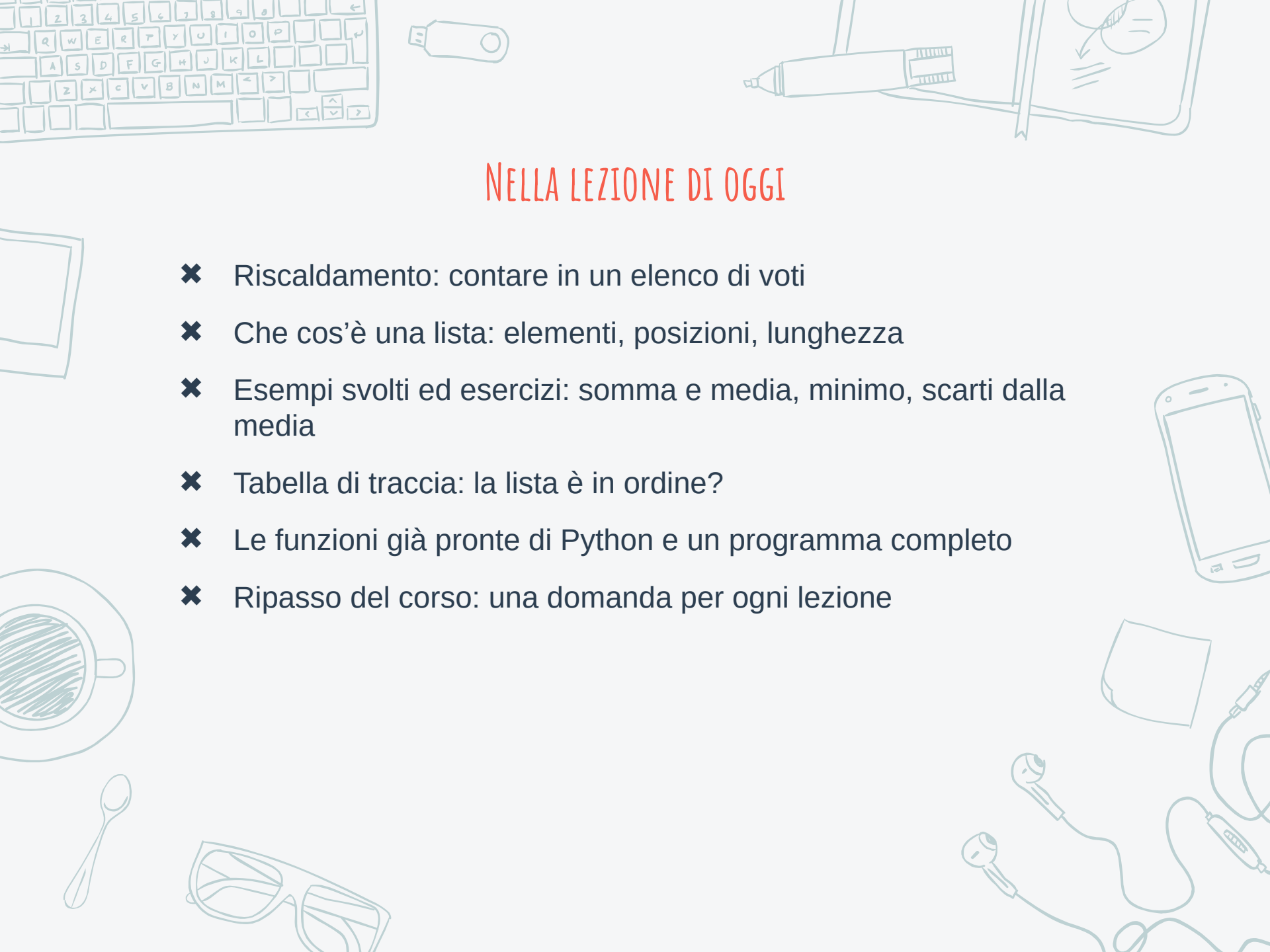




LABORATORIO DI PENSIERO COMPUTAZIONALE

Lezione 7 — Le liste

Simone Zennaro – simone.zennaro@unipd.it



NELLA LEZIONE DI OGGI

- ✗ Riscaldamento: contare in un elenco di voti
- ✗ Che cos'è una lista: elementi, posizioni, lunghezza
- ✗ Esempi svolti ed esercizi: somma e media, minimo, scarti dalla media
- ✗ Tabella di traccia: la lista è in ordine?
- ✗ Le funzioni già pronte di Python e un programma completo
- ✗ Ripasso del corso: una domanda per ogni lezione

MULTI VALORI, UN NOME

Per la media dei voti di un appello con 120 studenti servirebbero 120 variabili.

```
voto1 = 24  
voto2 = 18  
voto3 = 30  
...  
voto120 = 27
```

Con una variabile per voto il programma cambia ogni volta che cambia il numero di studenti.

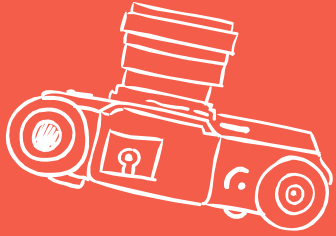
Una lista mette tutti i valori sotto un solo nome:

```
voti = [24, 18, 30, ..., 27]
```

Dato	Come lista
Voti di un appello	[24, 18, 30, 27]
Temperature di una settimana	[12.5, 14.0, 13.2, 9.8, 11.0, 12.1, 13.4]
Risposte a una domanda	["sì", "no", "sì", "sì"]

Nella lezione 1 la media dei voti partiva da «prendi l'elenco dei voti». L'elenco è una lista.

RISCALDAMENTO

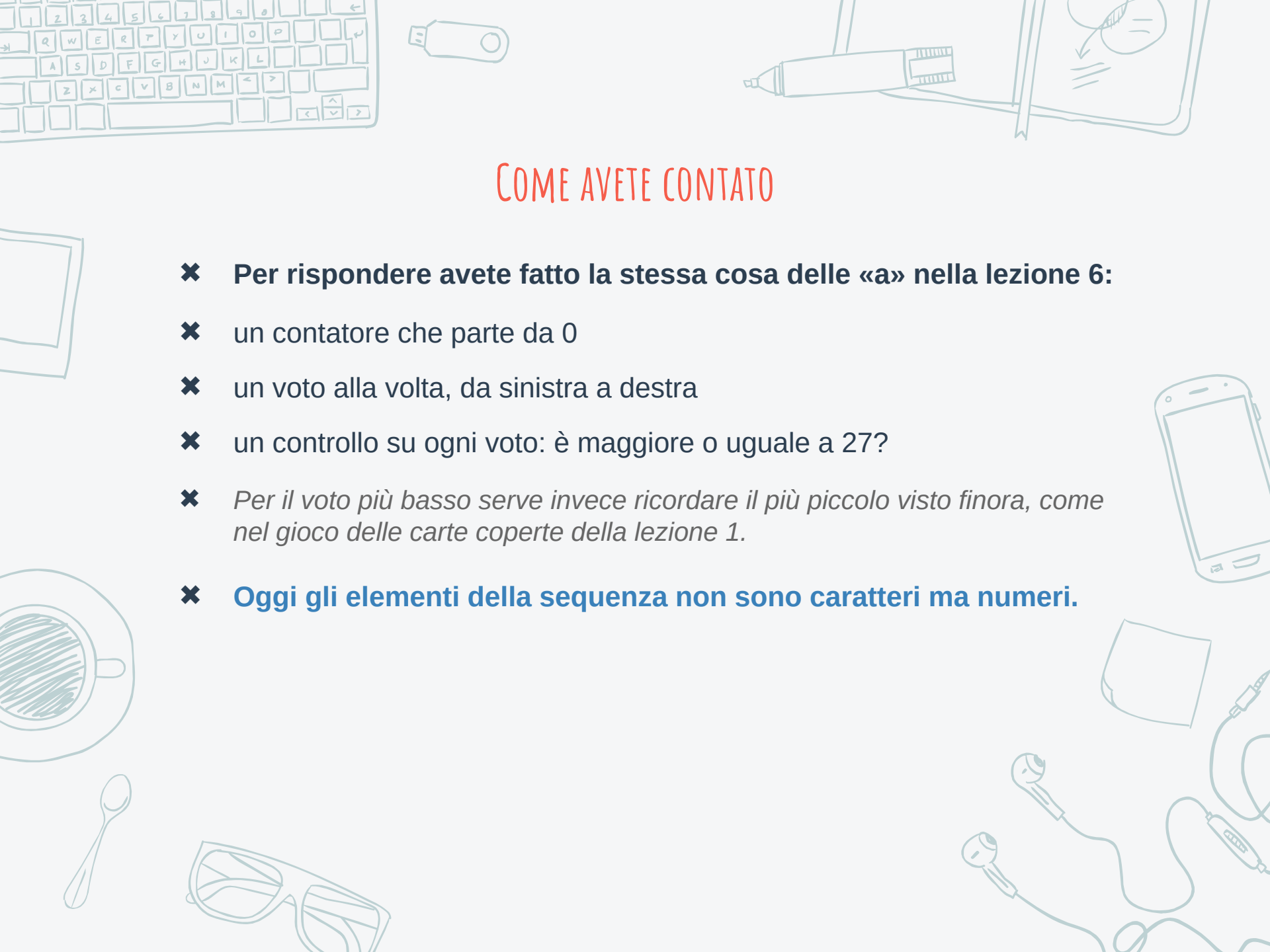




QUANTI ALMENO 27?

24 18 30 27 22 19 30 26 21 28 25 18 29 23 27

Qual è il voto più basso, e quante volte compare?



COME AVETE CONTATO

- ✗ **Per rispondere avete fatto la stessa cosa delle «a» nella lezione 6:**
- ✗ un contatore che parte da 0
- ✗ un voto alla volta, da sinistra a destra
- ✗ un controllo su ogni voto: è maggiore o uguale a 27?
- ✗ *Per il voto più basso serve invece ricordare il più piccolo visto finora, come nel gioco delle carte coperte della lezione 1.*
- ✗ **Oggi gli elementi della sequenza non sono caratteri ma numeri.**



LE LISTE

CHE COS'È UNA LISTA

Una lista è una sequenza di valori con un solo nome. Si scrive fra parentesi quadre.

```
voti = [24, 18, 30, 27]
```

24

18

30

27

0

1

2

3

Lista	Che cosa contiene
[24, 18, 30, 27]	quattro numeri interi
["Ada", "Bruno"]	due stringhe
[7]	un solo elemento
[]	nessun elemento: la lista vuota

Ogni valore della lista si chiama elemento. Come i caratteri di una stringa, gli elementi hanno una posizione che parte da 0.

POSIZIONI E LUNGHEZZA

Si scrive come per le stringhe. Con $v = [24, 18, 30, 27]$:

Scrittura	Significato	Valore
<code>lunghezza(v)</code>	numero di elementi	4
<code>v[0]</code>	primo elemento	24
<code>v[lunghezza(v) - 1]</code>	ultimo elemento	27
<code>v[1:3]</code>	la lista con le posizioni 1 e 2	[18, 30]
<code>v[4]</code>	posizione che non esiste	errore

Una lista lunga n ha le posizioni da 0 a $n - 1$, come una stringa.

STRINGHE E LISTE

Quasi tutto è uguale. Le differenze sono nelle ultime due righe.

	Stringa	Lista
Elementi	caratteri	valori di qualsiasi tipo
Si scrive	"Roma"	[4, 7, 1]
Vuota	""	[]
Lunghezza	lunghezza(s)	lunghezza(v)
Elemento	s[i]	v[i]
Cambiare un elemento	non si può	v[1] = 9 → [4, 9, 1]
Aggiungere in fondo	s = s + "x" (stringa nuova)	v = v + [5] → [4, 9, 1, 5]

Il + fra due liste le unisce, come fra due stringhe. [5] è una lista con un solo elemento: v + [5] mette 5 in fondo. Serve per costruire una lista nuova un elemento alla volta.

SCORRERE UNA LISTA

Le due forme della lezione 6, con «elemento» al posto di «carattere».

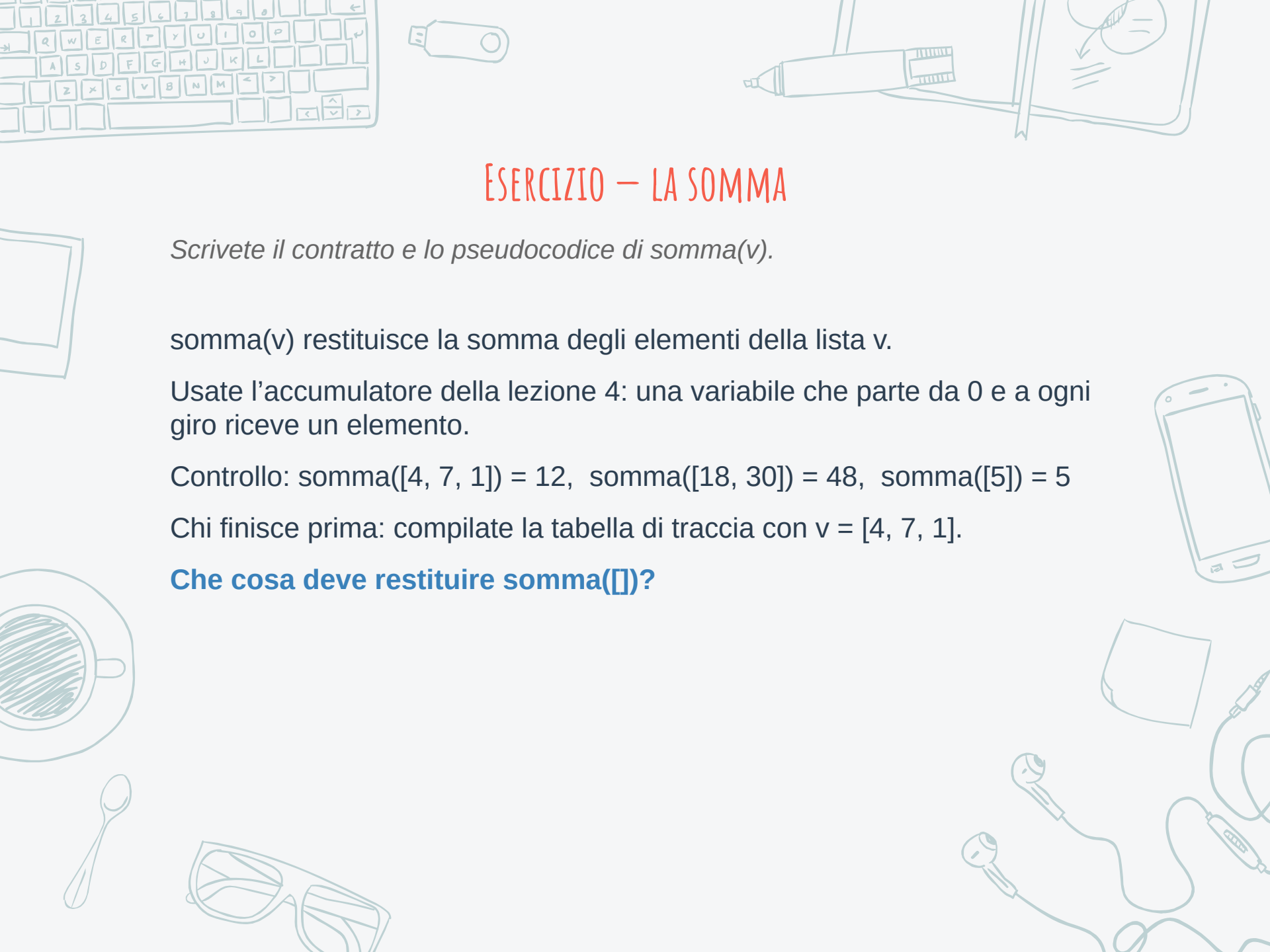
Con la posizione

```
i = 0
fintanto che i < lunghezza(v):
    x = v[i]
    ... esamina x ...
    i = i + 1
```

Senza la posizione

```
per ogni elemento x di v:
    ... esamina x ...
```

Se serve la posizione, per esempio per sapere dove si trova il minimo, si usa la prima forma. Altrimenti la seconda è più corta.



ESERCIZIO — LA SOMMA

Scrivete il contratto e lo pseudocodice di $\text{somma}(v)$.

$\text{somma}(v)$ restituisce la somma degli elementi della lista v .

Usate l'accumulatore della lezione 4: una variabile che parte da 0 e a ogni giro riceve un elemento.

Controllo: $\text{somma}([4, 7, 1]) = 12$, $\text{somma}([18, 30]) = 48$, $\text{somma}([5]) = 5$

Chi finisce prima: compilate la tabella di traccia con $v = [4, 7, 1]$.

Che cosa deve restituire $\text{somma}()$?

SOLUZIONE – SOMMA

Contratto: somma(v) riceve una lista di numeri e restituisce la loro somma.

```
funzione somma(v):  
    s = 0  
    per ogni elemento x di v:  
        s = s + x  
    restituisci s
```

Python

```
def somma(v):  
    s = 0  
    for x in v:  
        s = s + x  
    return s
```

Con v = [4, 7, 1]

x	s
—	0
4	4
7	11
1	12

somma([]) restituisce 0: il ciclo non parte.

È l'accumulatore della lezione 4 (somma dei primi n numeri): cambia solo da dove arrivano i valori da sommare.

ESERCIZIO — LA MEDIA

Scrivete il contratto e lo pseudocodice di $\text{media}(v)$ e di $\text{conta_sopra}(v)$. Usate somma, non riscrivetela.

- ✖ $\text{media}(v)$ restituisce la media aritmetica degli elementi di v .
 - Controllo: $\text{media}([4, 7, 1]) = 4$, $\text{media}([18, 30]) = 24$
- ✖ $\text{conta_sopra}(v)$ restituisce quanti elementi sono maggiori della media.
 - Controllo: $\text{conta_sopra}([4, 7, 1]) = 1$, $\text{conta_sopra}([5, 5, 5]) = 0$,
 $\text{conta_sopra}([24, 18, 30, 27, 22]) = 2$
- ✖ **Che cosa succede con $\text{media}([])$? Scrivetelo nel contratto.**

SOLUZIONE – MEDIA

Contratto: media(v) riceve una lista non vuota di numeri e restituisce la loro media.

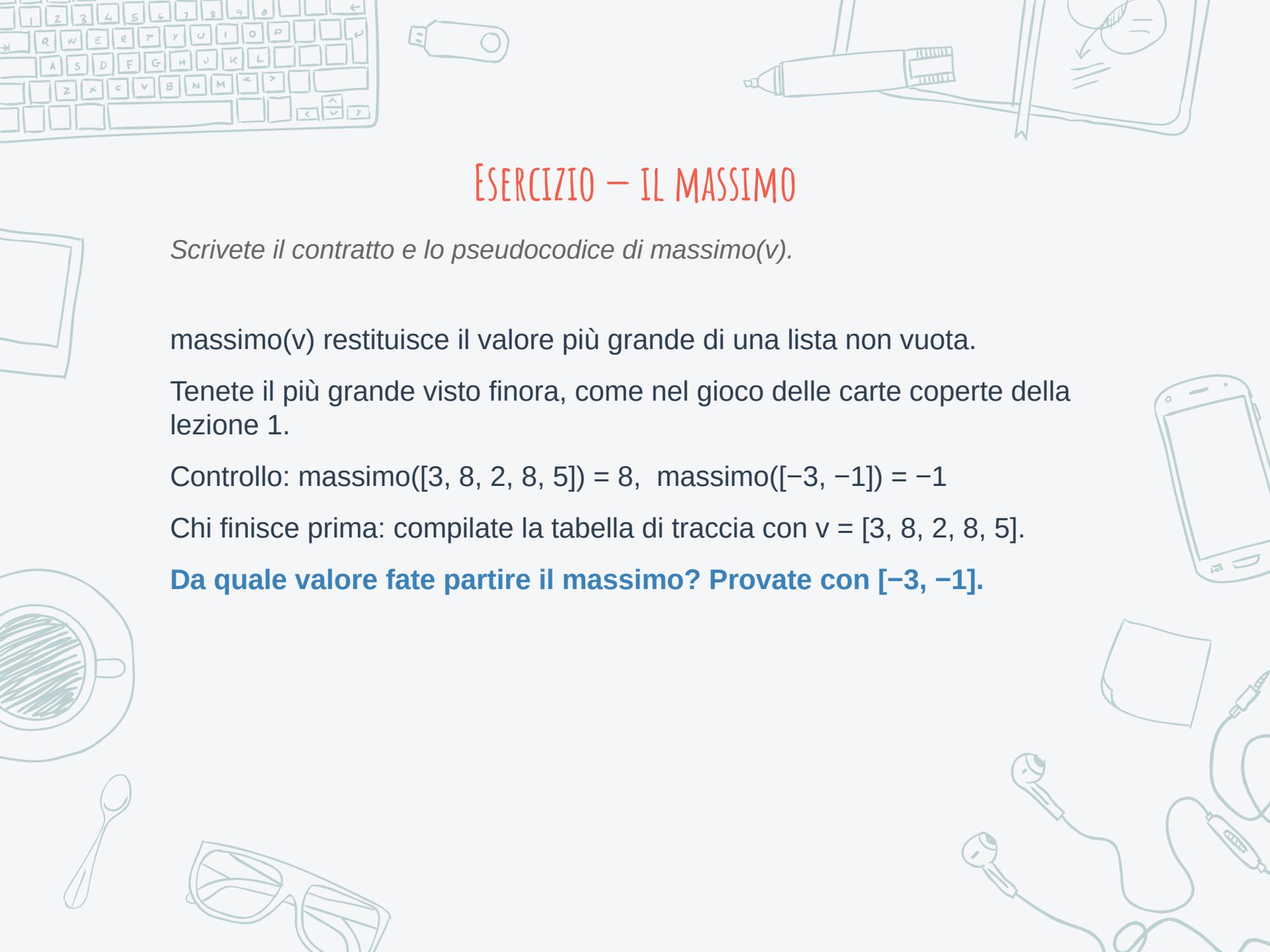
Pseudocodice

```
funzione media(v):  
    restituisci somma(v) / lunghezza(v)  
  
funzione conta_sopra(v):  
    m = media(v)  
    n = 0  
    per ogni elemento x di v:  
        se x > m allora  
            n = n + 1  
    restituisci n
```

Python

```
def media(v):  
    return somma(v) / len(v)  
  
def conta_sopra(v):  
    m = media(v)  
    n = 0  
    for x in v:  
        if x > m:  
            n = n + 1  
    return n
```

La media si calcola una volta, prima del ciclo. Dentro il ciclo verrebbe ricalcolata per ogni elemento: con 1000 elementi, 1000 somme di 1000 numeri.



ESERCIZIO — IL MASSIMO

Scrivete il contratto e lo pseudocodice di `massimo(v)`.

`massimo(v)` restituisce il valore più grande di una lista non vuota.

Tenete il più grande visto finora, come nel gioco delle carte coperte della lezione 1.

Controllo: `massimo([3, 8, 2, 8, 5]) = 8`, `massimo([-3, -1]) = -1`

Chi finisce prima: compilate la tabella di traccia con `v = [3, 8, 2, 8, 5]`.

Da quale valore fate partire il massimo? Provate con `[-3, -1]`.

SOLUZIONE – MASSIMO

Contratto: massimo(v) riceve una lista non vuota di numeri e restituisce il più grande.

```
funzione massimo(v):  
    m = v[0]  
    per ogni elemento x di v:  
        se x > m allora  
            m = x  
    restituisci m
```

Python

```
def massimo(v):  
    m = v[0]  
    for x in v:  
        if x > m:  
            m = x  
    return m
```

Con $v = [3, 8, 2, 8, 5]$

x	x > m ?	m
—	—	3
3	no	3
8	sì	8
2	no	8
8	no	8
5	no	8

m parte dal primo elemento, non da 0: con $[-3, -1]$ partendo da 0 si otterrebbe 0, che non è nella lista.

È il gioco delle carte coperte della lezione 1: si tiene in mano la carta migliore vista finora.

ESERCIZIO — DOV'È IL MINIMO?

Scrivete il contratto e lo pseudocodice di $\text{pos_minimo}(v)$.

- ✘ $\text{pos_minimo}(v)$ restituisce la posizione dell'elemento più piccolo, non il suo valore.
- ✘ Serve la scansione con la posizione i .
- ✘ Controllo: $\text{pos_minimo}([5, 2, 9, 7]) = 1$, $\text{pos_minimo}([7]) = 0$,
 $\text{pos_minimo}([4, 3, 2, 1]) = 3$
- ✘ **Con $\text{pos_minimo}([5, 2, 9, 2])$ la vostra funzione restituisce 1 o 3?**

SOLUZIONE – MINIMO

Contratto: `pos_minimo(v)` riceve una lista non vuota e restituisce la posizione del primo minimo.

Pseudocodice

```
funzione pos_minimo(v):  
    p = 0  
    i = 1  
    fintanto che i < lunghezza(v):  
        se v[i] < v[p] allora  
            p = i  
        i = i + 1  
    restituisci p
```

Python

```
def pos_minimo(v):  
    p = 0  
    i = 1  
    while i < len(v):  
        if v[i] < v[p]:  
            p = i  
        i = i + 1  
    return p
```

v	[5, 2, 9, 7]	[7]	[4, 3, 2, 1]	[5, 2, 9, 2]
pos_minimo(v)	1	0	3	1

Conoscendo la posizione si ricava anche il valore: `v[pos_minimo(v)]`.

ESEMPIO — COSTRUIRE UNA LISTA

Lo svolgiamo insieme: una lista nuova con gli elementi di v maggiori di soglia.

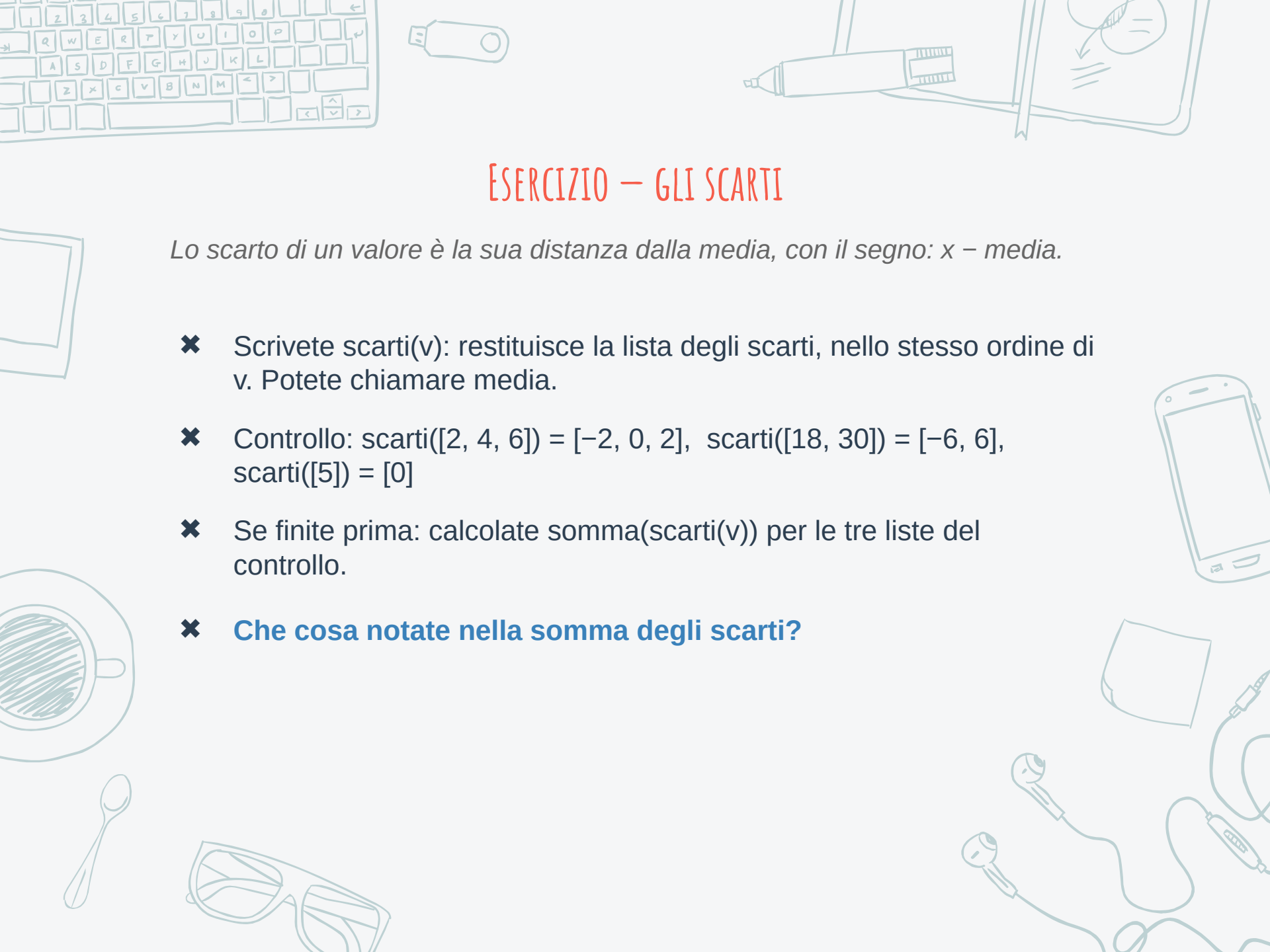
```
funzione sopra_soglia(v, soglia):  
    r = []  
    per ogni elemento x di v:  
        se x > soglia allora  
            r = r + [x]  
    restituisci r
```

È toglì_spazi della lezione 6: al posto di $r = ""$ c'è $r = []$, al posto di $r = r + c$ c'è $r = r + [x]$.

Con $v = [20, 27, 18, 30]$ e soglia = 25

x	20	27	18	30
r	[]	[27]	[27]	[27, 30]

La lista v non cambia: il risultato è una lista nuova.



ESERCIZIO — GLI SCARTI

Lo scarto di un valore è la sua distanza dalla media, con il segno: $x - \text{media}$.

- ✘ Scrivete `scarti(v)`: restituisce la lista degli scarti, nello stesso ordine di `v`. Potete chiamare `media`.
- ✘ Controllo: `scarti([2, 4, 6]) = [-2, 0, 2]`, `scarti([18, 30]) = [-6, 6]`, `scarti([5]) = [0]`
- ✘ Se finite prima: calcolate `somma(scarti(v))` per le tre liste del controllo.
- ✘ **Che cosa notate nella somma degli scarti?**

SOLUZIONE – SCARTI

Contratto: `scarti(v)` riceve una lista non vuota e restituisce una lista con gli scarti dalla media.

Pseudocodice

```
funzione scarti(v):  
    m = media(v)  
    r = []  
    per ogni elemento x di v:  
        r = r + [x - m]  
    restituisci r
```

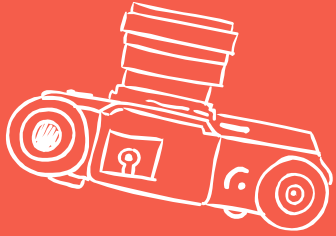
Python

```
def scarti(v):  
    m = media(v)  
    r = []  
    for x in v:  
        r = r + [x - m]  
    return r
```

v	[2, 4, 6]	[18, 30]	[5]
scarti(v)	[-2, 0, 2]	[-6, 6]	[0]
somma degli scarti	0	0	0

[x - m] è una lista con un solo elemento: il + la attacca in fondo a r.

LA LISTA È IN
ORDINE?



IN ORDINE CRESCENTE

crescente(v) dice se ogni elemento è maggiore o uguale al precedente.

funzione crescente(v):

 i = 1

 fintanto che i < lunghezza(v):

 se v[i] < v[i-1] allora

 restituisce falso

 i = i + 1

 restituisce vero

Con v = [1, 3, 3, 7], n = 4

1	3	3	7
0	1	2	3

i	i < n ?	v[i-1] v[i]	v[i] < v[i-1] ?	Che cosa succede
1	Vero	1 3	no	i = 2
2	Vero	3 3	no	i = 3
3	Vero	3 7	no	i = 4
4	Falso	—	—	restituisce vero

Si parte da i = 1 perché il primo elemento non ha un precedente.

PROVATE VOI

Stessa funzione, con $v = [2, 5, 6, 6, 4]$. Completate la tabella e dite che cosa restituisce.

2	5	6	6	4
0	1	2	3	4

i	$i < n$?	$v[i-1]$ $v[i]$	$v[i] < v[i-1]$?	Che cosa succede

Se finite prima: con $v = [9, 1, 2, 3, 4]$, quante righe ha la tabella?

SOLUZIONE

i	i < n ?	v[i-1] v[i]	v[i] < v[i-1] ?	Che cosa succede
1	Vero	2 5	no	i = 2
2	Vero	5 6	no	i = 3
3	Vero	6 6	no	i = 4
4	Vero	6 4	sì	restituisce falso

Con [9, 1, 2, 3, 4] la funzione si ferma alla prima riga ($i = 1$), senza guardare il resto.

Per rispondere «vero» la funzione deve guardare tutte le coppie. Per rispondere «falso» le basta trovarne una sbagliata.

VERSO PYTHON



FUNZIONI GIÀ PRONTE

Molte funzioni scritte nel corso esistono già in Python. Si usano conoscendone il contratto.

Nel corso	In Python	Contratto
lunghezza(v)	<code>len(v)</code>	numero di elementi o di caratteri
somma(v)	<code>sum(v)</code>	somma degli elementi; con [] restituisce 0
massimo(v)	<code>max(v)</code>	elemento più grande; con [] dà errore
minimo(v)	<code>min(v)</code>	elemento più piccolo; con [] dà errore
inverti(s)	<code>s[::-1]</code>	la stringa al contrario (lezione 6)
conta(v, x)	<code>v.count(x)</code>	quante volte x compare in v
–	<code>sorted(v)</code>	una lista nuova con gli elementi in ordine

Sapere come sono fatte dentro aiuta a prevederne il comportamento: `max([])` dà errore per lo stesso motivo per cui `massimo(v)` chiede una lista non vuota.

UN PROGRAMMA COMPLETO

I voti del riscaldamento, riassunti in Python con le funzioni già pronte.

```
voti = [24, 18, 30, 27, 22, 19, 30, 26,  
        21, 28, 25, 18, 29, 23, 27]  
  
print("Numero di voti:", len(voti))  
print("Minimo:", min(voti))  
print("Massimo:", max(voti))  
print("Media:", round(sum(voti) / len(voti), 2))  
print("Almeno 27:", len([x for x in voti if x >= 27]))
```

Sullo schermo

```
Numero di voti: 15  
Minimo: 18  
Massimo: 30  
Media: 24.47  
Almeno 27: 6
```

L'ultima riga costruisce la lista dei voti ≥ 27 e ne prende la lunghezza: è sopra_soglia scritta in una riga.



RIEPILOGO

✖ Lista

- Sequenza di valori con posizioni da 0 a lunghezza - 1. Si può modificare; con + se ne costruisce una nuova.

✖ Schemi già noti, applicati alle liste

- accumulatore (somma), contatore (conta_sopra), massimo e minimo (pos_minimo), costruzione di una lista nuova (sopra_soglia, scarti), uscita anticipata (crescente).

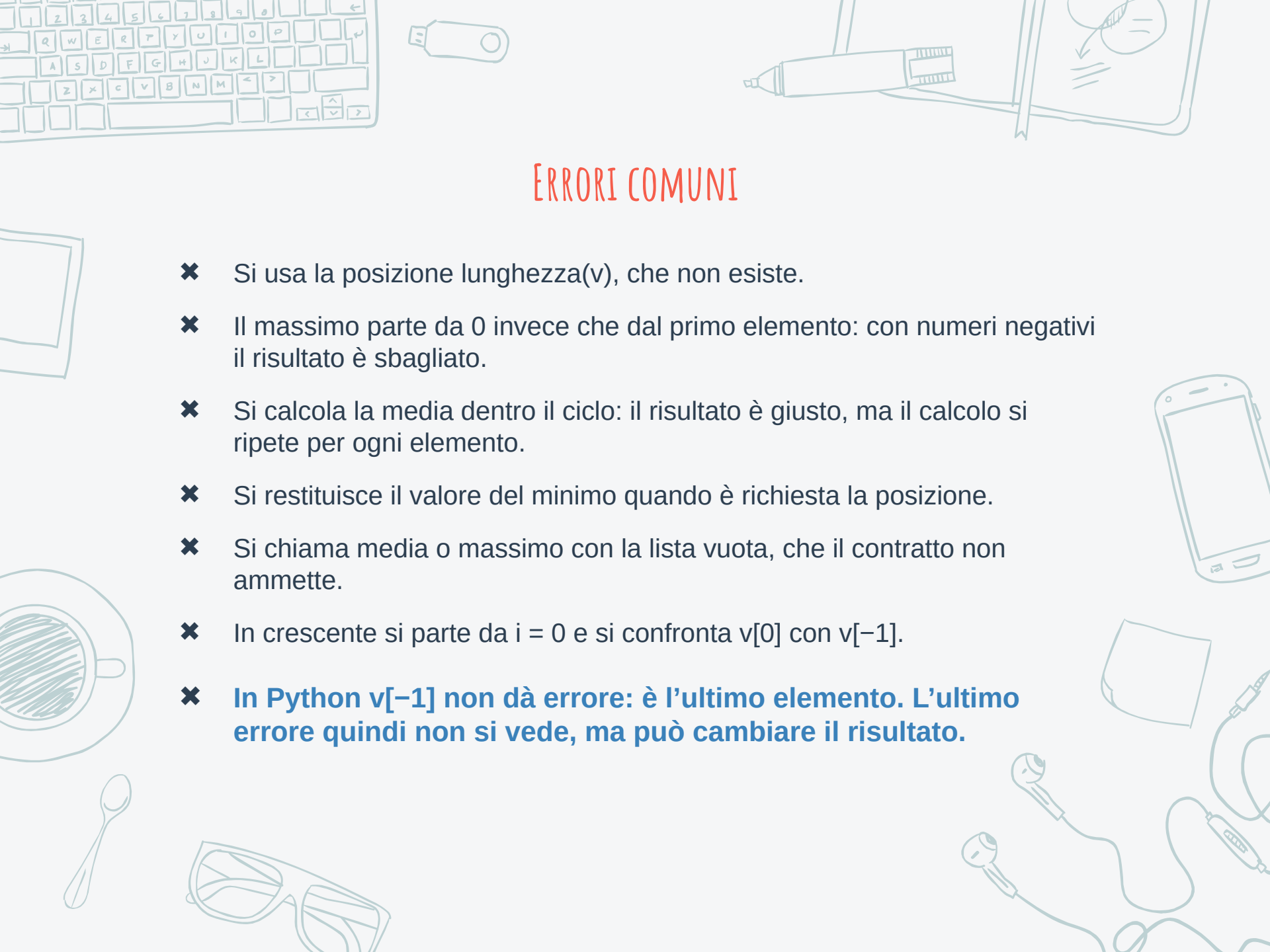
✖ Funzioni

- media chiama somma; conta_sopra e scarti chiamano media.

✖ Efficienza

- Un valore che non cambia durante il ciclo si calcola una volta, prima del ciclo.

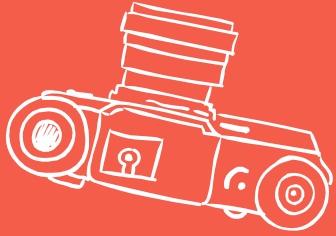
✖ Con numeri, stringhe e liste gli schemi sono sempre gli stessi.

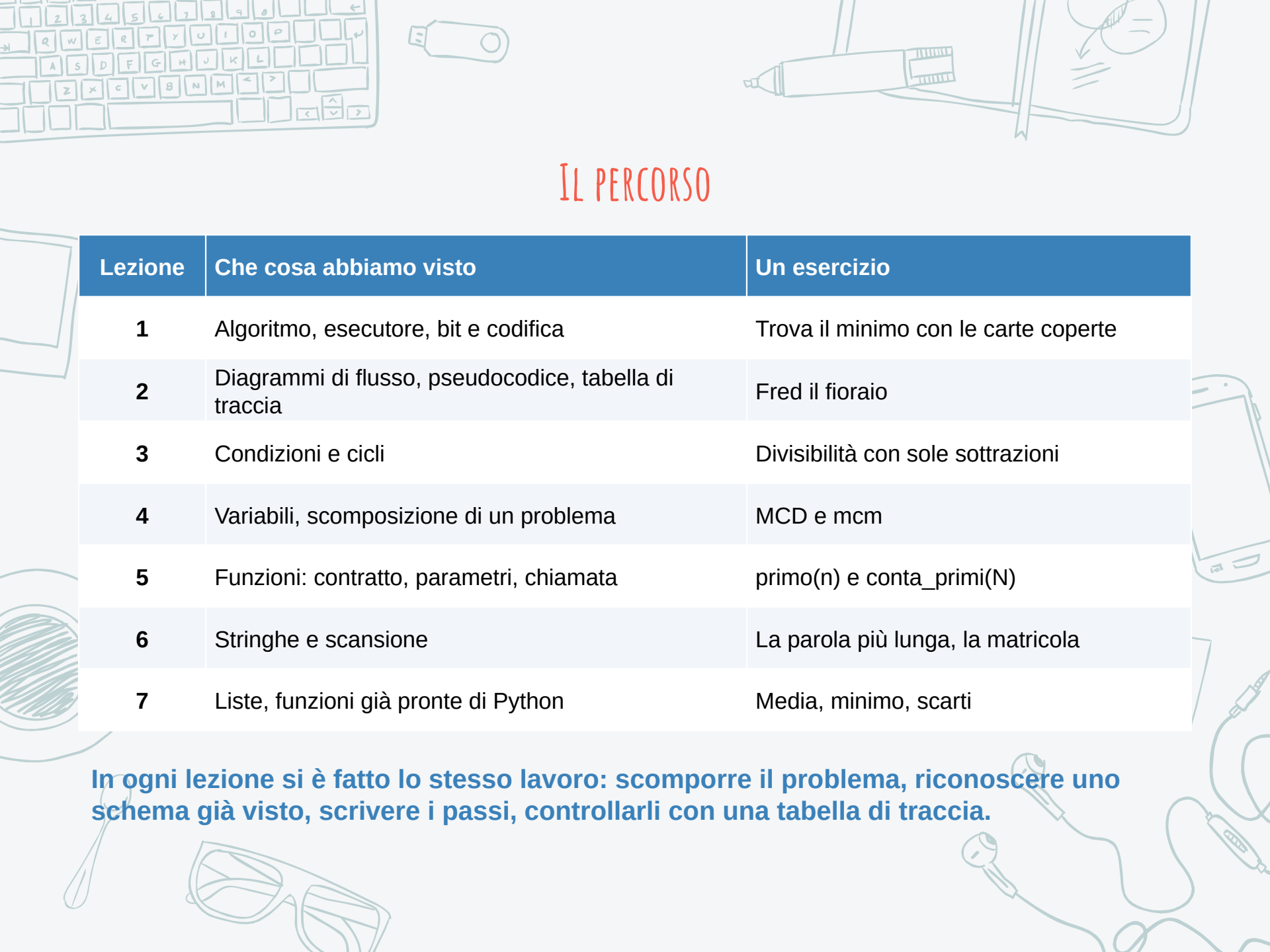


ERRORI COMUNI

- ✘ Si usa la posizione `lunghezza(v)`, che non esiste.
- ✘ Il massimo parte da 0 invece che dal primo elemento: con numeri negativi il risultato è sbagliato.
- ✘ Si calcola la media dentro il ciclo: il risultato è giusto, ma il calcolo si ripete per ogni elemento.
- ✘ Si restituisce il valore del minimo quando è richiesta la posizione.
- ✘ Si chiama `media` o `massimo` con la lista vuota, che il contratto non ammette.
- ✘ In crescente si parte da $i = 0$ e si confronta `v[0]` con `v[-1]`.
- ✘ **In Python `v[-1]` non dà errore: è l'ultimo elemento. L'ultimo errore quindi non si vede, ma può cambiare il risultato.**

IL CORSO IN SETTE LEZIONI





IL PERCORSO

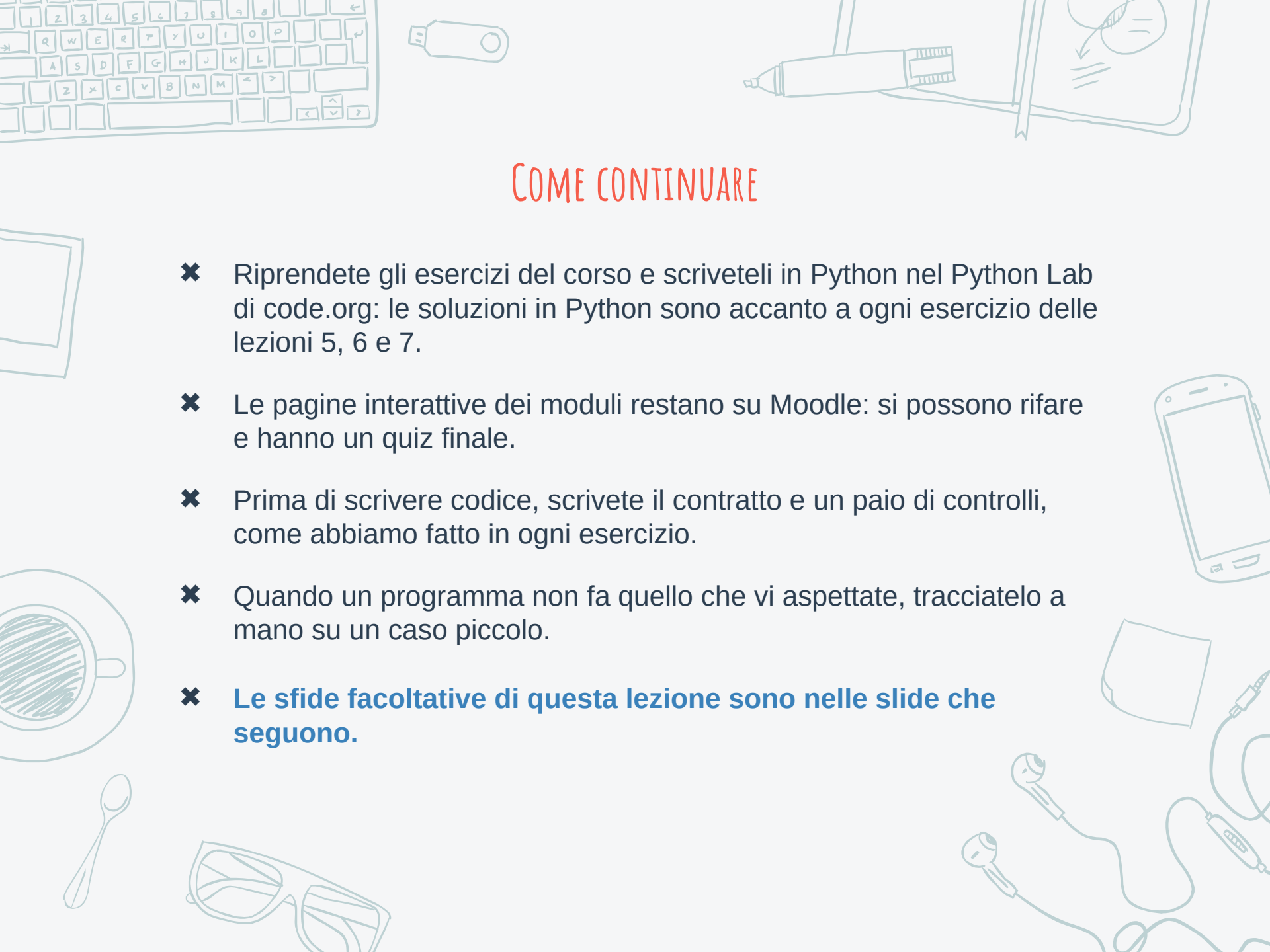
Lezione	Che cosa abbiamo visto	Un esercizio
1	Algoritmo, esecutore, bit e codifica	Trova il minimo con le carte coperte
2	Diagrammi di flusso, pseudocodice, tabella di traccia	Fred il fioraio
3	Condizioni e cicli	Divisibilità con sole sottrazioni
4	Variabili, scomposizione di un problema	MCD e mcm
5	Funzioni: contratto, parametri, chiamata	primo(n) e conta_primi(N)
6	Stringhe e scansione	La parola più lunga, la matricola
7	Liste, funzioni già pronte di Python	Media, minimo, scarti

In ogni lezione si è fatto lo stesso lavoro: scomporre il problema, riconoscere uno schema già visto, scrivere i passi, controllarli con una tabella di traccia.

UNA DOMANDA PER LEZIONE

Rispondete in chat con il numero della domanda e la risposta.

	Domanda
1	Quanti valori diversi si rappresentano con 3 bit?
2	$x = 3$; $x = x + 2$; scrivi x . Che cosa compare sullo schermo?
3	$i = 0$; fintanto che $i < 5$: $i = i + 2$. Quante volte si esegue il corpo del ciclo?
4	MCD con sole sottrazioni: da (12, 8) si toglie il minore dal maggiore finché sono uguali. Quanto vale il MCD?
5	primo(9): quante volte chiama divisibile?
6	$s = \text{"informatica"}$. Quanto vale $s[2:5]$?
7	$v = [4, 7, 1]$. Quanto vale $v[\text{lunghezza}(v) - 1]$?

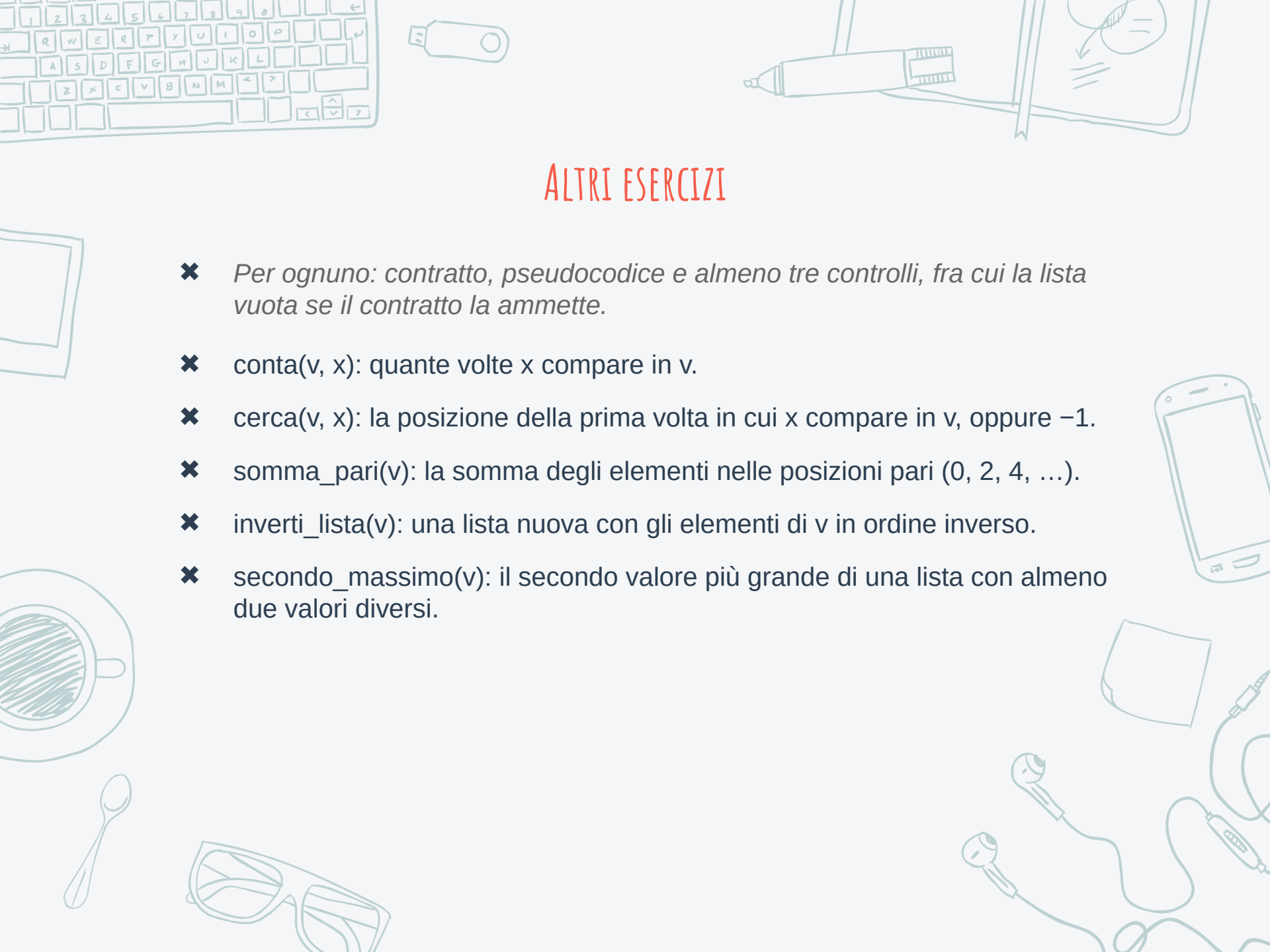


COME CONTINUARE

- ✘ Riprendete gli esercizi del corso e scriveteli in Python nel Python Lab di code.org: le soluzioni in Python sono accanto a ogni esercizio delle lezioni 5, 6 e 7.
- ✘ Le pagine interattive dei moduli restano su Moodle: si possono rifare e hanno un quiz finale.
- ✘ Prima di scrivere codice, scrivete il contratto e un paio di controlli, come abbiamo fatto in ogni esercizio.
- ✘ Quando un programma non fa quello che vi aspettate, tracciatelo a mano su un caso piccolo.
- ✘ **Le sfide facoltative di questa lezione sono nelle slide che seguono.**

SFIDE FACOLTATIVE





ALTRI ESERCIZI

- ✘ *Per ognuno: contratto, pseudocodice e almeno tre controlli, fra cui la lista vuota se il contratto la ammette.*
- ✘ `conta(v, x)`: quante volte `x` compare in `v`.
- ✘ `cerca(v, x)`: la posizione della prima volta in cui `x` compare in `v`, oppure `-1`.
- ✘ `somma_pari(v)`: la somma degli elementi nelle posizioni pari (0, 2, 4, ...).
- ✘ `inverti_lista(v)`: una lista nuova con gli elementi di `v` in ordine inverso.
- ✘ `secondo_massimo(v)`: il secondo valore più grande di una lista con almeno due valori diversi.

ALTRI ESERCIZI – SOLUZIONI

Contatore, ricerca con uscita anticipata e un passo diverso da 1.

conta(v, x)

```
funzione conta(v, x):  
    n = 0  
    per ogni elemento y di v:  
        se y = x allora  
            n = n + 1  
    restituisci n
```

cerca(v, x)

```
funzione cerca(v, x):  
    i = 0  
    fintanto che i < lunghezza(v):  
        se v[i] = x allora  
            restituisci i  
        i = i + 1  
    restituisci -1
```

somma_pari(v)

```
funzione somma_pari(v):  
    s = 0  
    i = 0  
    fintanto che i < lunghezza(v):  
        s = s + v[i]  
        i = i + 2  
    restituisci s
```

Controlli:

conta([1, 2, 1, 1], 1) = 3

cerca([4, 7, 1], 7) = 1

cerca([4, 7, 1], 5) = -1

somma_pari([5, 1, 4, 2, 3]) = 12

Con la lista vuota: 0, -1, 0.

ALTRI ESERCIZI – SOLUZIONI (2)

Costruire una lista nuova e riusare una funzione già scritta.

inverti_lista(v)

```
funzione inverti_lista(v):  
    r = []  
    per ogni elemento x di v:  
        r = [x] + r  
    restituisci r
```

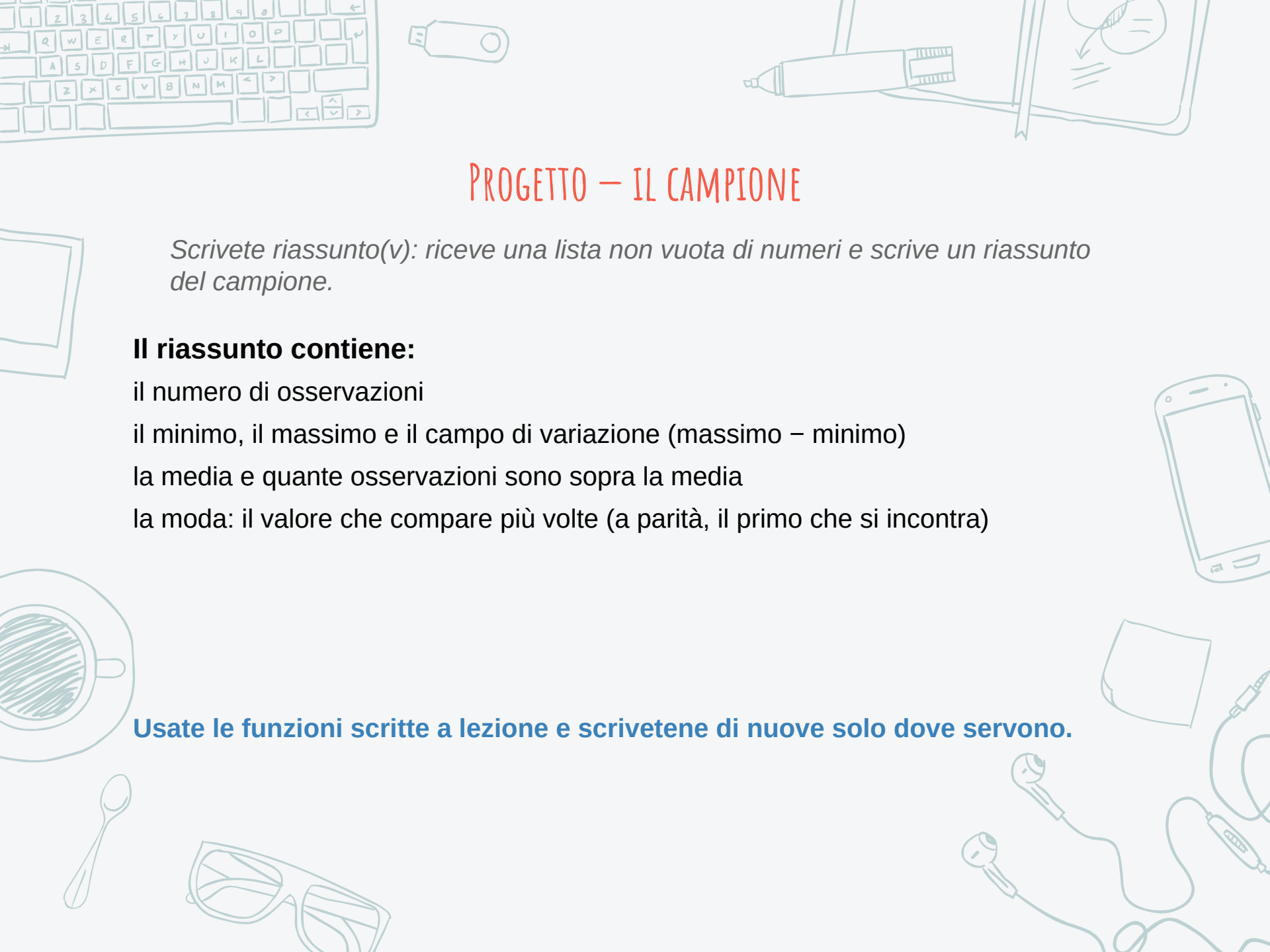
Come inverti(s) nella lezione 6: ogni elemento nuovo va davanti. $\text{inverti_lista}([4, 7, 1]) = [1, 7, 4]$.

secondo_massimo(v)

```
funzione secondo_massimo(v):  
    m = massimo(v)  
    r = []  
    per ogni elemento x di v:  
        se  $x \neq m$  allora  
            r = r + [x]  
    restituisci massimo(r)
```

Si tolgono tutte le copie del massimo e si cerca il massimo di quello che resta. $\text{secondo_massimo}([3, 8, 2, 8, 5]) = 5$.

Il contratto chiede almeno due valori diversi: altrimenti r resta vuota e massimo(r) non si può calcolare.



PROGETTO — IL CAMPIONE

Scrivete riassunto(v): riceve una lista non vuota di numeri e scrive un riassunto del campione.

Il riassunto contiene:

il numero di osservazioni

il minimo, il massimo e il campo di variazione (massimo – minimo)

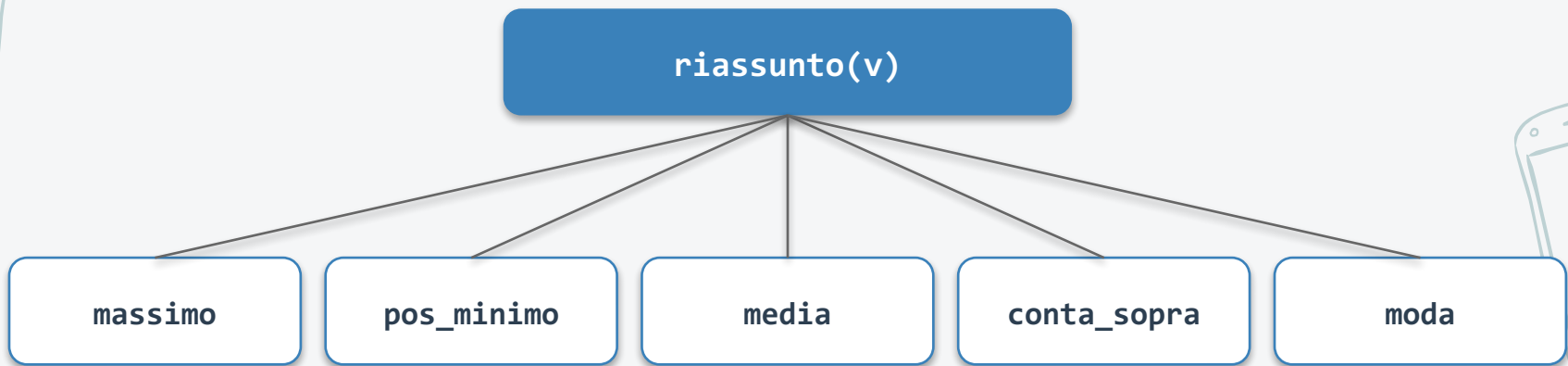
la media e quante osservazioni sono sopra la media

la moda: il valore che compare più volte (a parità, il primo che si incontra)

Usate le funzioni scritte a lezione e scrivetene di nuove solo dove servono.

CAMPIONE — SCOMPORRE

Prima dello pseudocodice, dividete il problema in funzioni.



Per ogni funzione scrivete il contratto. Il minimo si ricava da `pos_minimo`: `v[pos_minimo(v)]`. `moda` può chiamare `conta(v, x)` degli altri esercizi.

Quattro funzioni su cinque le avete già scritte a lezione.

CAMPIONE — CASI DI PROVA

v	n	min	max	campo	media	sopra	moda
[24, 18, 30, 27, 22]	5	18	30	12	24,2	2	24
[2, 4, 6]	3	2	6	4	4	1	2
[5]	1	5	5	0	5	0	5
[7, 7, 7]	3	7	7	0	7	0	7
[3, 1, 3, 2, 1, 3]	6	1	3	2	2,17	3	3

La media di [3, 1, 3, 2, 1, 3] è $13 / 6 = 2,1666\dots$: in tabella è arrotondata.

Consegna: diagramma di flusso o pseudocodice, e i casi di prova. Il Python è facoltativo.

CAMPIONE – SOLUZIONE

Due funzioni nuove: conta viene dagli altri esercizi, moda e riassunto sono qui.

moda(v)

```
funzione moda(v):  
    mo = v[0]  
    c = conta(v, v[0])  
    per ogni elemento x di v:  
        se conta(v, x) > c allora  
            mo = x  
            c = conta(v, x)  
    restituisci mo
```

È lo schema del massimo: al posto del valore si confronta quante volte compare. Con «>» a parità resta il primo.

riassunto(v)

```
funzione riassunto(v):  
    mi = v[pos_minimo(v)]  
    ma = massimo(v)  
    scrivi "osservazioni:", lunghezza(v)  
    scrivi "minimo:", mi, "massimo:", ma  
    scrivi "campo:", ma - mi  
    scrivi "media:", media(v)  
    scrivi "sopra la media:", conta_sopra(v)  
    scrivi "moda:", moda(v)
```

riassunto non calcola niente da sola: chiama le funzioni scritte a lezione.

CAMPIONE IN PYTHON

Le due funzioni nuove in Python. Le altre sono quelle delle soluzioni della lezione.

```
def moda(v):  
    mo = v[0]  
    c = conta(v, v[0])  
    for x in v:  
        if conta(v, x) > c:  
            mo = x  
            c = conta(v, x)  
    return mo
```

```
def riassunto(v):  
    mi = v[pos_minimo(v)]  
    ma = massimo(v)  
    print("osservazioni:", len(v))  
    print("minimo:", mi, "massimo:", ma)  
    print("campo:", ma - mi)  
    print("media:", round(media(v), 2))  
    print("sopra la media:", conta_sopra(v))  
    print("moda:", moda(v))
```

riassunto([24, 18, 30, 27, 22]) scrive:

```
osservazioni: 5  
minimo: 18 massimo: 30  
campo: 12  
media: 24.2  
sopra la media: 2  
moda: 24
```

round(x, 2) arrotonda a due cifre dopo la virgola: la media di [3, 1, 3, 2, 1, 3] viene scritta come 2.17.