THE PYTHON BASICS

COS'È PYTHON?

Python è un linguaggio di programmazione che permette di creare applicazioni e automatizzare i processi in modo facile e veloce.

Python è un lunguaggio interpretato e non compilato!

Si distingue dagli altri per la sua estrema semplicità nella sintassi e nella gestione della memoria!

ANCORA UN PAIO DI INFO UTILI

È stato sviluppato da Guido Van Rossum nel 1991.

È un linguaggio di «alto livello» ovvero molto automatizzato.

È un linguaggio open source.

LA FUNZIONE ‘print’

La prima funzione che andremo ad analizzare è la funzione ‘print’!

```
print('Hello World')
```

Mostra sulla riga di comando quello sta scritto tra parentesi

I TIPI E LE VARIABILI

In python è possibile rappresentare diversi tipi di valori, quelli basici sono:
numeri interi (`int`), numeri decimali (`float`),
stringhe (`str`) e valori booleani come vero o falso (`bool`).

5 => `int`; 4,66 => `float`; 'ciao' => `str`; `True` => `bool`

Questi valori possono essere contenuti in delle variabili che ne assumono il tipo e il valore!

```
x = 5
print(x)

word = 'Hello World'
print(word)
```

```
y = 7
print(y)

ciao = 'Hello World'
print(ciao)
```

Per creare una variabile vuota si può assegnare come valore `None`: `x = None`

LE OPERAZIONI CON I TIPI

Con questi valori si possono svolgere diverse operazioni:

Con i numeri (`int`, `float`) si possono svolgere le 4 operazioni (+, -, *, /)

```
x = 5
print(x + 5)
print(x - 2)
```

```
print(x * 2)
print(x / 5)
print(x ** 2)
```

Con le stringhe (`str`) si possono svolgere 2 operazioni (+, *)

```
x = 'ba'
print(x + 'na'*2)
```

TABELLA OPERATORI

NUMERICI:

Operatore	Esempio	Significato
+	a + b	Somma
-	a – b	Sottrazione
*	a * b	Moltiplicazione
/	a / b	Divisione
**	a ** b	Elevamento a potenza
//	a // b	Parte intera della divisione
%	a % b	Resto della divisione

COMPARAZIONE:

Operatore	Esempio	Significato
==	a == b	È uguale a
!=	a != b	È diverso da
<	a < b	È minore di
<=	a <= b	È minore o uguale a
>	a > b	È maggiore di
>=	a >= b	È maggiore o uguale a

LOGICI:

Operatore	Esempio	Significato
not	not p	Non vero, quindi falso
and	p and q	Entrambi veri
or	p or q	Almeno uno dei due è vero

CONDIZIONI

Le condizioni si basano sul valori booleani (`True` or `False`) e valutano se una determinata espressione è vera o falsa.

```
x = 3
if x + 2 == 5:
    print('x + 2 è uguale a 5')
```

In questo caso la condizione `x + 2 == 5` è vera, quindi vengono le eseguite le operazioni che hai definito

Se fosse stato diverso da 3 non sarebbe stata eseguita alcuna azione...

`else` serve ad eseguire delle azioni in caso la condizione non fosse vera.

Nell'esempio infatti la condizione `x + 2 == 5` non è vera e quindi vengono eseguite le azioni definite dopo `else`.

```
x = 5
if x + 2 == 5:
    print('x + 2 è uguale a 5')
else:
    print('x + 2 è diverso da 5')
```

LE FUNZIONI (function)

Una funzione è un'insieme di operazioni definite dal programmatore che accettano dei parametri che devi fornire alla chiamata.

In python le funzioni si definiscono con la parola chiave 'def':

```
def saluta():
    print('ciao!')
```

Le funzioni possono anche accettare dei parametri per operazioni più complesse:

```
def somma(n1, n2):
    return n1 + n2
```

In questo caso sono n1 e n2 parametri della funzione 'somma' che ne fa una addizzione e ne ritorna il valore (return)

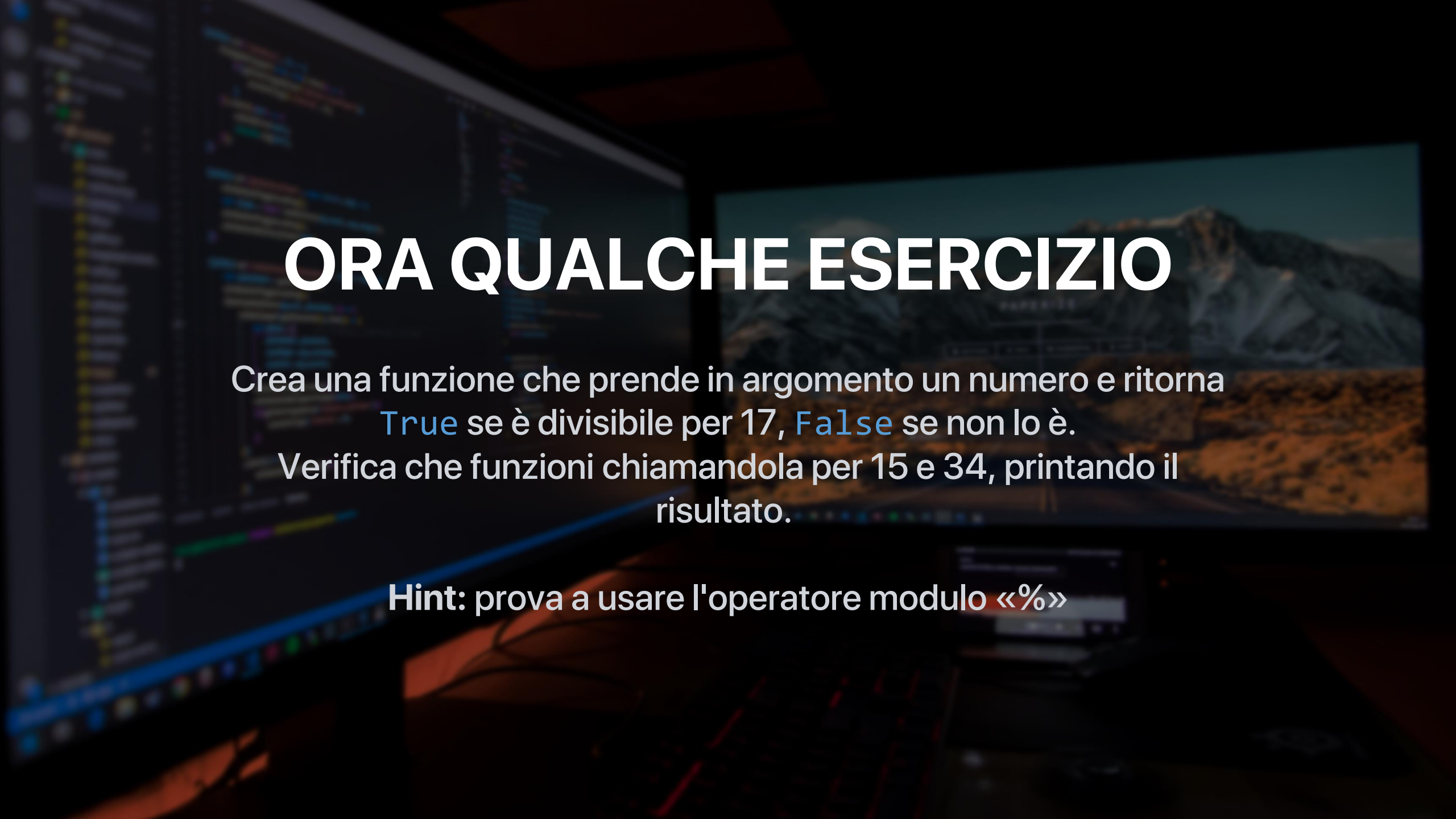
USARE LE FUNZIONI

Per utilizzare le funzioni definite in precedenza dobbiamo effettuare una chiamata

```
# chiamiamo la funzione saluta
saluta() # ora dovremmo vedere
# che la funzione è stata eseguita!
```

Per chiamare la funzione somma dobbiamo specificare anche i due parametri richiesti ovvero n1 e n2

```
# chiamo la funzione somma con parametri 5 e 3
s = somma(5, 3) # memorizzo l'output
# in una variabile
print(s) # s dovrebbe valere 8!
```



ORA QUALCHE ESERCIZIO

Crea una funzione che prende in argomento un numero e ritorna `True` se è divisibile per 17, `False` se non lo è.
Verifica che funzioni chiamandola per 15 e 34, printando il risultato.

Hint: prova a usare l'operatore modulo «%»

UN PAIO DI FUNZIONI UTILI

```
input('testo da visualizzare') # richiede un'input dalla console

min([1,2,3,4,5]) # ritorna il più piccolo elemento nella lista (1)

max([1,2,3,4,5]) # ritorna il più grande elemento nella lista (5)

sum([1,2,3,4,5]) # ritorna la somma di tutti gli elementi nella lista

len([1,2,3,4,5]) # conta quanti elementi contiene la lista (5)

int('5') # converte l'argomento in numero intero

str(5) # converte l'argomento in stringa di caratteri

round(2.55) # arrotonda il numero alla cifra dell'unità (3)

type(5) # ritorna il tipo dell'argomento passato (int)
```

SOLUZIONE:

```python
def divisibile_per_17(n) -> bool:
    # controlla che il resto della divisione
    # per 17 sia 0
    if n % 17 == 0:
        # se è 0, allora il numero è divisibile
        # per 17, ritorna True
        return True
    else:
        # altrimenti non lo è, ritorna False
        return False

print(divisibile_per_17(15)) # False
print(divisibile_per_17(34)) # True
```

LE LISTE (`list`)

Una lista è un insieme ordinato di elementi.
In python le liste si definiscono con le parentesi quadre: `l = [1,2,3]`

Si possono scegliere elemanti da una lista attaraverso l'indice dell'elemento nella lista:

```
# elemento nella lista al posto 0
print(l[0]) # cioè il primo elemento
```

Per aggiungere un'elemento ad una lista si può utilizzare la funzione '`append`':

```
l.append(5)
print(l)
```

Invece per rimuovere un'elemento si può usare la funzione '`remove`':

```
l.remove(5)
print(l)
```

IL CICLO `while`

Un ciclo è una ripetizione di una serie determinate operazioni.

In python esistono 2 tipi di cicli (loop): il ciclo `for` e il ciclo `while`

Il ciclo `while` si definisce con:

```
while expr:
    do_something
```

Se expr è falso allora il ciclo si interrompe sennò il ciclo continua.

Alcuni esempi:

```
x = 0
while x != 5:
    x = x + 1
    print(x)
```

```
word = input('Inserisci una parola: ')
# finchè la parola non è 'stop'
while word != 'stop': # continua a chiedere
    print('Hai scritto: ' + word)
    word = input('Inserisci una parola: ')
```

IL CICLO `for`

Un ciclo for ripete la stessa azione (itera) tra una determinata serie di elementi come le liste.

Il ciclo `for` è definito con:

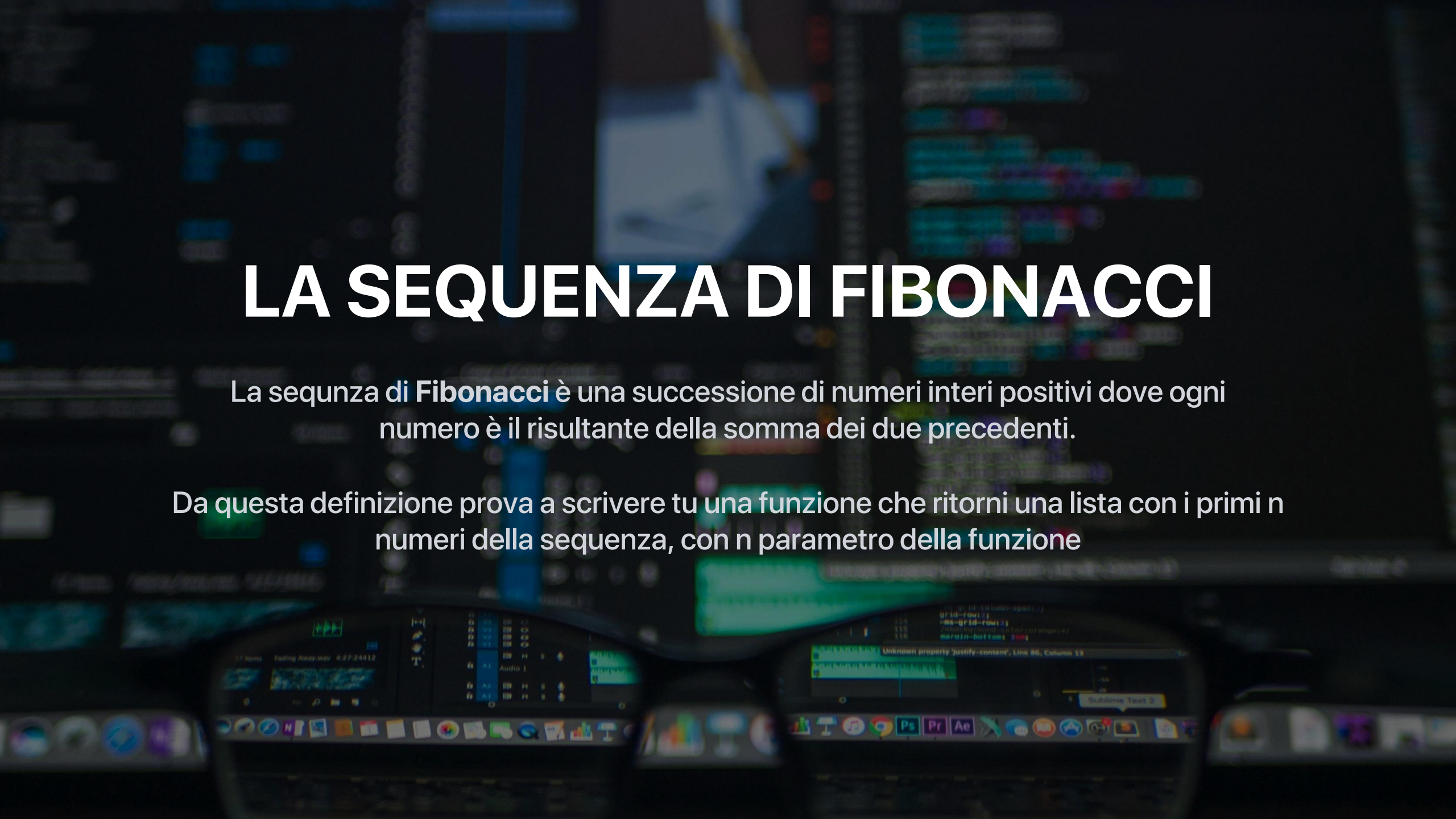
```
for i in x:
```

Dove `x` è la lista nella quale iterare e `i` è la variabile nella quale viene memorizzato l'elemento di iterazione.

Alcuni Esempi:

```
for i in [1,2,3,4,5]:
    print(i)
```

```
# range() itera per n volte,
# i assume il valore del numero
# della ripetizione partendo da 0
for i in range(5):
    print(i)
```



LA SEQUENZA DI FIBONACCI

La sequnza di **Fibonacci** è una successione di numeri interi positivi dove ogni numero è il risultante della somma dei due precedenti.

Da questa definizione prova a scrivere tu una funzione che ritorni una lista con i primi n numeri della sequenza, con n parametro della funzione

LA SEQUENZA DI FIBONACCI

Da questa definizione prova a scrivere tu una funzione che ritorni una lista con i primi n numeri della sequenza, con n parametro della funzione

Procedimento:

- Definisci una funzione che prende come argomento n, cioè quanti numeri di Fibonacci verranno calcolati
- Inizializza una lista contente 0, 1
- Avvia un ciclo for che itera la variabile i nei numeri da uno a n. Questo aggiungerà alla lista l'i-esimo numero di Fibonacci, dopo aver verificato che i sia maggiore di 1
- ritorna la lista
- chiama la funzione con parametro 20 per verificare che funzioni

SOLUZIONE:

```python
def fibonacci(n):
    # crea una lista contenete i primi 2 numeri della sequenza
    numeri = [0,1]

    # ripeti n volte
    for i in range(n):
        # se l'indice di ripetizione è maggiore di 1
        if i > 1:
            # il numero successivo è la somma dei due precedenti
            numeri.append(numeri[i-2] + numeri[i-1])

    # ritorna la lista di numeri
    return numeri

print(fibonacci(10)) # [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```


DO SOMETHING GREAT