione 4. L'unica differenza è che la condizione dentro il ciclo è una chiamata a funzione.



RIEPILOGO

✗ **Schemi ricorrenti**

- Lo stesso procedimento compare in più esercizi e si può scrivere una volta sola.

✗ **Astrazione**

- Chi usa una funzione sa che cosa fa, non come lo fa.

✗ **Contratto**

- Che cosa riceve, che cosa restituisce, a quali condizioni.

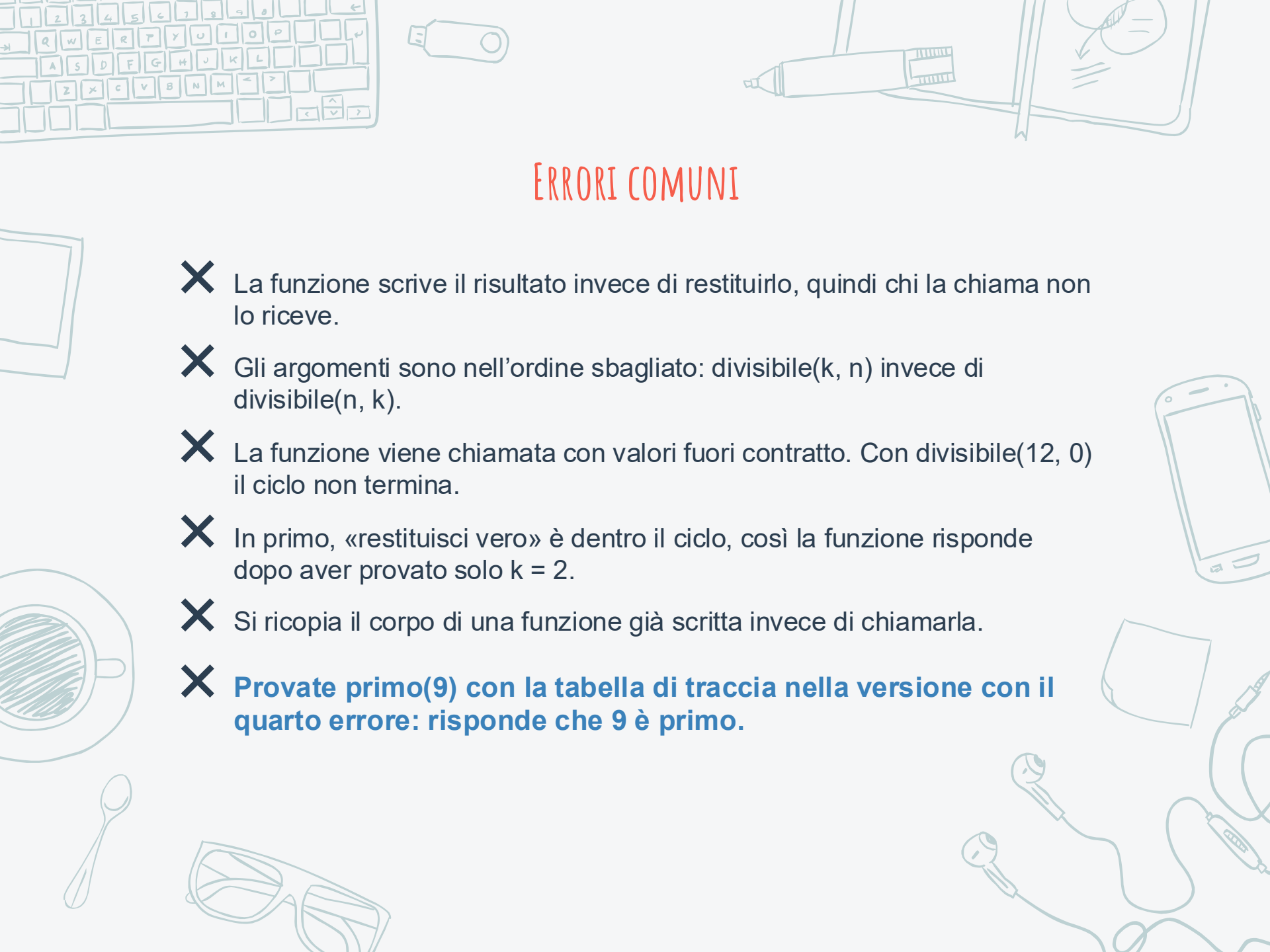
✗ **Parametri**

- I valori che possono cambiare diventano parametri; quelli che non servono si tolgono.

✗ **Composizione**

- Una funzione può chiamarne altre, come `conta_primi` con `primo` e `divisibile`.

✗ **Nella lezione 4 abbiamo diviso un problema in parti; oggi abbiamo costruito una soluzione mettendo insieme funzioni.**



ERRORI COMUNI

- ✗ La funzione scrive il risultato invece di restituirlo, quindi chi la chiama non lo riceve.
- ✗ Gli argomenti sono nell'ordine sbagliato: `divisibile(k, n)` invece di `divisibile(n, k)`.
- ✗ La funzione viene chiamata con valori fuori contratto. Con `divisibile(12, 0)` il ciclo non termina.
- ✗ In primo, «restituisce vero» è dentro il ciclo, così la funzione risponde dopo aver provato solo $k = 2$.
- ✗ Si ricopia il corpo di una funzione già scritta invece di chiamarla.
- ✗ **Provate `primo(9)` con la tabella di traccia nella versione con il quarto errore: risponde che 9 è primo.**

CINQUE MINUTI DI PYTHON

La funzione divisibile scritta in pseudocodice e in Python.

Pseudocodice

```
funzione divisibile(n, k):  
    m = n  
    fintanto che m ≥ k:  
        m = m - k  
    se m = 0 allora  
        restituisci vero  
    altrimenti  
        restituisci falso
```

Python

```
def divisibile(n, k):  
    m = n  
    while m >= k:  
        m = m - k  
    if m == 0:  
        return True  
    else:  
        return False
```

«funzione» diventa def e «restituisci» diventa return; il resto l'avete già visto. Anche le funzioni già pronte di Python, come len, max e sum, si usano conoscendone il contratto, senza sapere come sono scritte.

SFIDE FACOLTATIVE





SFIDA — LA CALCOLATRICE

Esercizio facoltativo in Python: partendo dal codice fornito, aggiungete nuove funzioni alla calcolatrice. Per ogni funzione scrivete prima il contratto.

CALCOLATRICE — IL CODICE

```
set scelta = 1
while scelta != 0
  print ( " --- Menu --- " )
  print ( " 1 - Somma " )
  print ( " 0 - Esci " )
  set scelta = int ( input ( " Scelta: " ) )
  if scelta == 1
    set a = int ( input ( " Inserisci a: " ) )
    set b = int ( input ( " Inserisci b: " ) )
    set c = somma ( a , b )
    print ( " Somma = " , c )
```

```
define somma
  parameters: a , b
  return a + b
```