

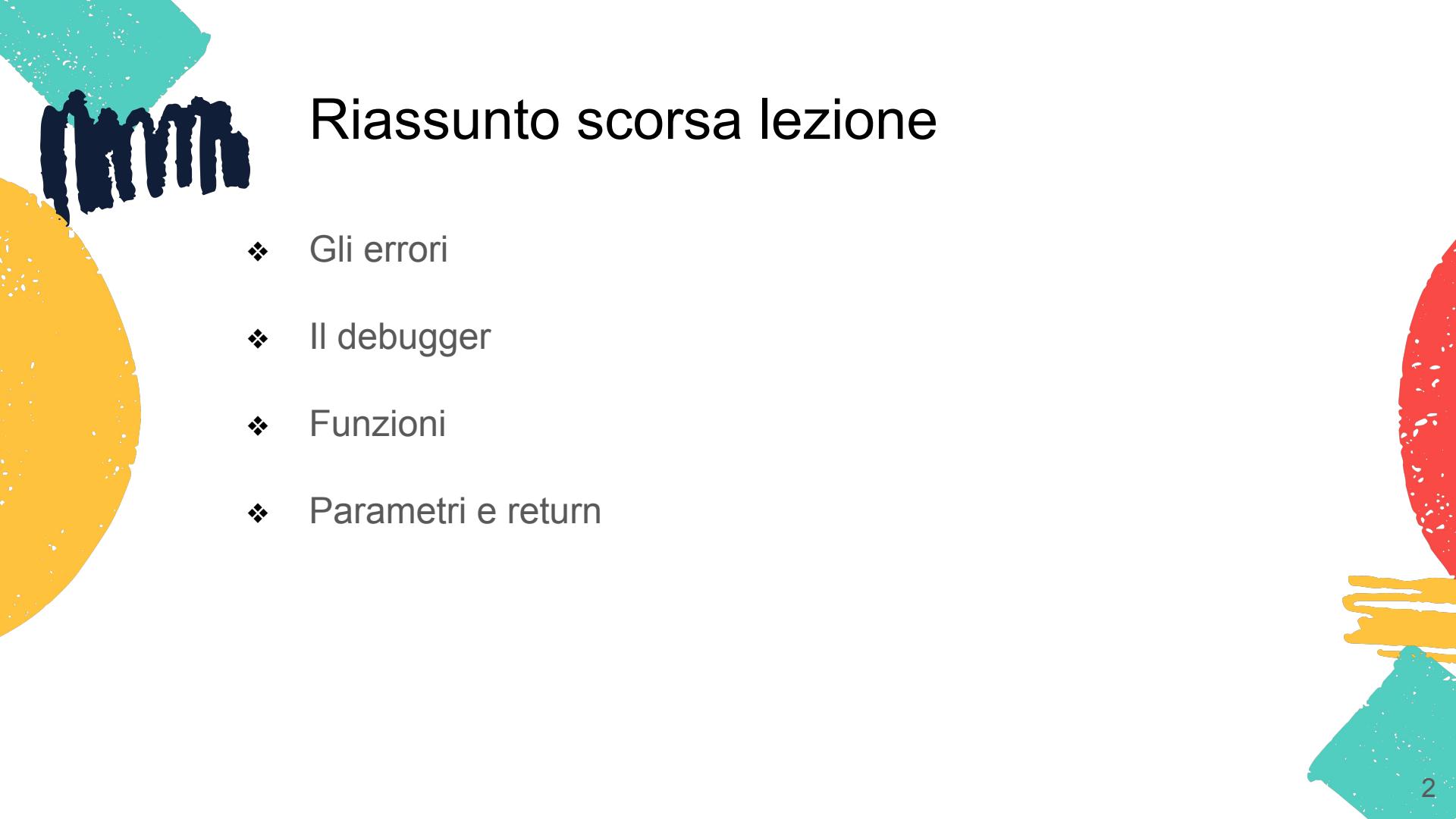
CORSO PYTHON

Quinta parte

Liste ed operatori sulle liste

Carlo Tacchella





Riassunto scorsa lezione

- ❖ Gli errori
- ❖ Il debugger
- ❖ Funzioni
- ❖ Parametri e return

Gli errori

Esistono due tipi di errori

- **Errori sintattici** : Abbiamo commesso un errore di scrittura nel codice.
Python li segnala con una riga rossa.
- **Errori semantici** : Il programma non si comporta come dovrebbe, ovvero il
programma non fa quello che dovrebbe. Non sono individuati da Python.



Il debugger

Siccome gli errori di tipo semantico non sono identificati da Python, spesso dobbiamo cercarli noi stessi; per farlo possiamo usare uno strumento chiamato “debugger” che ci aiuta nell’identificare gli errori di funzionamento.

Il debugger di Python ci permette di eseguire una linea di codice alla volta e ci permette di visualizzare i valori che le variabili assumono durante l’esecuzione del codice.



Le funzioni

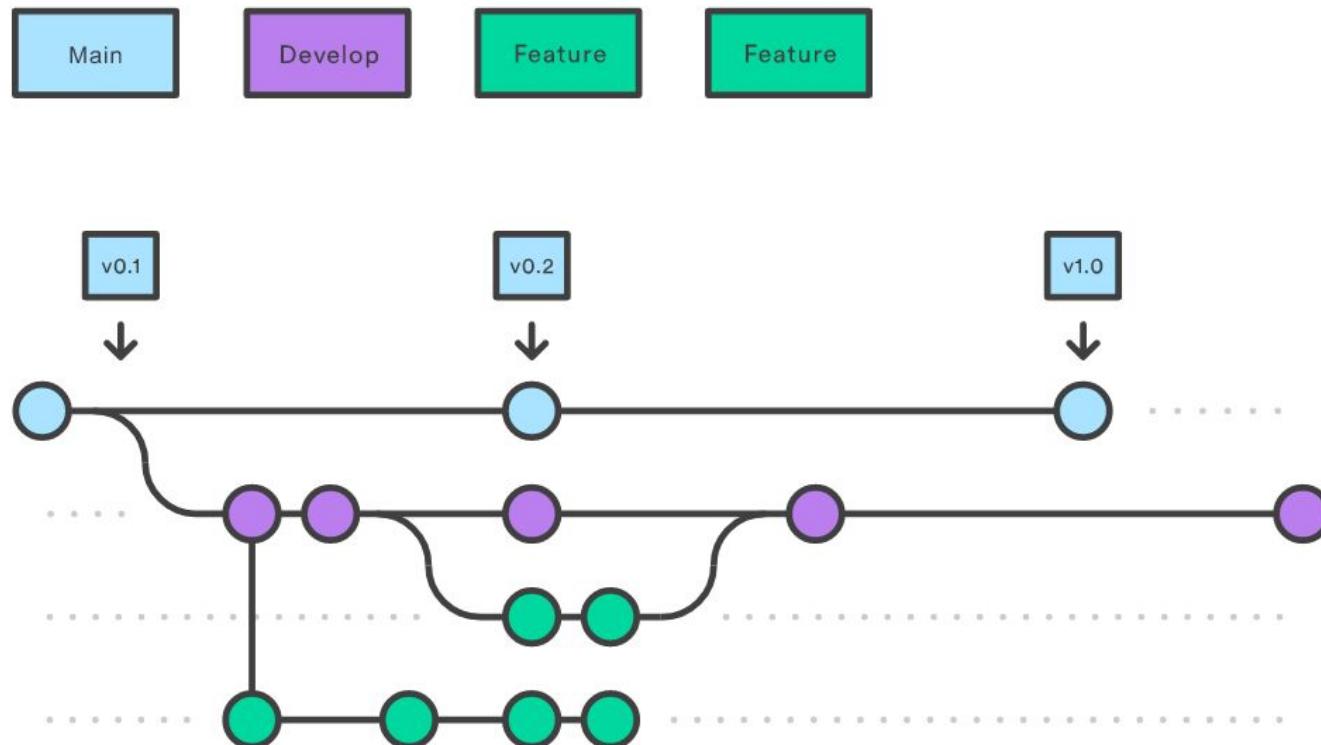
Una funzione consiste nel raggruppare alcune linee di codice sotto un solo nome per poterle utilizzare molte volte senza doverle riscrivere.

Inoltre le funzioni possono interfacciarsi con il resto del codice con un ingresso (parametri) e con un'uscita (return).

I parametri ci permettono di fare in modo che la funzione svolga calcoli differenti o si comporti in maniera diversa quando ne abbiamo bisogno.

Il valore di return invece permette alla funzione di comunicare un valore al nostro programma.

L'approccio GitFlow

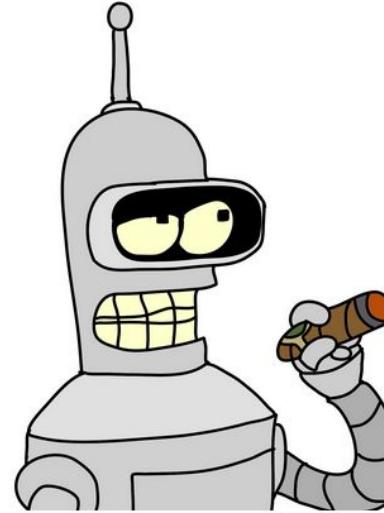




Tocca a voi!

Creiamo il main, sopra il main creiamo il dev

Sopra al dev creo un branch tutto mio, con blackjack e commit di lusso



Le liste

Abbiamo visto che le funzioni ci permettono di raggruppare linee di codice, ma spesso ci può far comodo anche raggruppare variabili.

Se ad esempio volessimo memorizzare l'età di 10 persone diverse potremmo scrivere:

```
anni1= 5  
anni2= 23  
anni3= 24  
anni4= 50  
anni5= 19  
anni6= 42
```

Le liste

Tuttavia questo è davvero molto scomodo e ripetitivo.

Le liste ci vengono dunque incontro permettendoci di essere molto più rapidi e ordinati; lo stesso codice di prima lo possiamo infatti scrivere come:

```
anni = [5, 23, 24, 50, 19, 42]
```

e scrivere `anni[2]` corrisponde a scrivere `anni3` (L'enumerazione delle liste parte infatti da 0).

`anni[0]` è 5

`anni[1]` è 23

Metaforicamente parlando, una lista può essere vista come un insieme di cassetti.



Le liste

Inoltre una lista può essere formata anche da oggetti di tipi differenti.

```
miscellanea = ['ok',1,0.90,'pippo',[1,2,3]]
```

Python è uno dei pochi linguaggi che permette di utilizzare liste formate da oggetti di tipi diversi. Sebbene questa possibilità permetta di diminuire molto gli errori sintattici, aumenta molto il verificarsi di errori semantici, quindi fate moltissima attenzione quando usate liste miste.



Le liste

La sintassi per **definire** una lista è la seguente:

```
nome_lista = [ elem1, elem2, elem3, ... , elemN]
```

La sintassi per **accedere** al valore di una lista è la seguente

```
nome_lista [posizione]
```

Per **modificare** il valore in posizione pos di una lista possiamo scrivere

```
nome_lista [pos] = nuovo_valore
```

Le liste

La posizione della lista può essere un qualsiasi numero intero (ad esempio il risultato di un'operazione o di una funzione), e in particolare può avere anche valori negativi.

`lista[-1]` accede all'ultimo valore della lista

`lista[-2]` accede al penultimo valore della lista e così via

Infine possiamo concatenare due liste con l'operatore ‘+’

```
list1 = [1,2,3,4]
list2 = [5,6,7,8]
print(list1+list2)
>>> [1,2,3,4,5,6,7,8]
```

Esempio di utilizzo delle liste

Scriviamo un programma che chiede 10 numeri li inserisce in una lista, poi la stampa.

```
lista = []
i=0
while i<10:
    val = input("Inserisci un valore: ")
    lista = lista + [val]
    i=i+1
print("Hai inserito: ",lista)
```

Operazioni sulle liste

Lunghezza di una lista: `len(lista)`

Appartenenza di valore alla lista: `valore in lista` (ritorna True o False)

Ripetere una lista n volte: `lista*n`

Aggiungere un valore a una lista: `lista.append(valore)`

Rimuovere il valore in posizione `pos`: `lista.pop(pos)`

Rimuovere la prima occorrenza di valore: `lista.remove(valore)`

Conta quante volte valore compare in lista: `lista.count(valore)`

Ordina la lista; `lista.sort()`

Inverte la lista: `lista.reverse()`

Le stringhe e le liste

E' possibile trasformare una stringa di caratteri in una lista di caratteri, per farlo utilizziamo il comando di cast con il tipo voluto: `list` a questo punto possiamo trattarle come faremmo con una lista.

```
st = "Francesco"  
st = list(st)  
st.remove('e')  
print(st)
```

Sottoliste

Un'altra possibile operazione sulle liste consiste nell'indicizzarne alcune sottoparti.

```
list = [1,2,3,4,5,6]
```

```
list[1:3] ritorna [2,3] # NB: a:b -> a è incluso, b è escluso!
```

```
list[4:] ritorna [5,6]
```

```
list[:3] ritorna [1,2,3]
```

Funzioni con le liste

Come tutte le variabili, le liste possono essere usate come parametro di alcune funzioni. Per esempio cosa farà la seguente funzione?

```
def funzione (lista):  
    i=0  
    res = ""  
    while(i<len(lista)):  
        res = res+str(lista[i])  
        i=i+1  
    return res
```



Funzioni con le liste

Inoltre le funzioni possono restituire una lista come variabile. Cosa farà questa funzione?

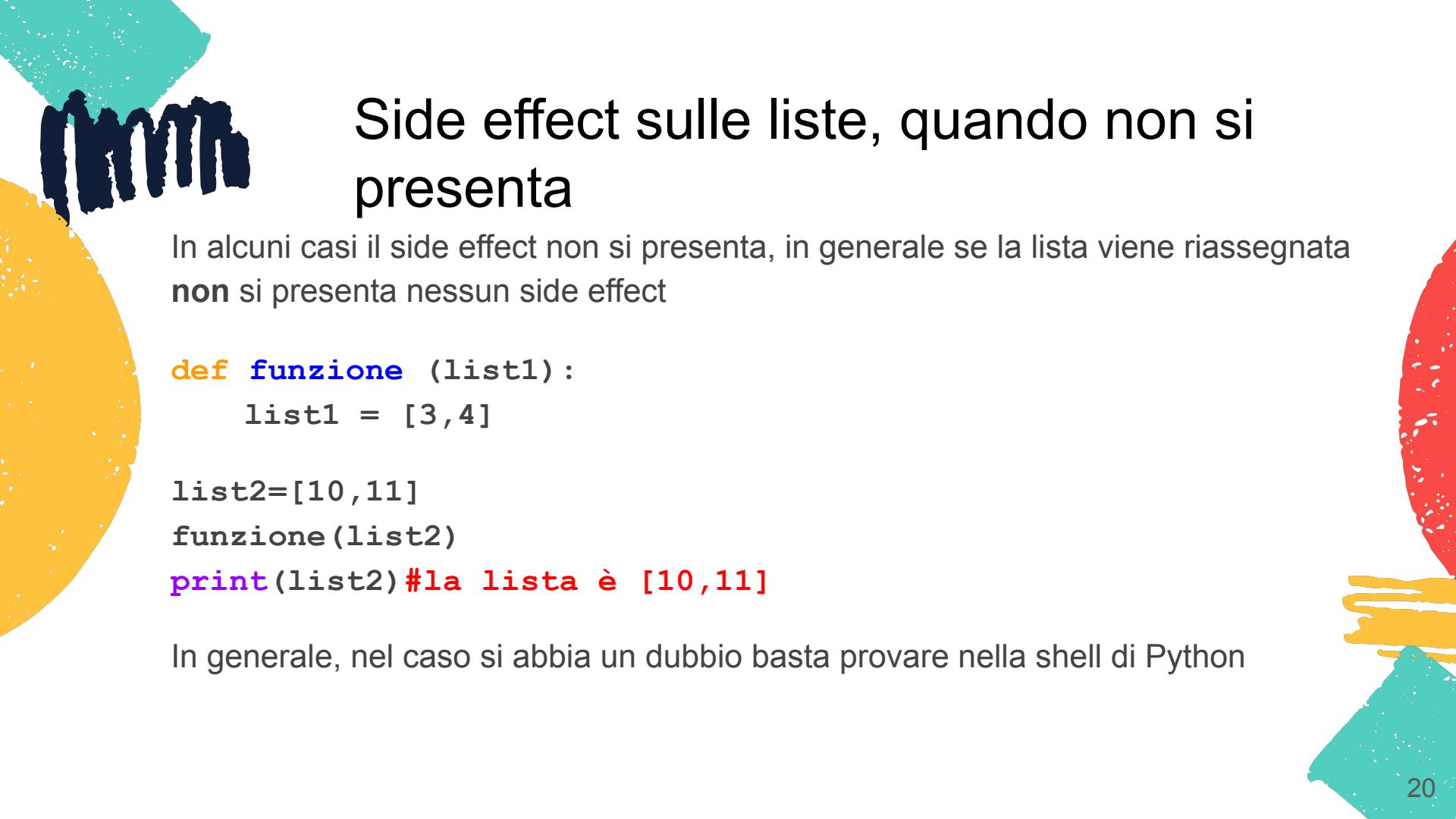
```
def funzione (lst1,lst2) :  
    res=[]  
    i=0  
    while ( i<len(lst1) and i<len(lst2) ) :  
        res.append([lst1[i],lst2[i]])  
        i=i+1  
    return res
```



Side effect sulle liste

Al contrario delle variabili nel caso delle liste si presenta il side effect, ovvero la lista passata come parametro viene modificata dalla funzione

```
def funzione (list1):  
    list1[0]= 3  
  
list2=[10,11]  
funzione(list2)  
print(list2) #la lista è [3,11]
```



Side effect sulle liste, quando non si presenta

In alcuni casi il side effect non si presenta, in generale se la lista viene riassegnata **non si presenta nessun side effect**

```
def funzione (list1):  
    list1 = [3,4]  
  
list2=[10,11]  
funzione(list2)  
print(list2) #la lista è [10,11]
```

In generale, nel caso si abbia un dubbio basta provare nella shell di Python

Dizionari

I dizionari sono delle liste che contengono coppie chiave:valore.

A differenza delle liste, non usiamo gli interi per accedere a un dizionario, bensì le chiavi.

Facciamo un esempio:

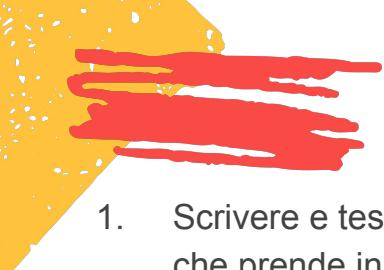
```
voti = {"paolo":9 , "antonio":10, "giorgio":10, "francesco": [3,6]}
```

è un dizionario che associa a ogni nome un voto. Per esempio

```
voti["paolo"]      ritorna 9
```

```
voti["francesco"] ritorna [3,6]
```

```
1 ages = {'Matt': 30, 'Katie': 29, 'Nik': 31, 'Jack': 43, 'Alison': 32, 'Kevin': 38}
2
3 # Get the max value in a Python dict
4 max_value = max(ages.values())
5 print(max_value)
6 # Returns: 43
7
8
9 # Get the key for a dict's max value
10 max_key = max(ages, key=ages.get)
11 print(max_key)
12 # Returns Jack
```



Esercizi

1. Scrivere e testare una funzione `sostituisci(lista, valore1, valore2)` che prende in input una lista e due valori e ritorna una lista in cui tutte le occorrenze del valore1 sono sostituite da valore2
Esempio :

```
print(sostituisci(['p','i','p','p','o'],'p','l'))
>>> ['l','i','l','l','o']
```
 2. Scrivere una funzione che data in input (alla funzione) una lista, ritorna una stringa formata dalla concatenazione degli elementi della lista.
 3. Scrivere una funzione che prende in input una lista, la controlla e se contiene almeno un valore che non sia un intero ritorna un errore. Altrimenti la funzione ritorna una lista che in prima posizione contiene il valore minimo dell'input, in seconda posizione la media e in terza posizione contiene il valore massimo.
- 



Altri esercizi

1. Data una lista di numeri restituire una lista in cui tutti gli elementi adiacenti uguali sono stati ridotti ad un singolo elemento.

Esempio:

con [1,2,2,3,2,2,3] la funzione ritorna [1,2,3,2,3]

Nota: si può sia restituire una nuova lista sia modificare quella passata come argomento.

2. Date due liste ordinate in ordine crescente, creare e restituire un'unica lista di tutti gli elementi ordinati.

E' possibile modificare le liste passate come argomenti.

Idealmente l'algoritmo dovrebbe implicare una singola iterazione su ciascuna lista.

Suggerimento: le liste sono ordinate: confrontare il primo elemento delle due liste ed aggiungere il più piccolo alla lista da restituire come risultato (rimuovendolo dalla lista originaria). Continuare fino a che una delle due liste non si svuota.





Esercizio del registro (molto difficile)

Si vuole creare un programma che riesca a gestire un registro che prevede coppie studente-voti e i voti sono una lista di interi.

Scrivere queste funzioni:

- chiedi_azione(): stampa 1 : aggiungi studente
 2 : aggiungi voto a studente
 3 : rimuovi studente
 4 : visualizza registro
 5 : esci
- aggiungi_studente(): aggiunge uno studente al registro senza voti
- aggiungi_voto(): chiede a quale studente aggiungere un voto e, se esiste, lo fa
- rimuovi_studente(): chiede quale studente eliminare dal registro, e se esiste lo fa
- visualizza_registro(): stampa tutti gli studenti e tutti i loro voti

Infine scrivere un corpo principale che cicli fino a quando non viene scelto di uscire e che chieda quale azione eseguire.



Registro - Continuazione

Aggiungere

- `media_studente()` che restituisce la media di uno studente dato in input
- `studente_piu_bravo()` che restituisce il nome dello studente con la media più alta



Esercizio del distributore di merendine

Si vuole creare un programma che riesca a gestire un distributore di merendine usando un dizionario.
Usiamo al posto degli Euro i centesimi.

Il distributore contiene:

merendine fiesta da 80 centesimi

merendine kinder bueno da 90 centesimi

succhi di frutta da 45 centesimi

merendine oreo da 90 centesimi

Scrivere una funzione che mi indichi il più costoso, ricordando che il massimo si può trovare attraverso i comandi:
`massima_chiave = max(dizionario, key=dizionario.get).`

Scrivere una funzione che stampi una lista di merendine che abbia sia il costo minore di 1 euro sia un numero di lettere pari nel nome.

Scrivere una funzione che stampi una lista di merendine che abbia il costo minore di 50 centesimi oppure un numero di lettere dispari.



Creazione di pozioni magiche

Il programma simula un'app di creazione di pozioni magiche. Ogni pozione ha un nome (la chiave), un costo e una lista di ingredienti necessari. Gli utenti possono visualizzare l'elenco delle pozioni disponibili, cercare una pozione per nome e creare una nuova pozione.

```
# Menu principale
while True:
    print("\n==== App Creazione Pozioni
Magiche ===")
    print("1. Mostra elenco delle pozioni")
    print("2. Cerca una pozione")
    print("3. Crea una nuova pozione")
    print("4. Esci")
```

