

# PYTHON BOOTCAMP

## Lezione 3



# LISTE



# LISTE

- ▶ Tramite le liste posso accorpare più entità in un'unica variabile

*Shell*

```
>>> lista = [1, 4, 5, 7, 234, 54]  
[1, 4, 5, 7, 234, 54]
```

- ▶ Posso accedere alle liste tramite l'**indice**, il numero della posizione di un oggetto all'interno della lista (il primo oggetto ha indice 0, il secondo 1 etc..)
- ▶ Le liste sono ordinate, vuol dire che gli oggetti all'interno hanno un ordine definito che non cambia

# LISTE

- Le liste possono essere di qualsiasi TIPO

```
lista_stringhe = ['Banane', 'Mele', 'Arance', 'Kiwi']
```

```
lista_bool = [True, False, True]
```

- Possono anche avere oggetti con TIPI diversi

```
lista_mista = ['carro', 243, True, 'pesche']
```

- Come definirle e come accedere agli elementi:

```
Shell  
>>> lista = ['Banane', 'Mele', 'Kiwi']  
>>> print(lista[1])  
'Mele'
```

Tramite [] e l'indice di riferimento posso accedere ai valori

# LISTE

- ▶ Se uso indici negativi accedo agli elementi in ordine inverso: se l'indice è **-1** accedo all'ultimo elemento, con **-2** al penultimo etc..
- ▶ Posso usare `len(lista)` per sapere il numero di elementi della lista
- ▶ Posso accedere a sottosezioni della lista

Shell

```
>>> lista = ['Banane', 'Mele', 'Kiwi', 'Pere', 'Meloni', 'Limoni', 'Fragole']
>>> print(lista[1:4])
['Mele', 'Kiwi', 'Pere']
>>> print(lista[4:])
['Meloni', 'Limoni', 'Fragole']
>>> print(lista[:4])
['Banane', 'Mele', 'Kiwi', 'Pere']
```



# LISTE

- Posso usare il for ... in ... per scorrere le liste

```
Shell
>>> lista = ['Banane', 'Mele', 'Kiwi']
>>> for i in lista:
        print(i)
'Banane'
'Mele'
'Kiwi'
```

- Posso inserire valori all'interno della lista o 'appenderli' alla coda della lista, li mette in fondo alla lista:

```
Shell
>>> lista.append('motorino')
['Banane', 'Mele', 'Kiwi', 'motorino']
>>> lista.insert(1, 'cavalli')
['Banane', 'cavalli', 'Mele', 'Kiwi', 'motorino']
```

# LISTE

- Per rimuovere un elemento dalla lista posso usare `lista.remove(element)`

```
Shell  
['Banane', 'Mele', 'Kiwi', 'motorino']  
>>> lista.remove('motorino')  
['Banane', 'cavalli', 'Mele', 'Kiwi']
```

- Se ho due elementi uguali elimino solo la prima occorrenza

```
Shell  
['motorino', 'Banane', 'Mele', 'Kiwi', 'motorino']  
>>> lista.remove('motorino')  
['Banane', 'cavalli', 'Mele', 'Kiwi', 'motorino']
```

# TURTLE





# TURLE

- Posso usare i comandi di **turtle** per modificare lo Screen

*Script*

```
import turtle  
  
sc = turtle.Screen()  
sc.title('Il mio schermo!')  
sc.bgcolor('black')  
sc.setup(width=1000, height=600)
```

- Posso chiudere lo screen con **turtle.bye()**

# TURTLE

- Usare gli input da tastiera

Script

```
import turtle  
  
sc = turtle.Screen()  
sc.setup(width=1000, height=600)  
  
sc.listen()  
sc.onkeypress(mia_funzione, 'e')
```

Script

```
def mia_funzione():  
    sc.bye()
```

# TURTLE

- Posso modificare la mia **turtle**
- Posso creare più **turtle**

Script

```
import turtle

square = turtle.Turtle()
square.shape('square')
square.color('red')

circle = turtle.Turtle()
circle.shape('circle')
circle.color('blue')
```

- Posso usare sapere la posizione di ogni **turtle** tramite **nome\_turtle.xcor()** e **nome\_turtle.ycor()** per le coordinate x e y rispettivamente. Se uso le funzioni **nome\_turtle.setx()** e **nome\_turtle.sety()** posso settare nuove coordinate alle turtle
- Posso anche modificare la velocità della turtle con **nome\_turtle.speed()**...

# TURTLE

- Refresh dello schermo `screen.update()` dentro un `while` se voglio che il codice aggiorni lo screen in continuazione. Esempio se voglio muovere una `turtle` a una certa velocità (tramite `nome_turtle.dx` e `nome_turtle.dy`)

Script

```
Import turtle

sc = turtle.Screen()

my_turtle = turtle.Turtle()
my_turtle.speed(5)
my_turtle.shape("circle")
my_turtle.color("blue")
my_turtle.penup()
my_turtle.dx = 1

while True:
    sc.update()

    my_turtle.setx(my_turtle.xcor() + my_turtle.dx)
```

# TURTLE

- Posso anche comandare alla mia turtle di cambiare direzione con la tastiera

Script

```
import turtle

Myturtle ...

def giraSinistra():
    my_turtle.dx = -1

def giraDestra():
    my_turtle.dx = 1

sc.listen()
sc.onkeypress(giraSinistra, 'Left')
sc.onkeypress(giraDestra, 'Right')
```

# Esercizi

- ▶ Costruire una lista di 50 numeri da 1 a 1000 e stampare a video qual è il valore più grande
- ▶ Scrivere una funzione che rimuova i duplicati all'interno di una lista.
- ▶ Scrivere una funzione che prenda in input una lista e restituisce una lista con gli stessi elementi ma ordinati al contrario rispetto all'input
- ▶ Costruire una lista con 30 numeri randomici (da 0 a 100) e ordinarla mettendo all'inizio il valore più piccolo e poi a salire fino al valore più grande che sarà all'ultimo posto
- ▶ Fare una funzione python che trovi la sequenza più lunga di numeri consecutivi all'interno di una lista

# Esercizi

- ▶ Disegnare una spirale con turle 'arcobaleno' a intervalli regolari deve cambiare colore, per un set di 7 colori, e poi ricominciare
- ▶ Muovere una pallina ad una certa velocità. ma non devo farla uscire dallo schermo!
- ▶ Muovere una *turtle* **player** con le freccette della tastiera. Creare poi una *turtle* **target** in punto randomico, una volta che ci vado sopra con il **player**, il target scompare e riappare in un altro punto dello schermo. Continua così finché non arrivo a 10 **target** catturati. Usare colori diversi per **player** e **target**  
extra : aumentare la velocità del player ogni volta che viene preso un target



# Esercizi

- ▶ Riprodurre questa figura in turtle  
*cosa fa il terzo argomento di circle() ?*
- ▶ Creare il gioco **snake** (che nel nostro caso di chiamerà Pitone). Usare le freccette per muovere la turtle (che sarà il nostro **snake**). Creare i target (turtle statiche) in un posto randomico dentro lo schermo. Una volta che lo **snake** mangia un target, in automatico compare un altro target ne compare un altro e lo **snake** aumenta di velocità. Se lo **snake** esce dallo schermo faccio rinascere lo **snake** dal centro con la velocità di partenza.

