

Agenda

- Introduzione al Coding e ai Linguaggi di Programmazione
- Programmazione Visuale (Scratch)
- Transizione al Coding Testuale: Python
- Attività Pratica in Python

Obiettivi

- Introdurre il concetto di coding e il suo ruolo nell'educazione.
- Familiarizzare con un linguaggio di programmazione visuale (Scratch) per poi iniziare a passare al linguaggio testuale (Python).
- Avviare un'attività pratica che mostri i costrutti fondamentali della programmazione: sequenza, condizione e iterazione.

Definizione di Coding

Il coding è il processo di scrittura di istruzioni comprensibili per un computer affinché esegua determinate azioni. Spesso viene associato alla programmazione, ma il suo significato va oltre il semplice scrivere codice. Il coding, infatti, sviluppa una serie di abilità cognitive che sono fondamentali anche nella vita quotidiana.

Nel contesto dell'educazione moderna, il coding assume un ruolo chiave in quanto:

- **Promuove il pensiero computazionale**, ossia la capacità di scomporre un problema complesso in parti più gestibili (decomposizione), individuare schemi (pattern recognition), astrarre dettagli irrilevanti e creare soluzioni algoritmiche.
- **Stimola il problem solving**, ovvero l'abilità di affrontare problemi e cercare soluzioni strutturate, valutando diverse opzioni e sperimentando.
- **Fornisce competenze logiche**: il coding insegna a pensare in modo sequenziale e logico, poiché i computer seguono istruzioni lineari, per cui ogni passaggio deve essere accuratamente strutturato per produrre il risultato corretto.
- **Favorisce la creatività e l'innovazione**: attraverso il coding, gli studenti imparano a progettare e costruire cose nuove, come app, giochi o simulazioni, e possono trasformare idee astratte in realizzazioni concrete.

Inoltre, il coding non è confinato alle discipline tecnico-scientifiche, ma può essere applicato in molte altre materie, come le scienze umane, l'arte o la matematica, rendendo il processo di apprendimento più coinvolgente e interattivo per gli studenti.

Panoramica sui Linguaggi Visuali e Testuali

- I linguaggi di programmazione visuali, come **Scratch**, utilizzano un'interfaccia grafica a blocchi che permette di creare programmi semplicemente trascinando e collegando blocchi colorati.
- I linguaggi di programmazione testuali, come **Python**, rappresentano un livello successivo di complessità, ma offrono anche maggiore potenza e flessibilità.



Vantaggi a confronto

Visuali

Ideale per introdurre il coding, soprattutto per principianti o giovani studenti, per diversi motivi.

- Semplicità d'uso
- Visualizzazione immediata
- Approccio creativo

Testuali

Richiedono la scrittura di istruzioni utilizzando una sintassi specifica, con regole precise per l'uso di parole chiave, variabili e strutture.

- Maggiore flessibilità e controllo
- Applicazioni reali
- Modularità

Transizione dal Visuale al Testuale

- **Logica comune:** I concetti di base (sequenze, condizioni, cicli) che si imparano nei linguaggi visuali sono gli stessi che si trovano nei linguaggi testuali. Ad esempio, un ciclo che si ripete in Scratch può essere tradotto facilmente in un ciclo *for* o *while* in Python.
- **Crescente attenzione alla sintassi:** Mentre Scratch "perdona" eventuali errori di blocchi, in Python (e in altri linguaggi testuali) è necessario rispettare rigorosamente le regole sintattiche. Questo rappresenta una sfida iniziale, ma permette agli studenti di comprendere il valore della precisione nella programmazione.
- **Maggiore controllo:** Nei linguaggi testuali, gli studenti hanno più controllo sui dettagli del programma, il che permette loro di creare applicazioni più complesse e personalizzate, spingendoli a ragionare in termini più avanzati.

In sintesi, i **linguaggi visuali** offrono un primo approccio semplice e **intuitivo** al coding, mentre i **linguaggi testuali**, come Python, permettono di compiere un passo successivo verso una programmazione più **potente** e applicabile a problemi complessi.

Esperienze con il Coding

QUALI SONO LE VOSTRE ESPERIENZE PRECEDENTI CON IL CODING?

- Hai già insegnato coding nella tua materia?
- In che modo?
- Hai utilizzato strumenti come Scratch, Blockly, Python o altri linguaggi?
- Hai seguito corsi di coding o partecipato a workshop?
- Se non hai esperienza, quali sono le tue percezioni sul coding?

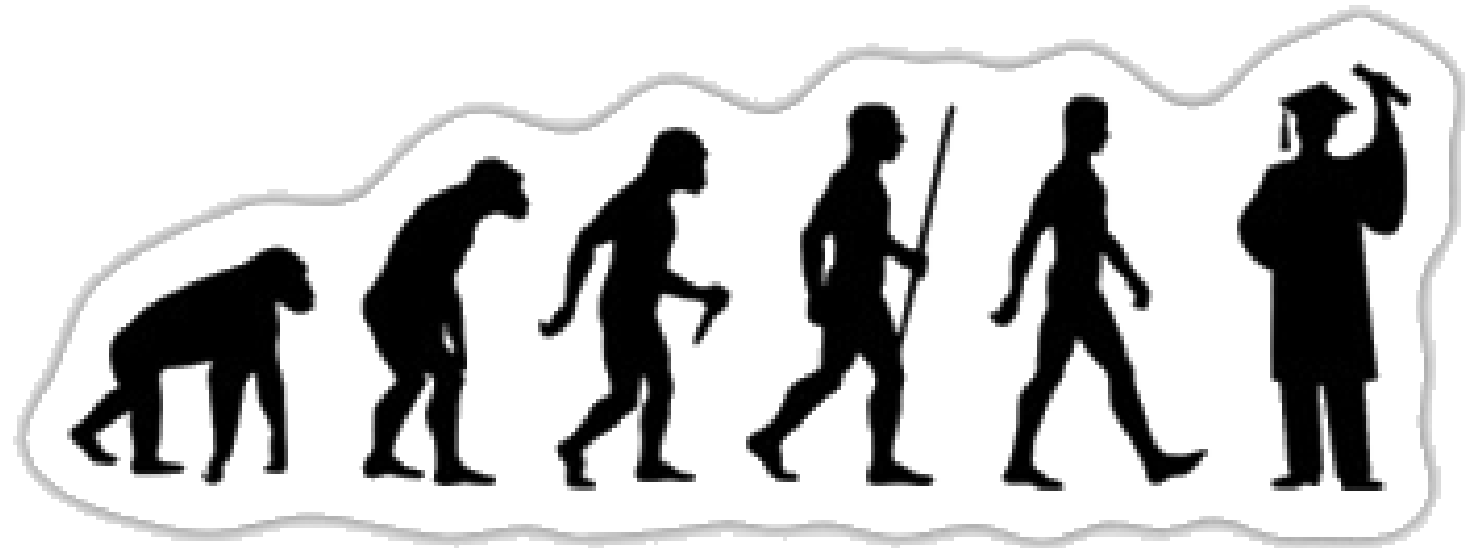
Cosa Vorresti Ottenere da Questo Corso?

QUALI SONO LE TUE ASPETTATIVE E OBIETTIVI RIGUARDO AL CODING?

- Vuoi acquisire competenze tecniche specifiche?
- Vorresti integrare il coding nella tua materia?
- In che modo?
- Come pensi che il coding possa influenzare il modo in cui i tuoi studenti apprendono?
- Quali competenze pensi di dover sviluppare per sentirti a tuo agio nell'insegnare coding?

Sintesi delle Vostre Esperienze e Obiettivi

LE VOSTRE ESPERIENZE E ASPETTATIVE VI AIUTERANNO A COSTRUIRE LE BASI
PER UN CODING EFFICACE IN CLASSE!

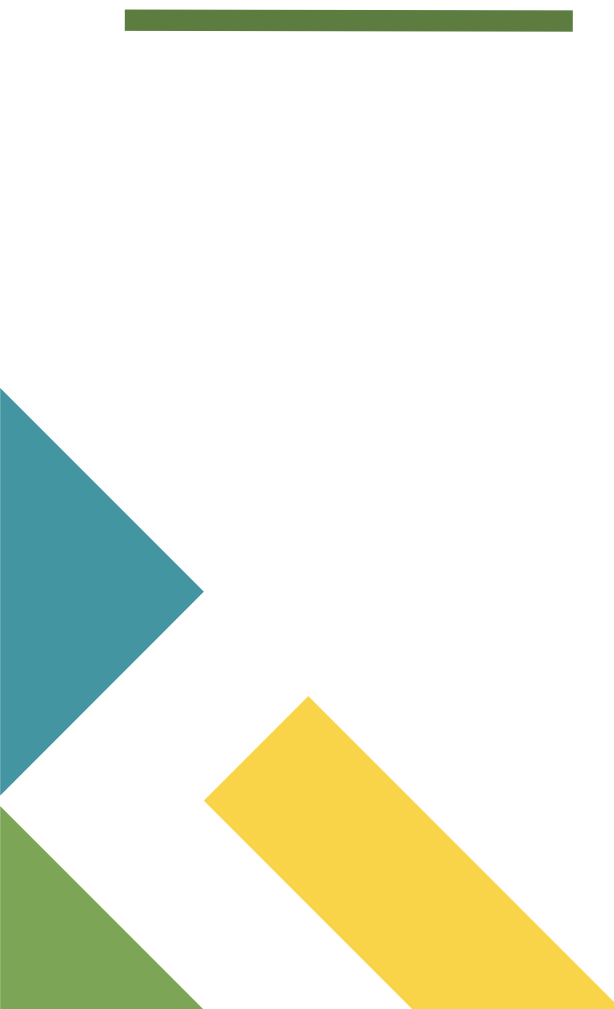


Scratch

- Software di **programmazione visuale** (usa blocchi di programmazione) sviluppato dal **MIT Media Lab**.
- Software **gratuito**.
- Fascia d'età: **8-16 anni**.

The Scratch logo, featuring the word "SCRATCH" in a stylized, orange, blocky font with a blue outline, set against a white background with a blue border.

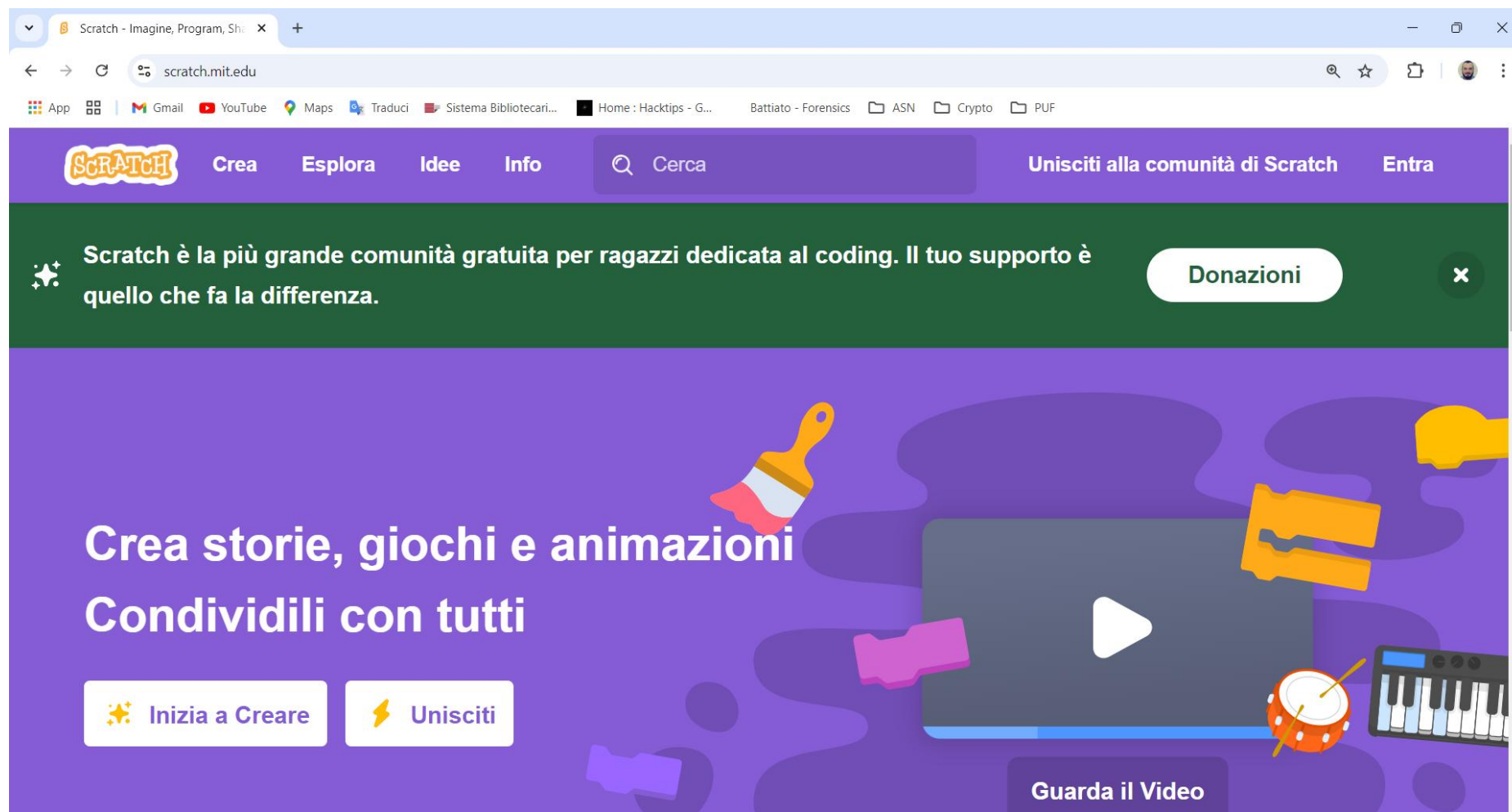
Introduzione a Scratch



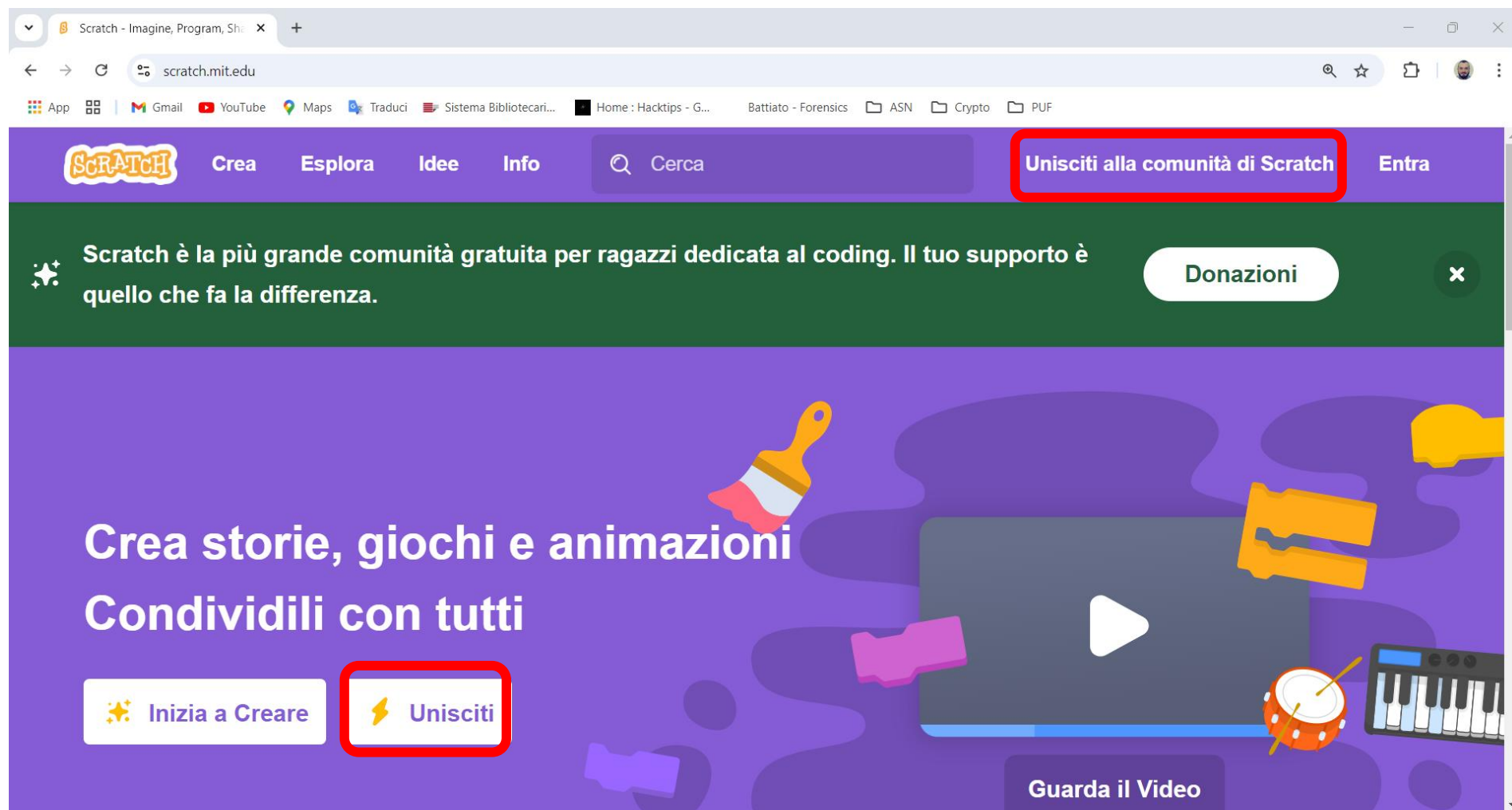
- **Cos'è?** Scratch è un linguaggio di programmazione visuale sviluppato dal MIT Media Lab. È progettato specificamente muovere i primi passi nel mondo del coding.
- **Come funziona?** Gli utenti programmano usando blocchi colorati che rappresentano comandi o azioni. Questi blocchi possono essere assemblati insieme per creare animazioni, storie interattive, giochi e altro ancora.
- **Perché è utile nell'educazione?** Scratch aiuta gli studenti a comprendere concetti di base della programmazione, come sequenze, cicli e condizioni, senza dover affrontare la complessità della sintassi testuale. Promuove anche la creatività e il pensiero computazionale attraverso progetti visivi.

Esempio di utilizzo: Creare un gioco interattivo in cui un personaggio si muove sullo schermo in base agli input dell'utente (es. frecce della tastiera)

https://scratch.mit.edu/



https://scratch.mit.edu/



https://scratch.mit.edu/

Scratch - Join Scratch

scratch.mit.edu/join

App | Gmail | YouTube | Maps | Traduci | Sistema Bibliotecari... | Home : Hacktips - G... | Battiato - Forensics | ASN | Crypto | PUF

100% Reimposta

Unisciti alla comunità di Scratch

Crea progetti, condividi le tue idee, fai degli amici.
E' gratuito!

Crea uno username

Nome utente

Non usare il tuo nome

Crea una password

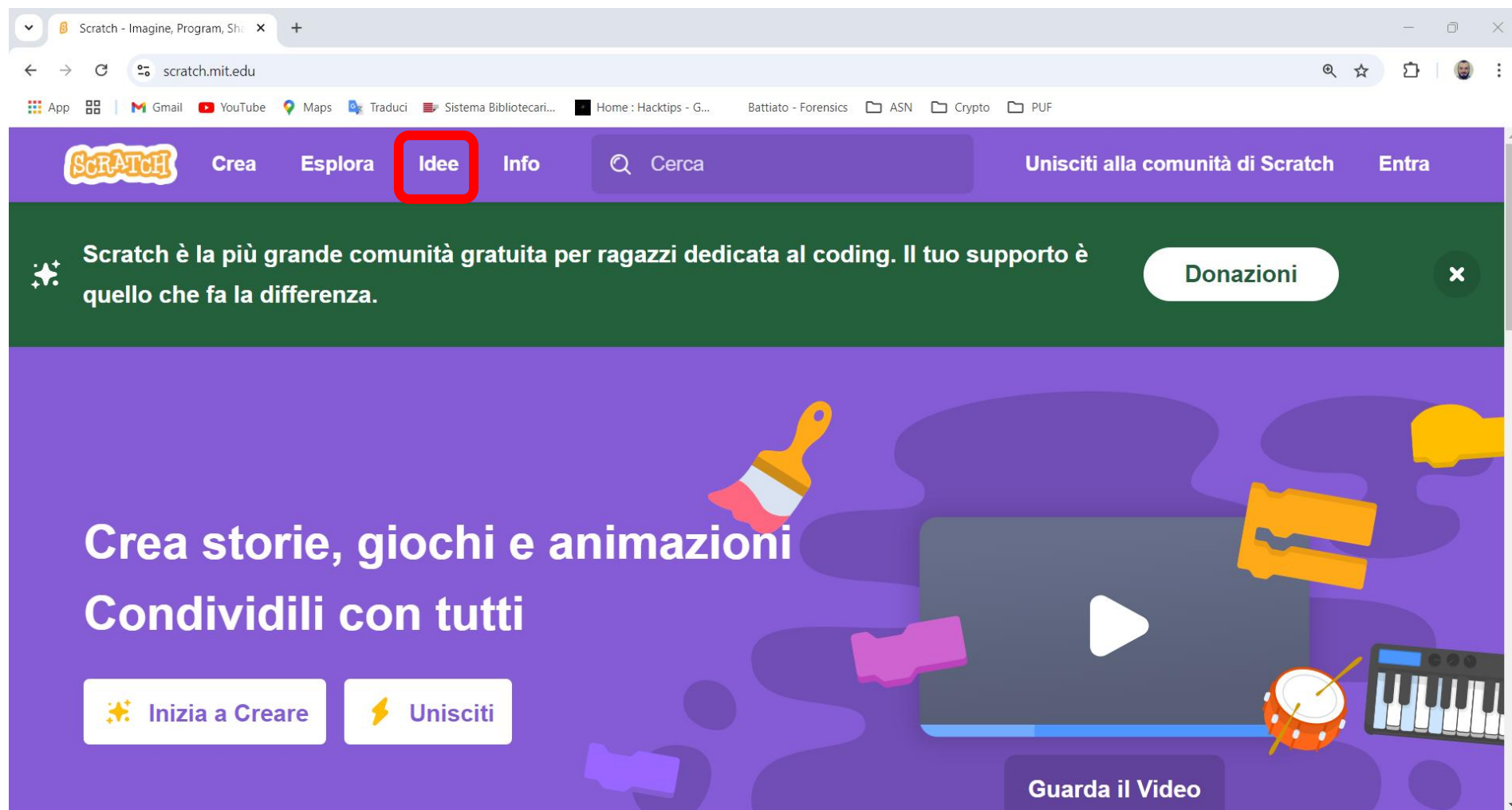
Password

Digita di nuovo la password

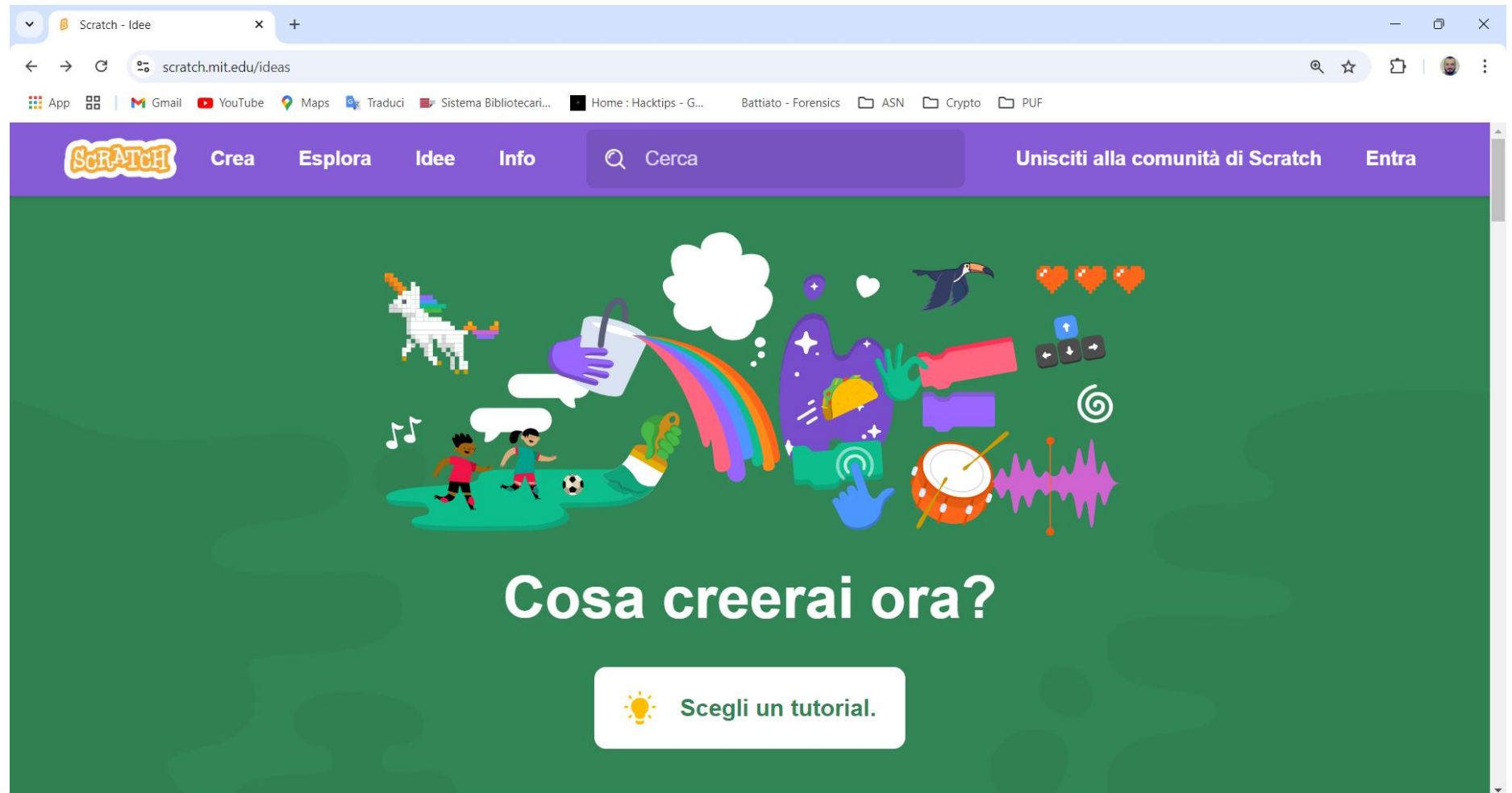
☒ Mostra la password

Avanti

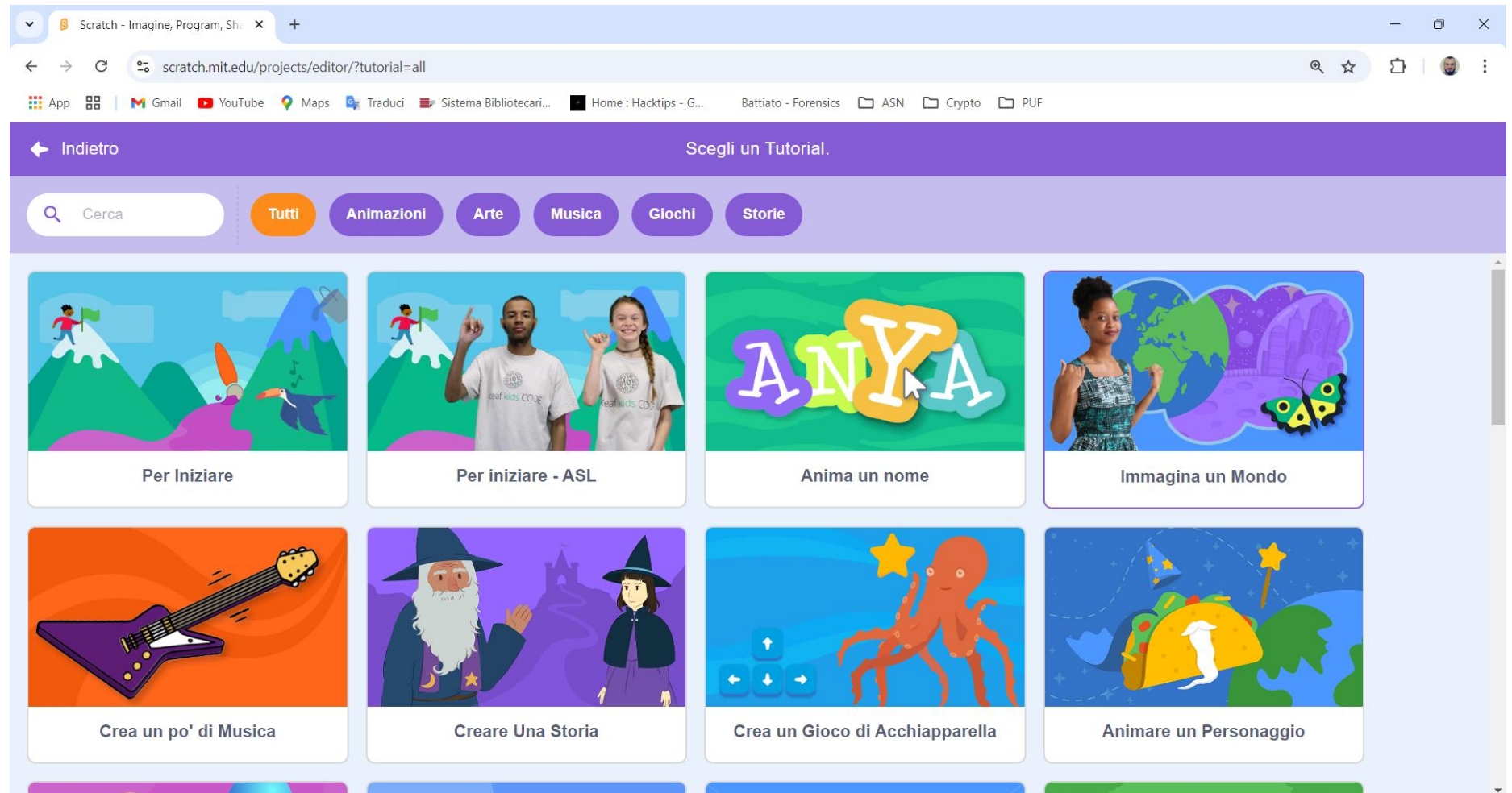
Un mondo di idee...



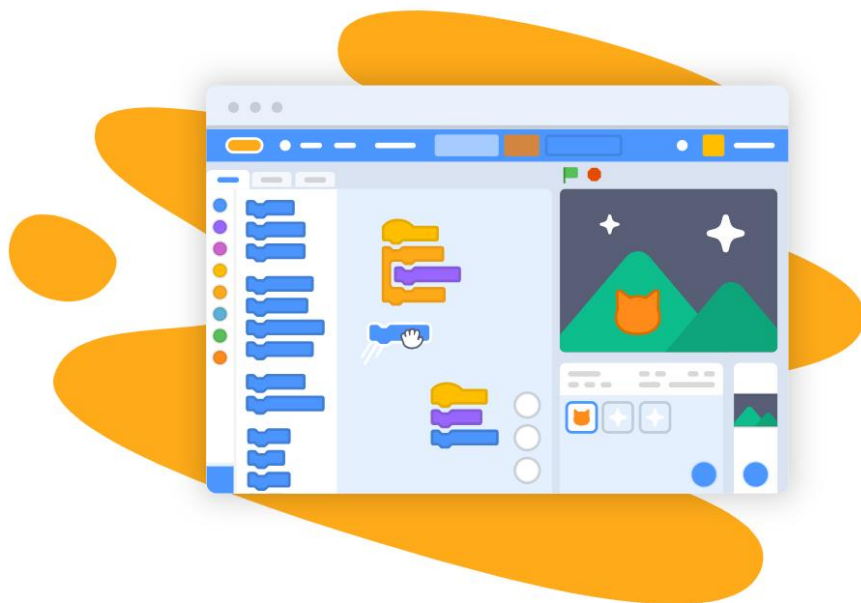
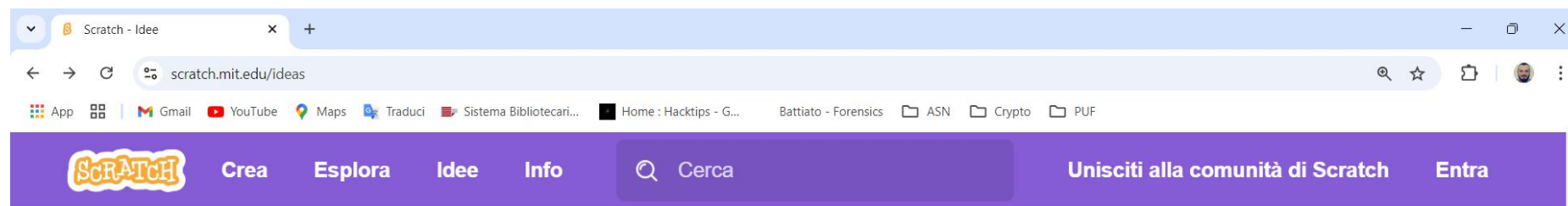
Tutorial



...per tutti i gusti!



Progetti di esempio

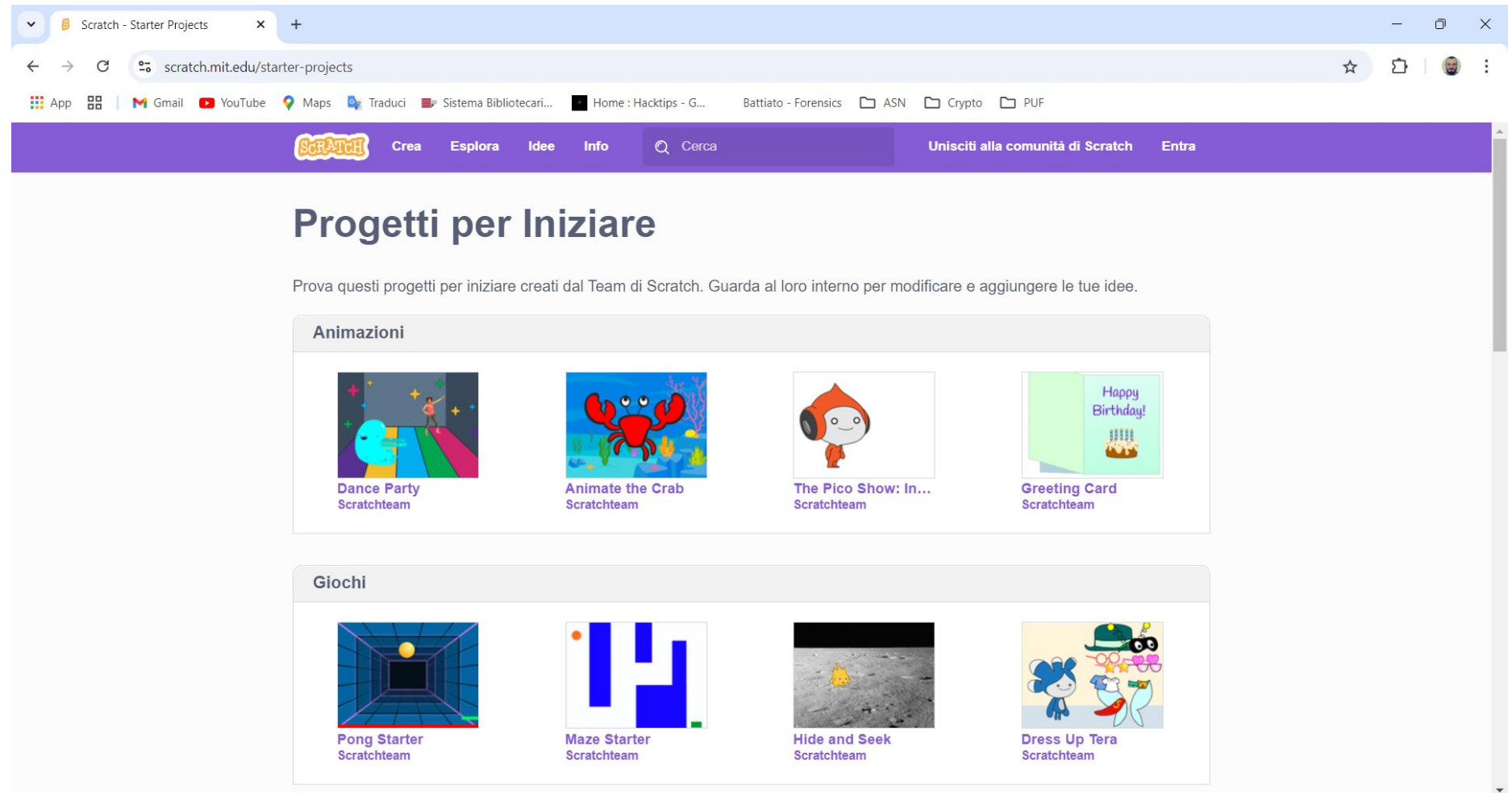


Progetti per Iniziare

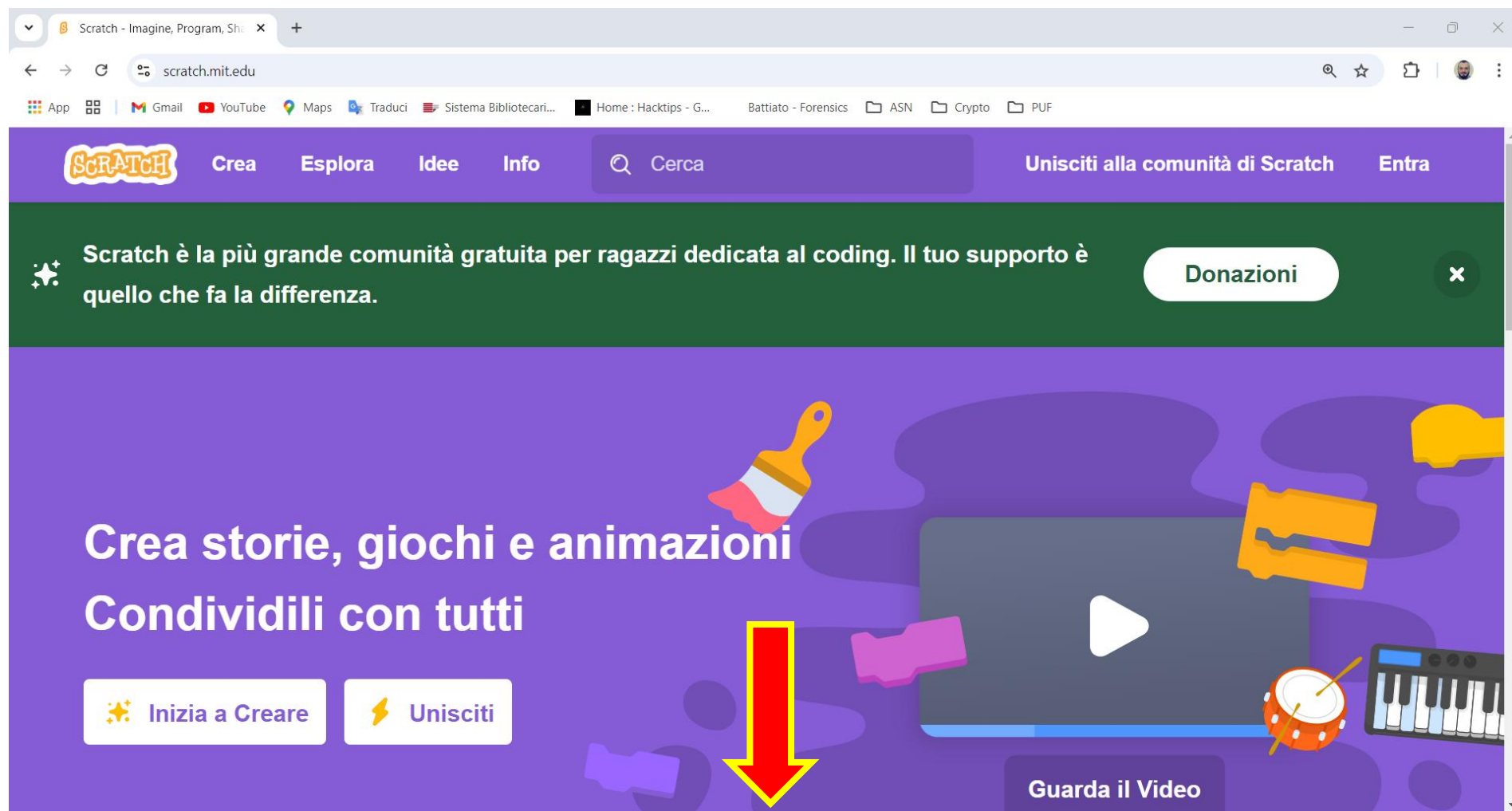
Puoi giocare con i Progetti per Iniziare e puoi remixarli per creare i tuoi progetti personali.

[Esplora i Progetti Per Iniziare](#)

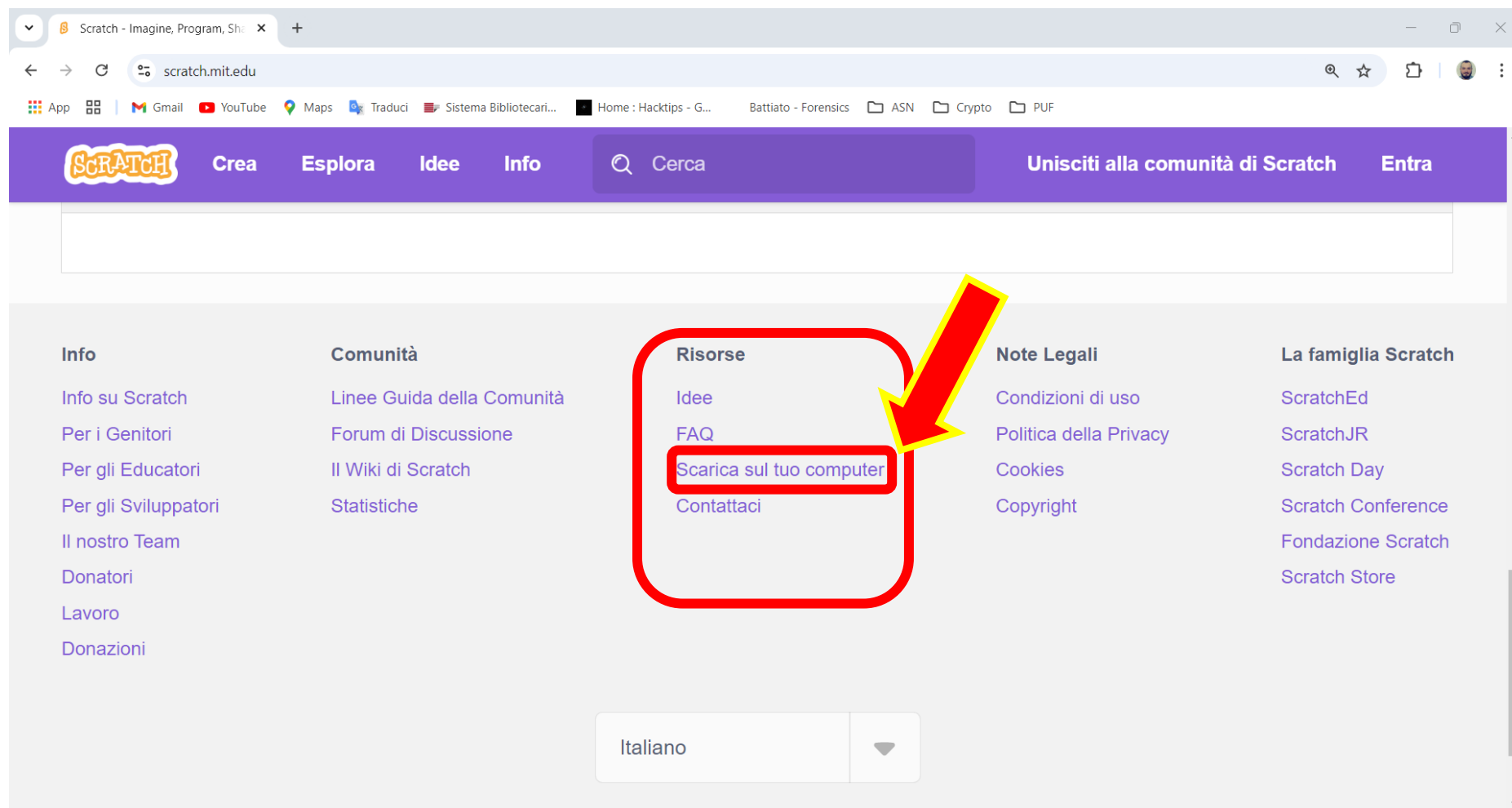
...per sperimentare e trarre ispirazione!



Anche disconnessi!



...si possono creare tanti progetti!



Requisiti fondamentali



The screenshot shows a web browser window with the address bar displaying 'scratch.mit.edu/download'. The browser's tab is labeled 'Scratch - Scratch Offline Editor'. The website's header is purple and contains the Scratch logo, navigation links ('Crea', 'Esplora', 'Idee', 'Info'), a search bar with the text 'Cerca', and links to 'Unisciti alla comunità di Scratch' and 'Entra'. The main content area has a purple background and features the heading 'Scarica l'App di Scratch' next to the Scratch logo. Below this, a paragraph states: 'Vuoi creare e salvare i progetti Scratch anche senza una connessione internet? Scarica l'app gratuita di Scratch.' Under the heading 'Requisiti', there are four operating system options: 'Windows 10+', 'macOS 10.13+', 'ChromeOS', and 'Android 6.0+'. To the right of the text, there is an illustration of a yellow cardboard box with a green arrow pointing down into it, set against a background of stylized clouds. A large, thick red arrow with a yellow outline points downwards from the bottom right of the text area towards the box illustration.

Scratch - Scratch Offline Editor

scratch.mit.edu/download

App Gmail YouTube Maps Traduci Sistema Bibliotecari... Home : Hacktips - G... Battiato - Forensics ASN Crypto PUF

Scratch Crea Esplora Idee Info Cerca Unisciti alla comunità di Scratch Entra

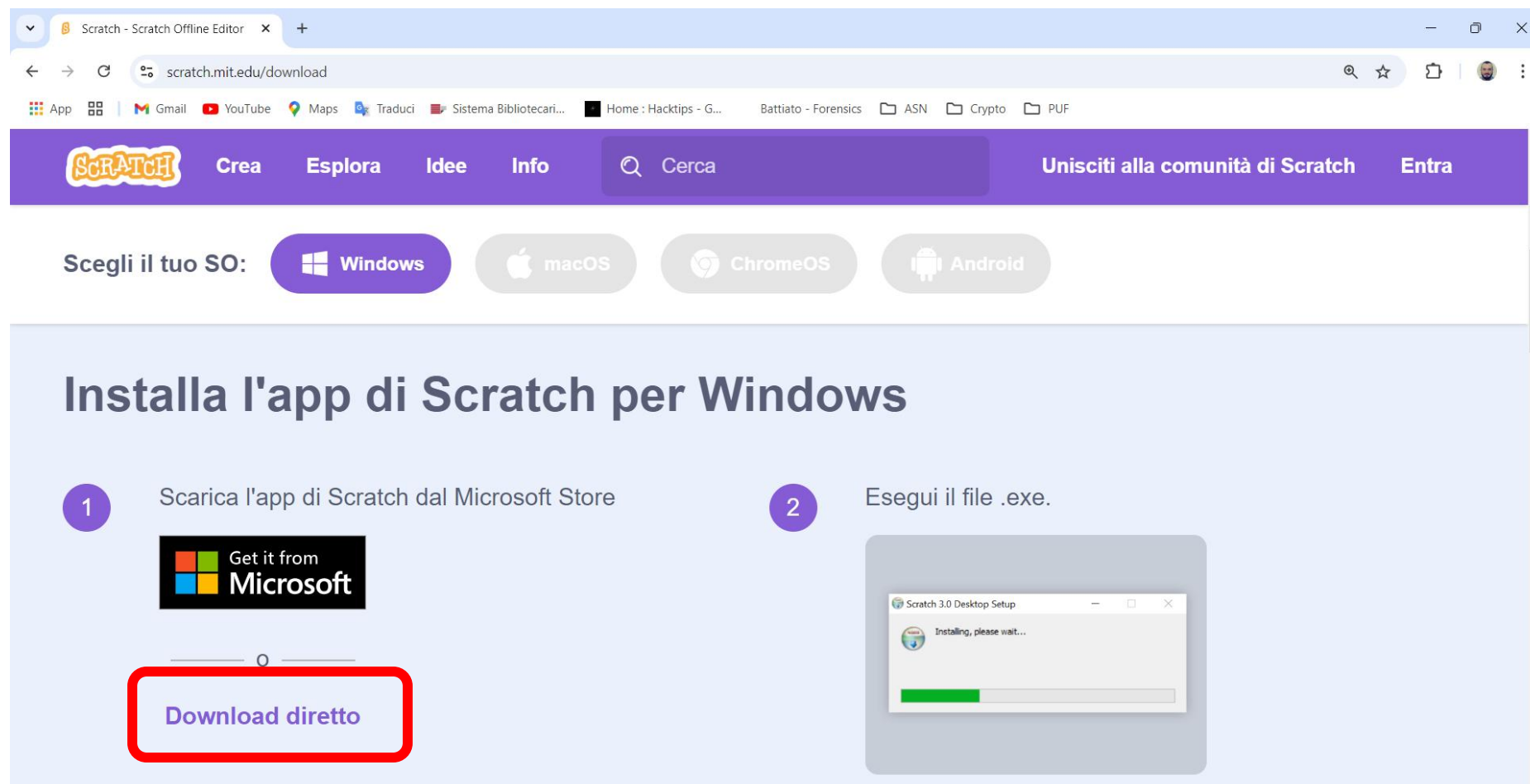
Scarica l'App di Scratch

Vuoi creare e salvare i progetti Scratch anche senza una connessione internet? Scarica l'app gratuita di Scratch.

Requisiti

- Windows 10+
- macOS 10.13+
- ChromeOS
- Android 6.0+

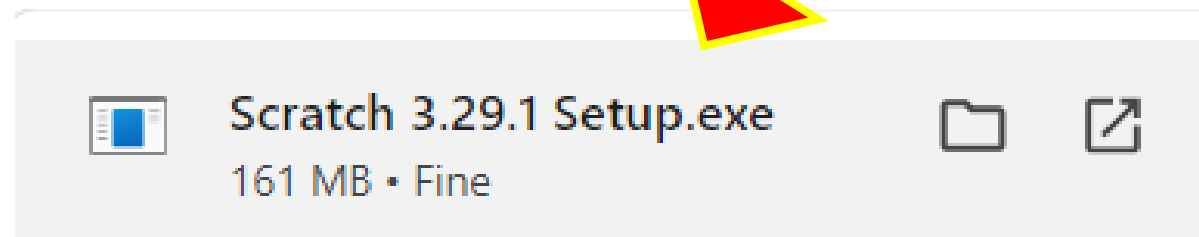
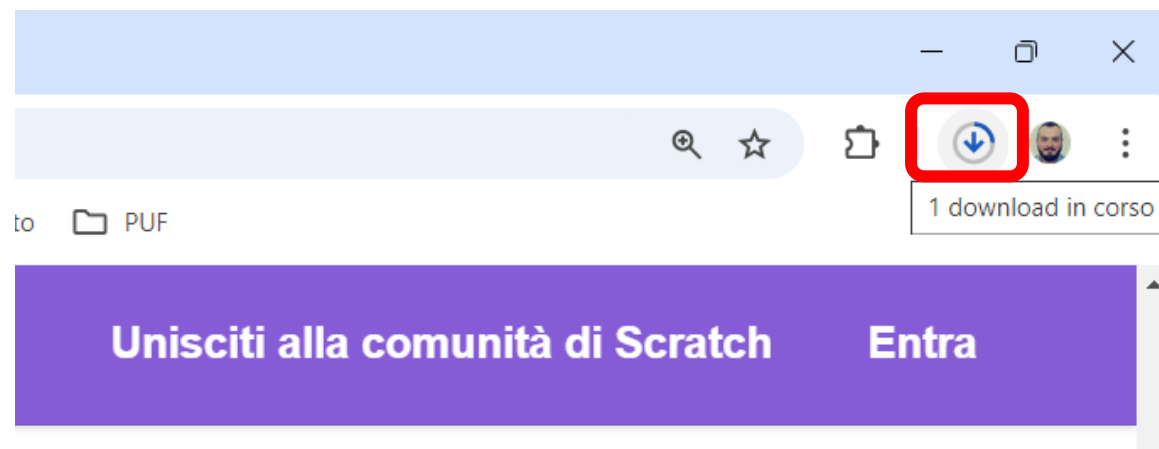
Scelta del Sistema Operativo



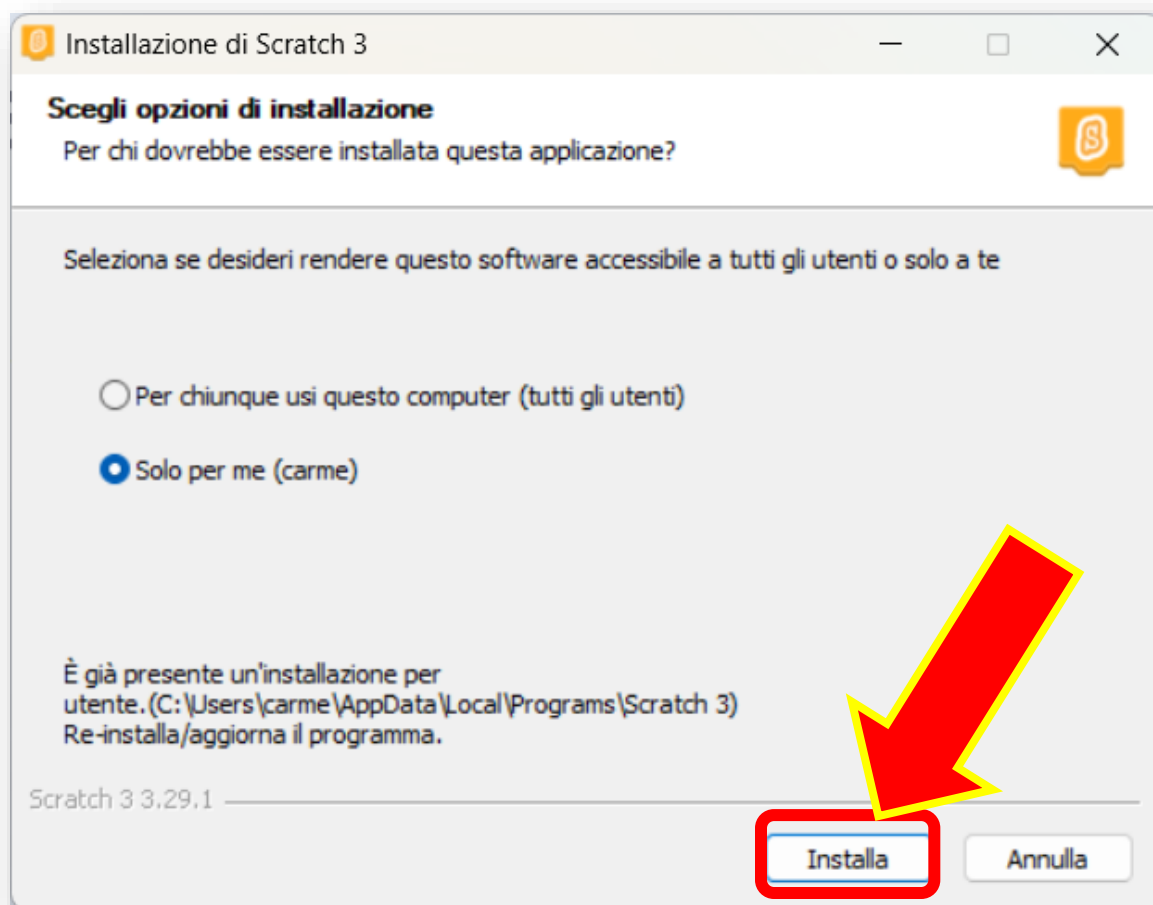
The screenshot shows the Scratch website's download page for Windows. The browser address bar shows 'scratch.mit.edu/download'. The page has a purple header with the Scratch logo and navigation links: 'Crea', 'Esplora', 'Idee', 'Info', 'Cerca', 'Unisciti alla comunità di Scratch', and 'Entra'. Below the header, there's a section 'Scegli il tuo SO:' with four buttons: 'Windows' (highlighted in purple), 'macOS', 'ChromeOS', and 'Android'. The main content area is titled 'Installa l'app di Scratch per Windows' and contains two steps:

- 1** Scarica l'app di Scratch dal Microsoft Store
This step includes a 'Get it from Microsoft' logo and a button labeled 'Download diretto' which is highlighted with a red rectangle.
- 2** Esegui il file .exe.
This step includes an image of the 'Scratch 3.0 Desktop Setup' window showing 'Installing, please wait...' with a progress bar.

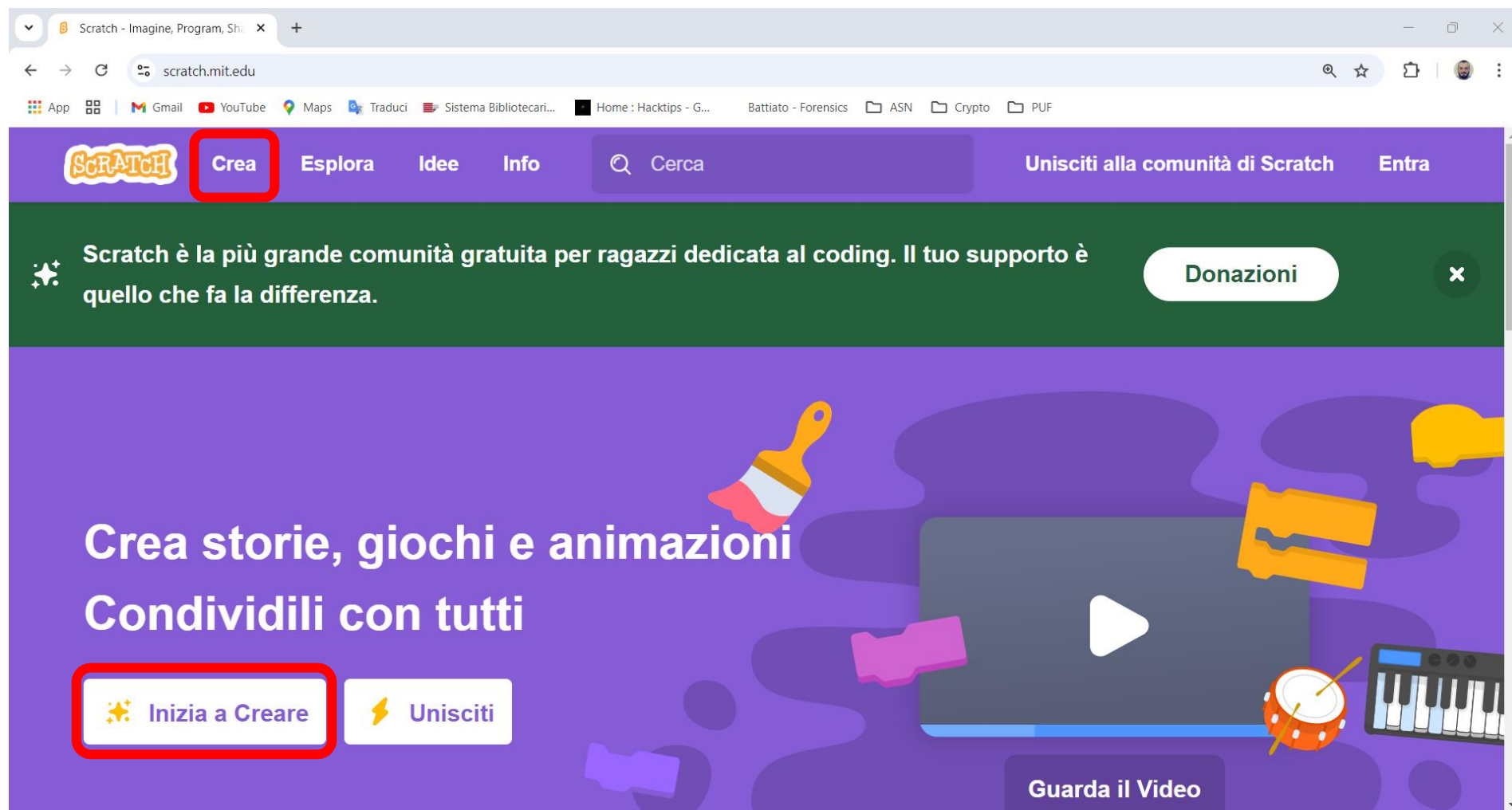
Download



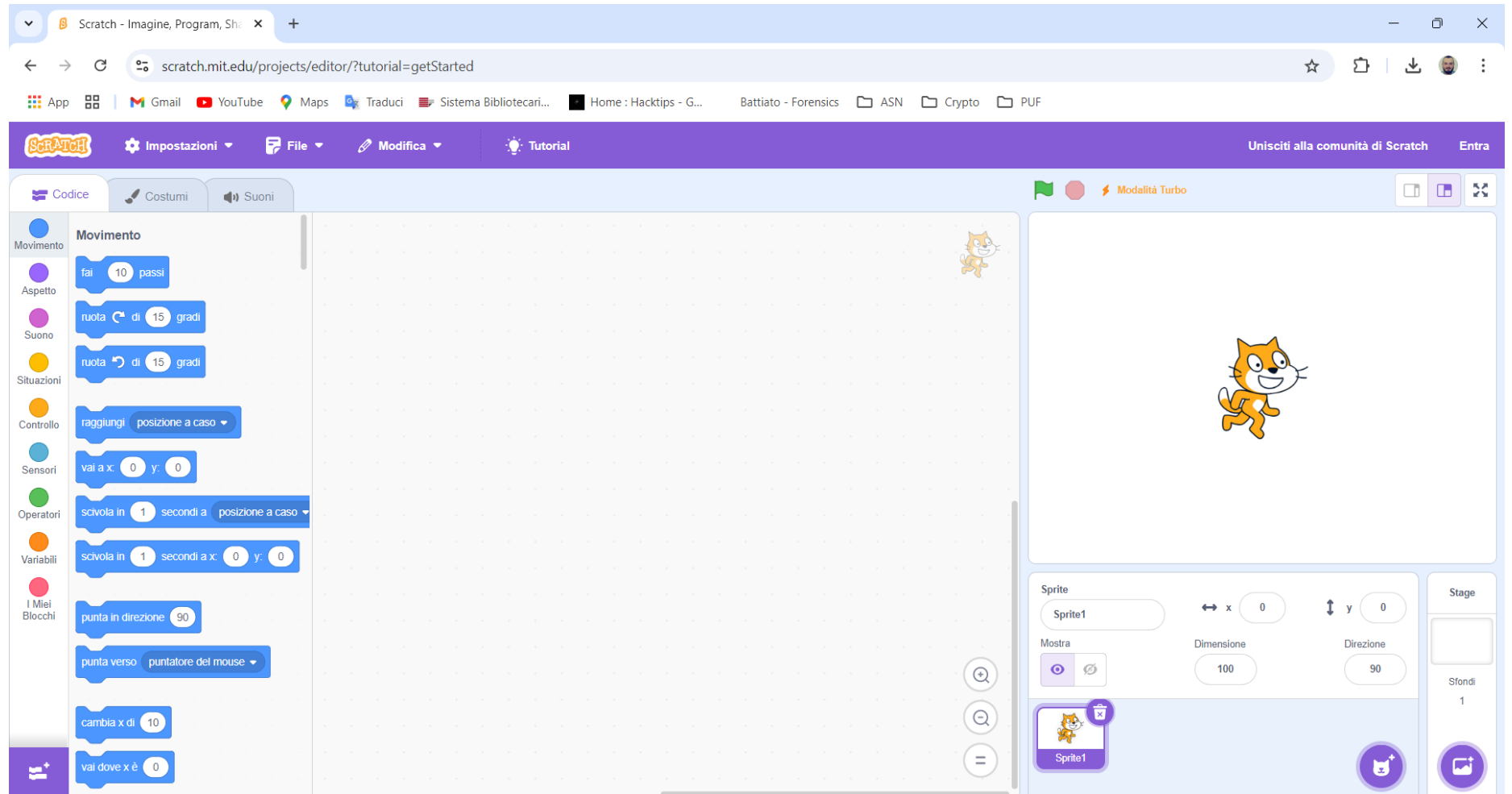
Installazione del programma



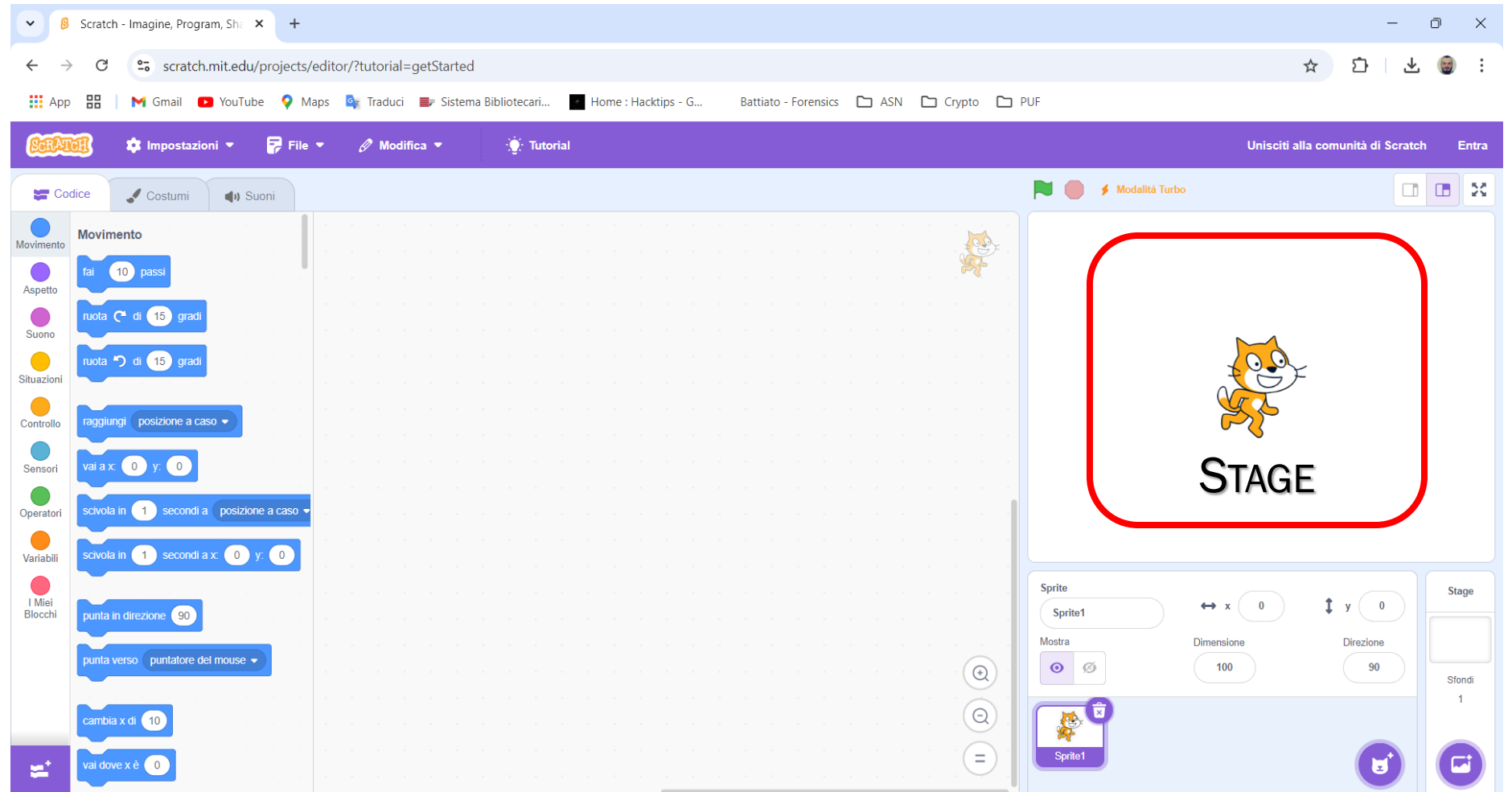
Iniziamo a creare!



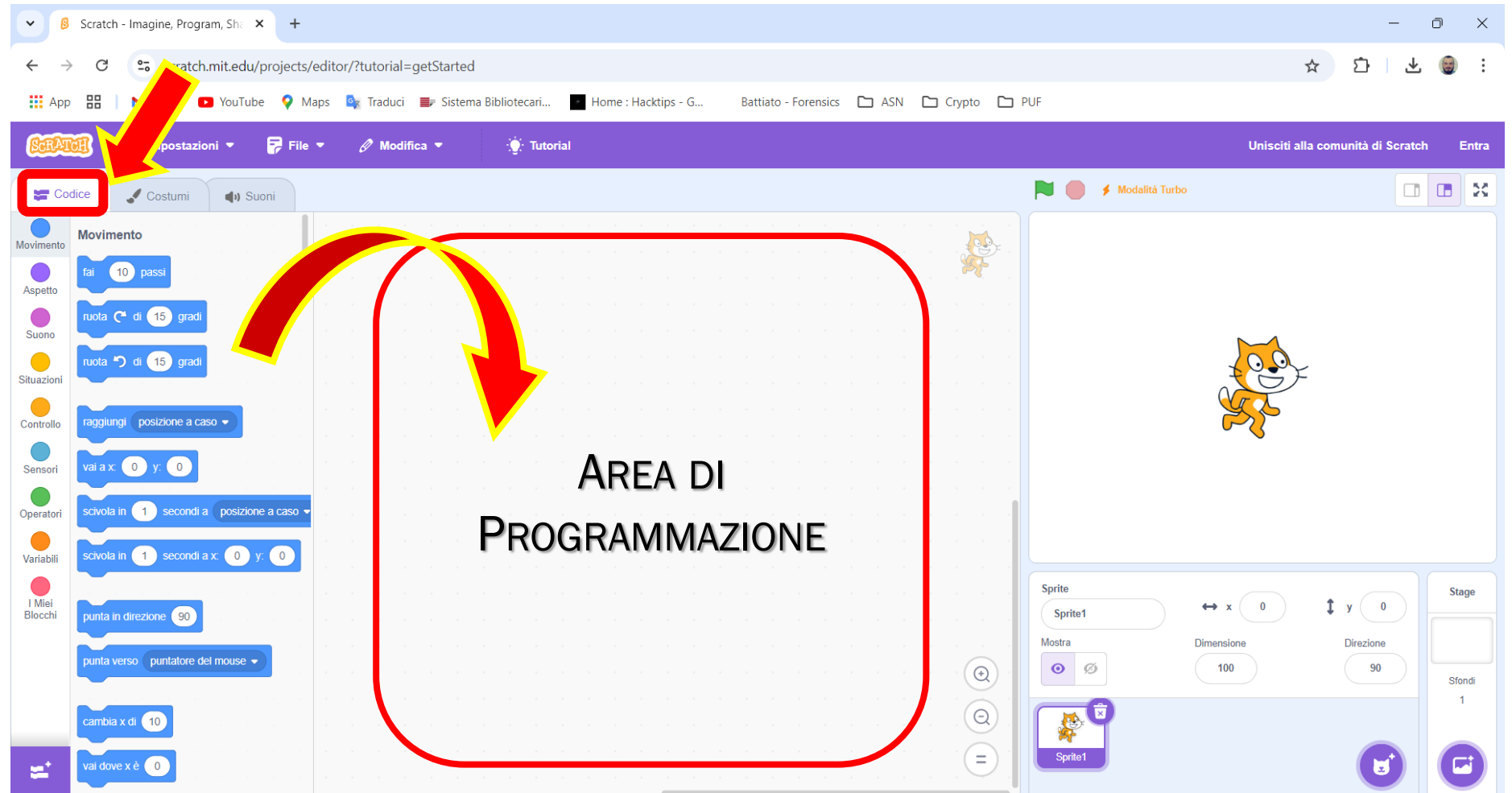
Interfaccia di Scratch



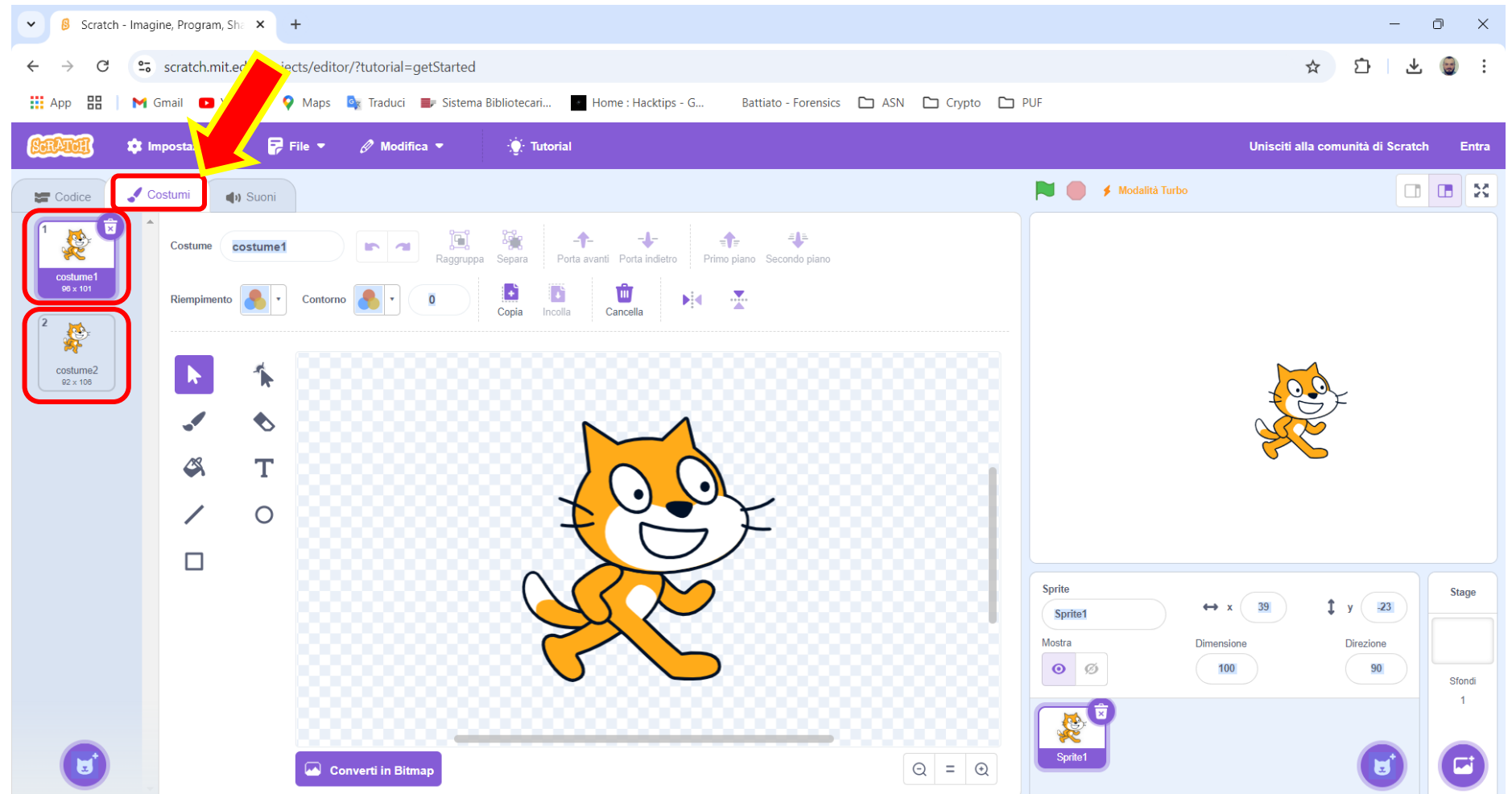
Stage



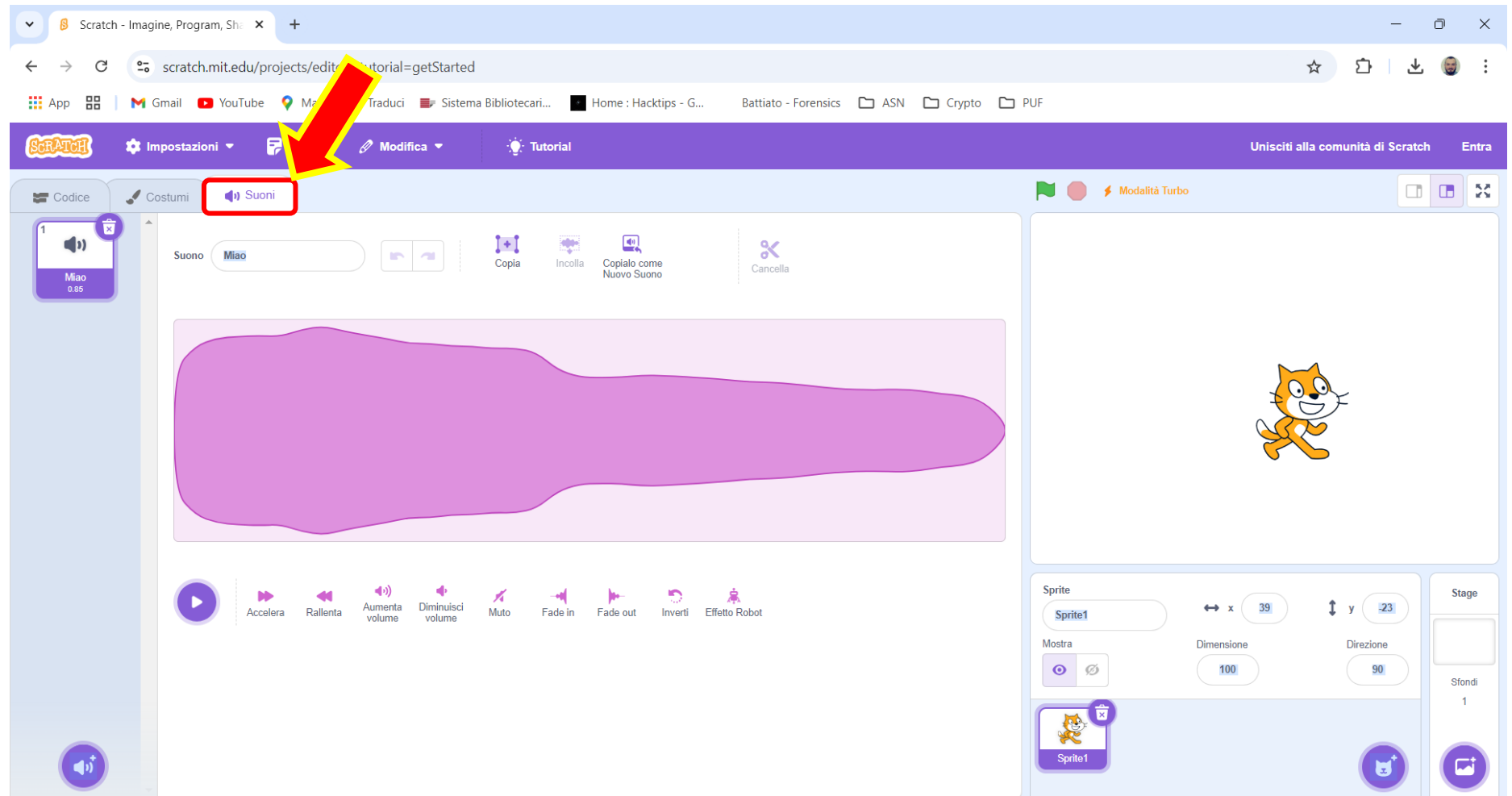
Area di programmazione



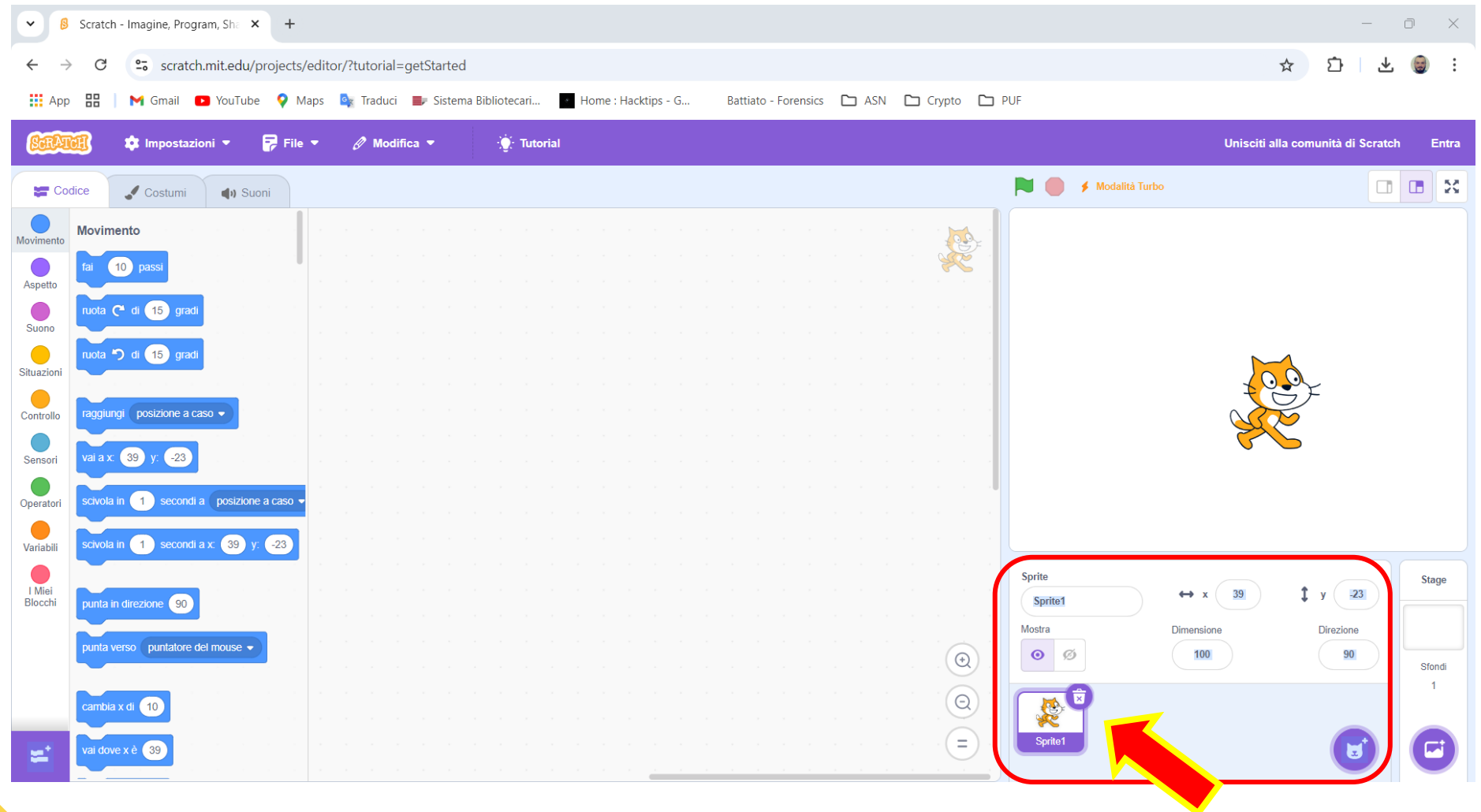
Costumi



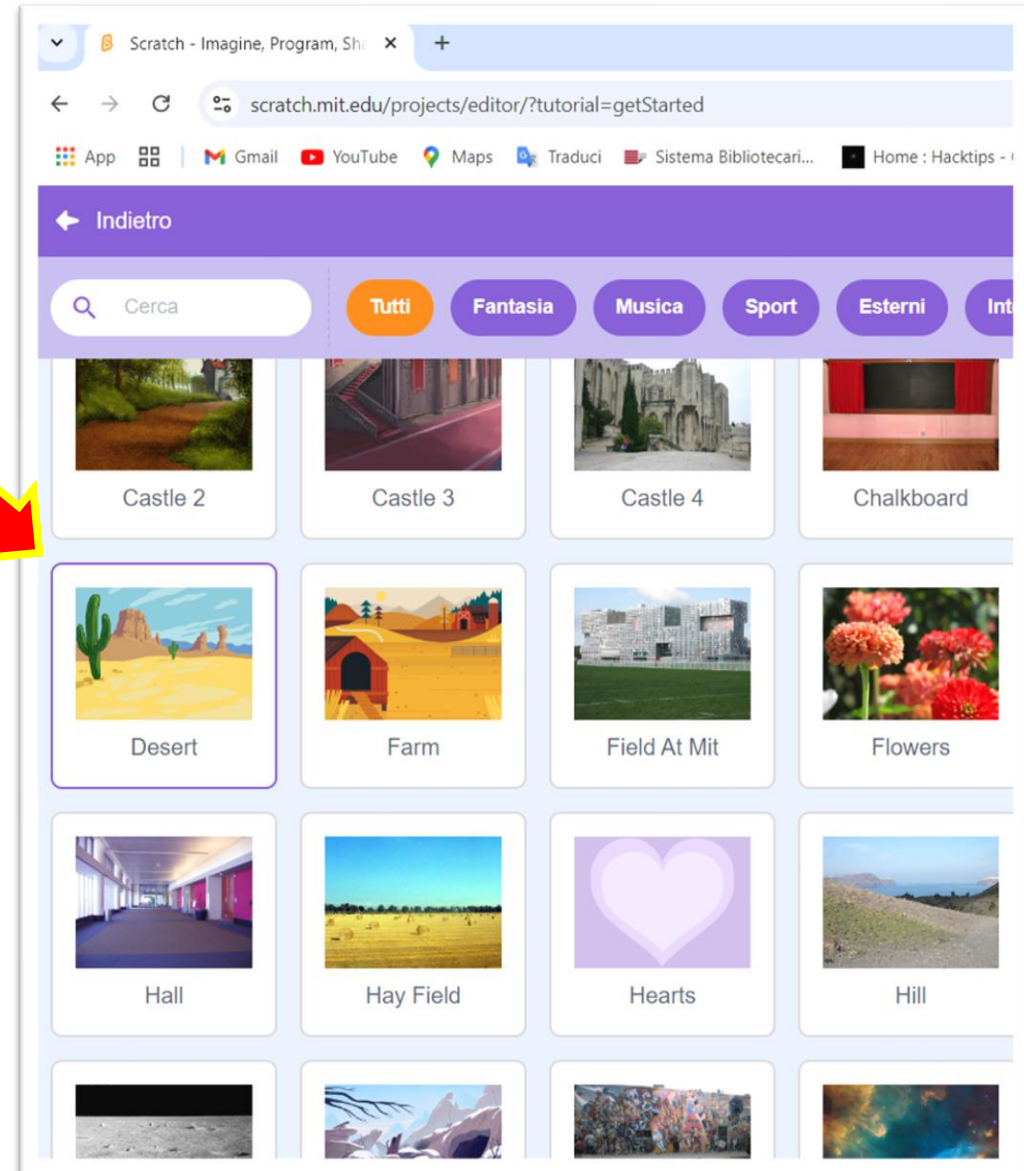
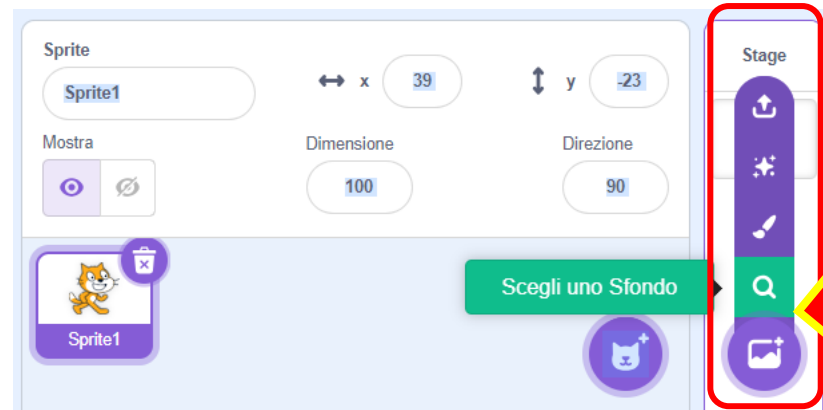
Suoni



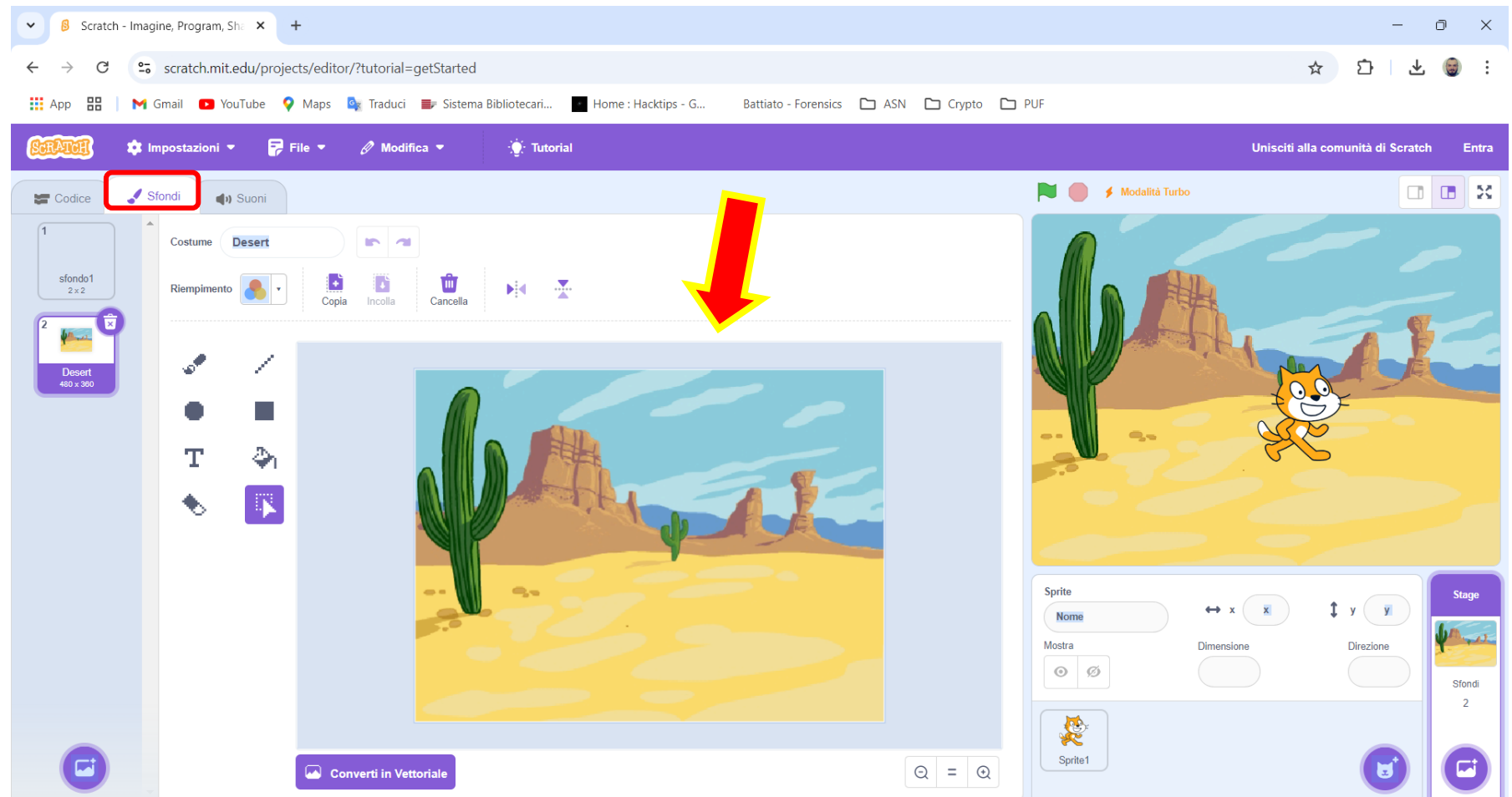
Sprite



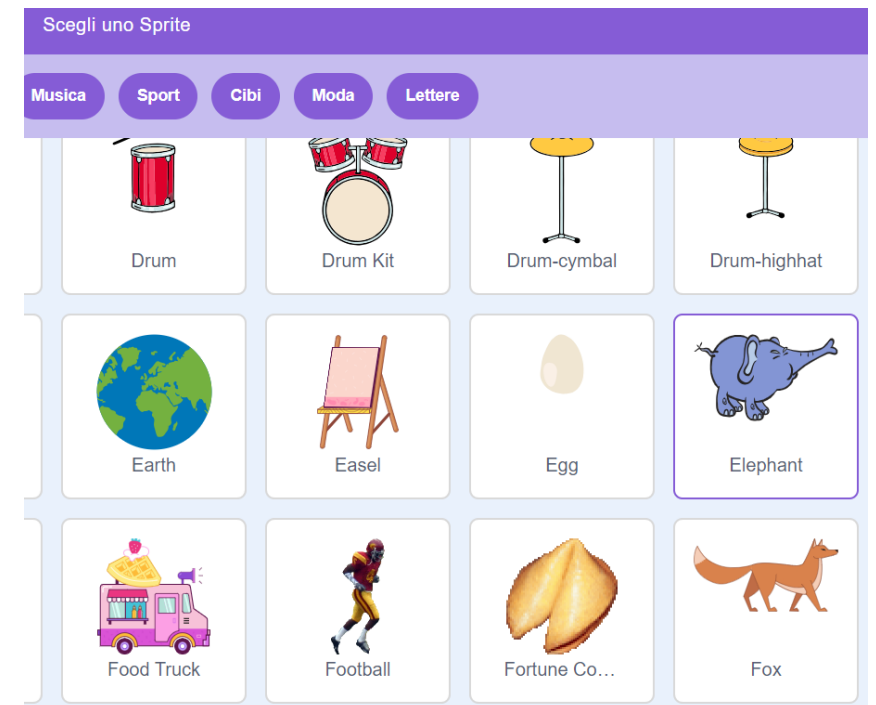
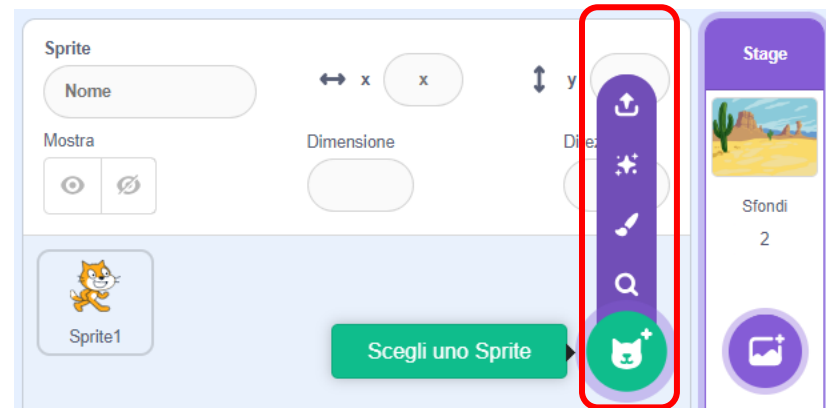
Sfondi



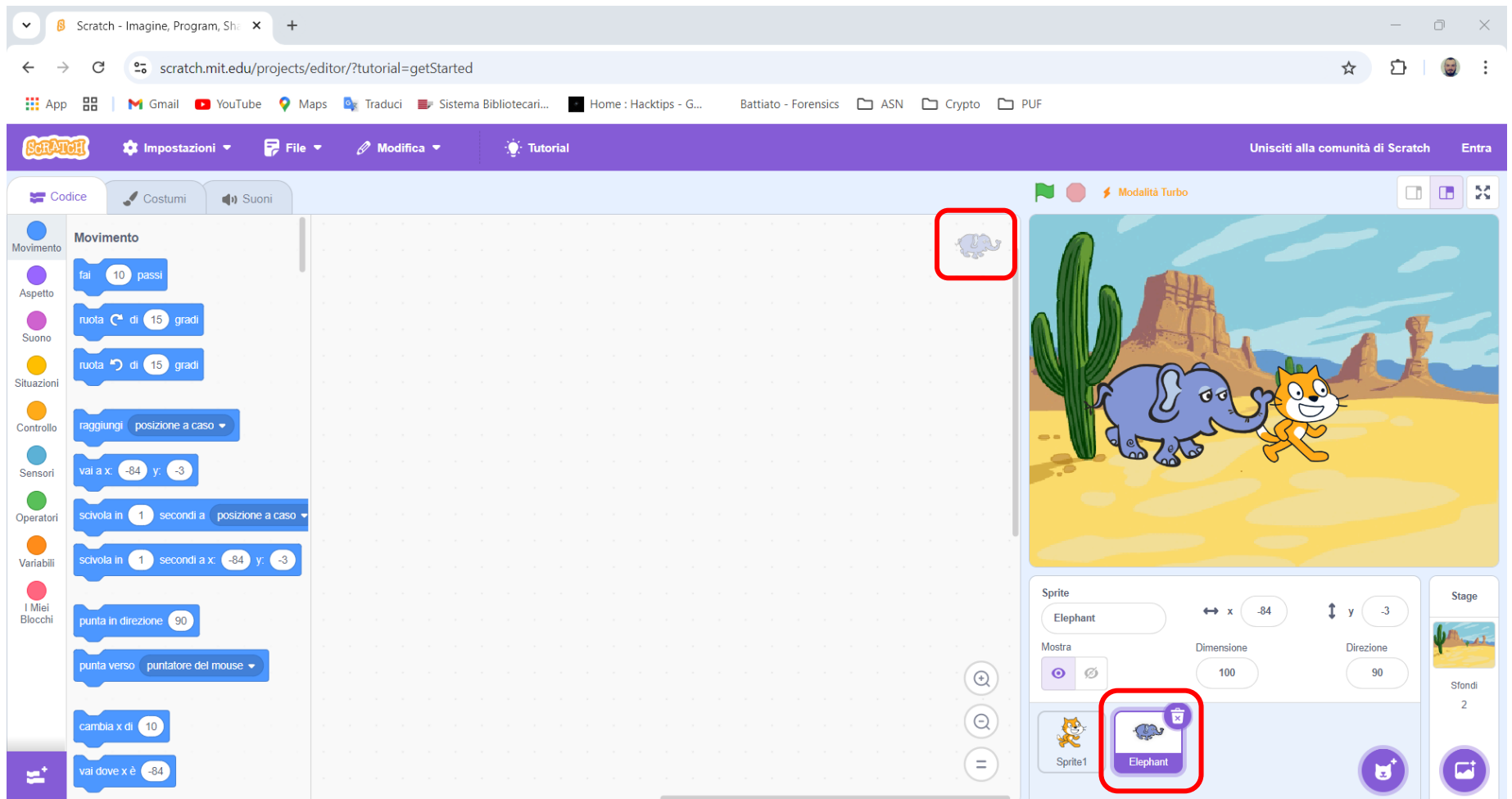
Personalizzazione



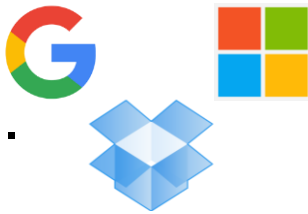
Personaggi multipli



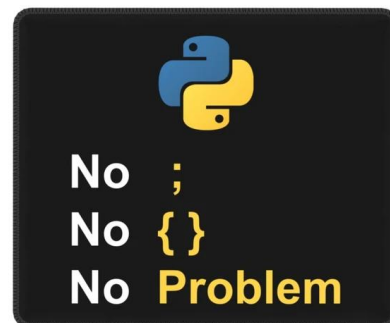
Personaggi multipli



Python



- Linguaggio di programmazione testuale sviluppato da Guido van Rossum.
- Software gratuito e open-source.
- Ampia gamma di applicazioni: ***sviluppo web, analisi dei dati, intelligenza artificiale.***



Programmazione per tutti

Nel 1999, Van Rossum presentò una proposta al DARPA chiamata ***Computer programming for everybody*** (programmazione per tutti), in cui definì i suoi obiettivi per Python:

- un linguaggio semplice, intuitivo e potente quanto i suoi maggiori avversari;
- codice sorgente aperto, in modo che ognuno avrebbe potuto partecipare al suo sviluppo;
- un codice facilmente comprensibile, come l'inglese parlato;
- ottimo per i compiti di tutti i giorni, poiché in grado di consentire tempi di sviluppo brevi.

Molti di questi punti sono stati già realizzati. Python è cresciuto, diventando un linguaggio popolare, particolarmente nell'ambiente di Internet.

Introduzione a Python



- **Cos'è?** Python è un linguaggio di programmazione **testuale** ad **alto livello**, noto per la sua **semplicità** e **leggibilità**. È ampiamente utilizzato nell'**industria** e nella **ricerca**, oltre che nell'**educazione**.
- **Come funziona?** Python utilizza una **sintassi minimalista** e **intuitiva**, il che lo rende uno dei linguaggi più accessibili per i principianti. Supporta una vasta gamma di applicazioni, dalla programmazione web, all'intelligenza artificiale.
- **Perché è utile nell'educazione?** Python è ideale per introdurre alla programmazione testuale in quanto consente di concentrarsi più facilmente sulla logica del programma piuttosto che sulle regole della sintassi. Inoltre, Python è un linguaggio versatile che permette di espandere le conoscenze verso settori avanzati come la **scienza dei dati** e la **robotica**.

Un nuovo ambiente di sviluppo

The image is a screenshot of a Google search results page for the query 'Thonny'. The search bar at the top shows 'Thonny' with a clear button (X), a voice search button (microphone), an image search button (camera), and a search button (magnifying glass). Below the search bar are tabs for 'Tutti', 'Immagini', 'Video', 'Notizie', 'Web', 'Libri', 'Finanza', and 'Strumenti'. The first search result is highlighted with a red rounded rectangle. It features the Thonny logo (a circle with 'Th'), the title 'Thonny', the URL 'https://thonny.org', a 'Traduci questa pagina' link, and the subtitle 'Thonny, Python IDE for beginners'. The description states: 'Thonny comes with Python 3.10 built in, so just one simple installer is needed and you're ready to learn programming. (You can also use a separate Python ...'. The second result is from Softonic, titled 'Thonny - Download', with the URL 'https://thonny.it.softonic.com'. Its description says: 'Thonny è un programma di sviluppo gratuito per PC creato da un sviluppatore indipendente che utilizza lo stesso nome. Si tratta di un ambiente di sviluppo ...'. Below these results is a section 'Le persone hanno chiesto anche' with a link 'Come installare Thonny?'. On the right side of the page is a 'Thonny Software' card. It shows the 'Th' logo, a screenshot of the Thonny IDE interface with Python code, and a button 'Altre immagini'. Below the card, there is a translation note: 'Tradotto dall'inglese - Thonny è un ambiente di sviluppo integrato gratuito e open source per Python progettato per i principianti. È stato creato da Aivar Annamaa, un programmatore estone. Wikipedia (inglese)'. At the bottom of the card, it says 'Vedi la descrizione originale' and 'Linguaggio di programmazione: Python'.

Google

Thonny

Tutti Immagini Video Notizie Web Libri Finanza Strumenti

Thonny
https://thonny.org · Traduci questa pagina
Thonny, Python IDE for beginners
Thonny comes with Python 3.10 built in, so just one simple installer is needed and you're ready to learn programming. (You can also use a separate Python ...

Softonic
https://thonny.it.softonic.com
Thonny - Download
Thonny è un programma di sviluppo gratuito per PC creato da un sviluppatore indipendente che utilizza lo stesso nome. Si tratta di un ambiente di sviluppo ...

Le persone hanno chiesto anche :
Come installare Thonny?

Feedback

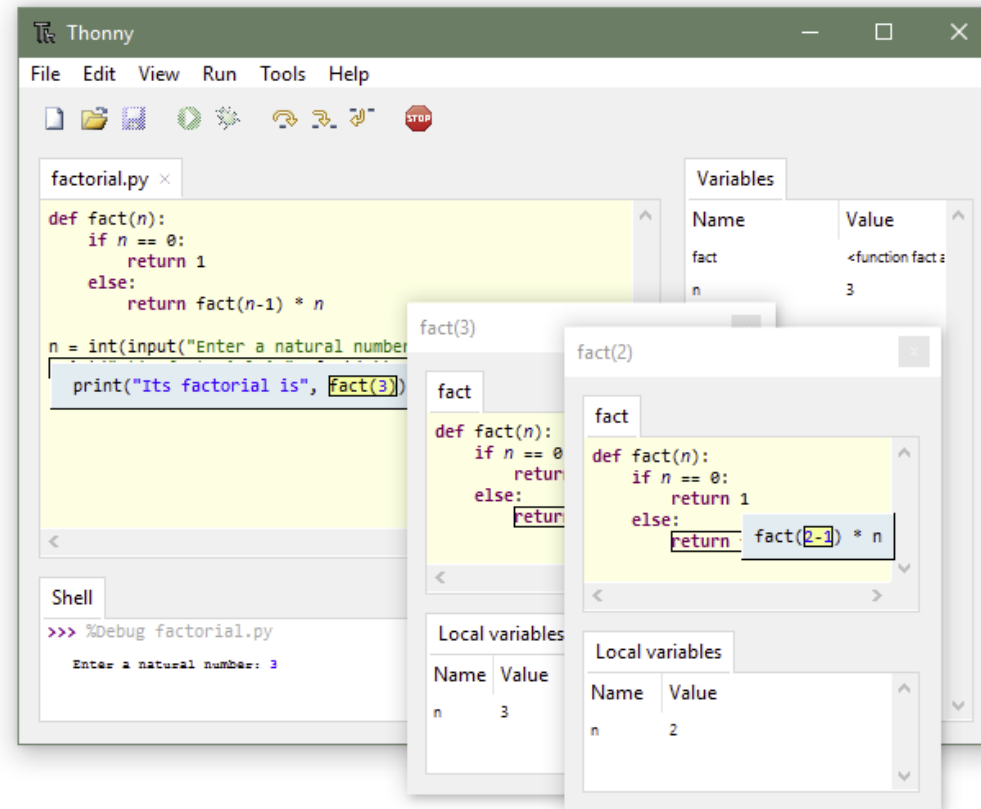
Thonny
Software
Tradotto dall'inglese - Thonny è un ambiente di sviluppo integrato gratuito e open source per Python progettato per i principianti. È stato creato da Aivar Annamaa, un programmatore estone. Wikipedia (inglese)
Vedi la descrizione originale
Linguaggio di programmazione: Python

Benvenuto Thonny (<https://thonny.org/>)

Thonny
Python IDE for beginners



Download version [4.1.6](#) for
[Windows](#) • [Mac](#) • [Linux](#)



Benvenuto Thonny (<https://thonny.org/>)

Thonny



Download version [4.1.6](#) for
Windows • Mac • Linux

Official downloads for Windows

Installer with 64-bit Python 3.10, requires 64-bit Windows 8.1 / 10 / 11
[thonny-4.1.6.exe \(21 MB\)](#) ← *recommended for you*

Installer with 32-bit Python 3.8, suitable for all Windows versions since 7
[thonny-py38-4.1.6.exe \(20 MB\)](#)

Portable variant with 64-bit Python 3.10
[thonny-4.1.6-windows-portable.zip \(31 MB\)](#)

Portable variant with 32-bit Python 3.8
[thonny-py38-4.1.6-windows-portable.zip \(29 MB\)](#)

Re-using an existing Python installation (for advanced users)
`pip install thonny`

Shell

```
>>> %Debug factorial.py
Enter a natural number: 3
```

Local variables

Name	Value
n	3

Local variables

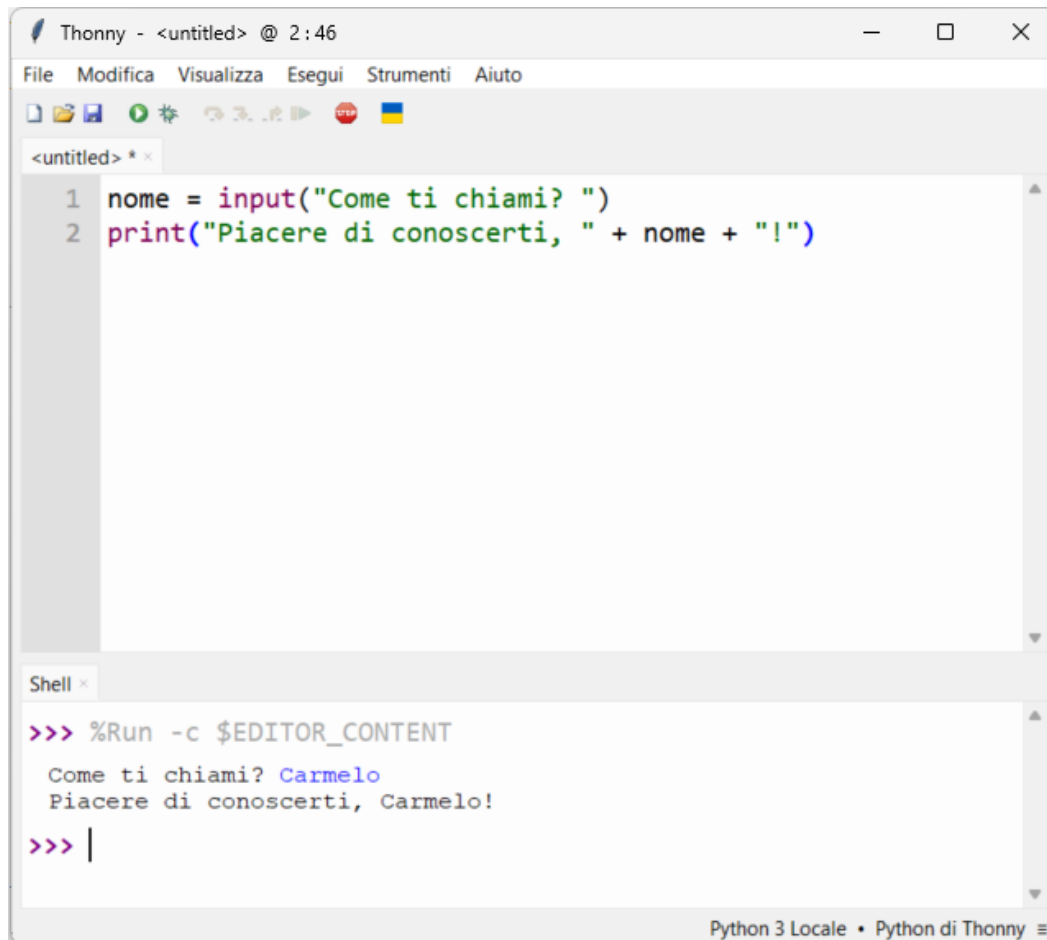
Name	Value
n	2

Primi Passi con Python



- **print()** è una funzione integrata in Python che permette di visualizzare messaggi o risultati sullo schermo.
- Le parentesi tonde **()** sono utilizzate per contenere il contenuto da visualizzare, che in questo caso è la stringa **"Hello, World!"**.
- Le stringhe di testo in Python sono racchiuse tra doppi apici **"** o apici singoli **'**. In questo esempio abbiamo utilizzato i doppi apici.

Variabili e Input dell'Utente



```
Thonny - <untitled> @ 2:46
File Modifica Visualizza Esegui Strumenti Aiuto
<untitled> * x
1 nome = input("Come ti chiami? ")
2 print("Piacere di conoscerti, " + nome + "!")

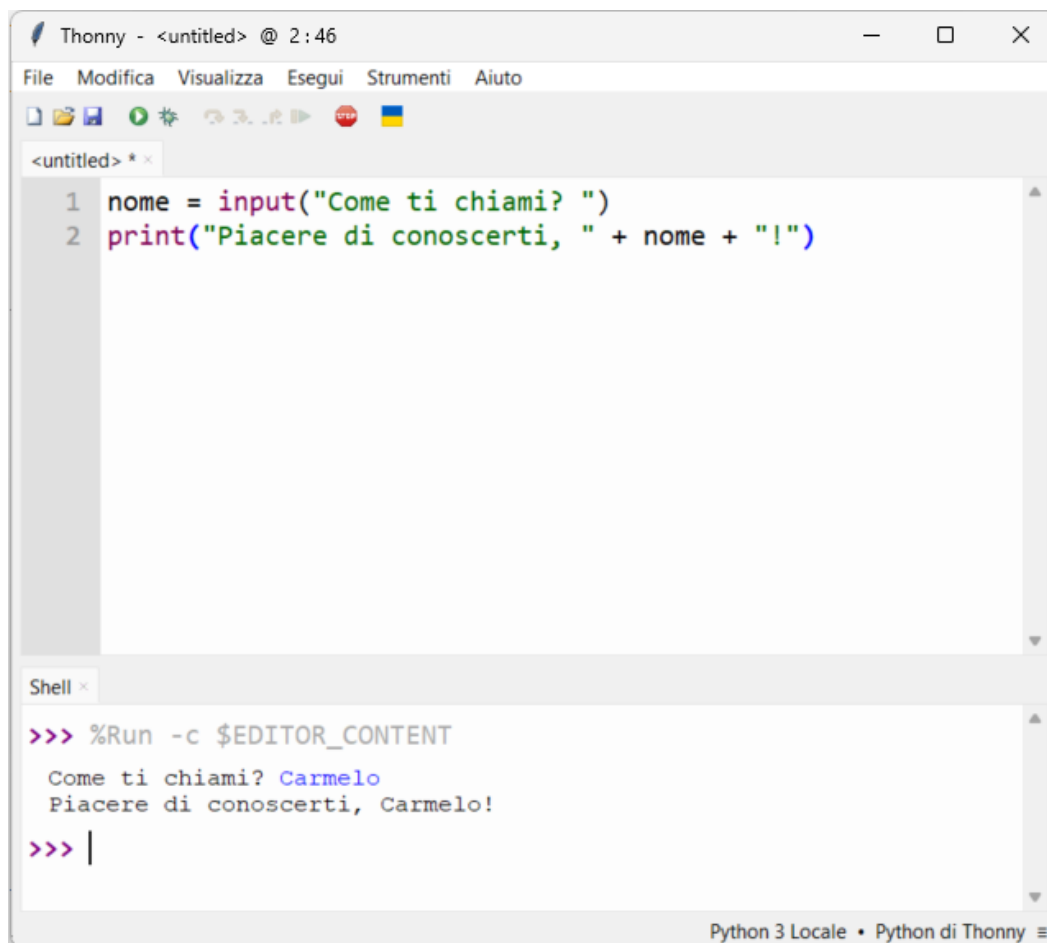
Shell x
>>> %Run -c $EDITOR_CONTENT
Come ti chiami? Carmelo
Piacere di conoscerti, Carmelo!
>>> |

Python 3 Locale • Python di Thonny
```

Dichiarazione di variabili:

- Una variabile è un contenitore che può memorizzare dati. In Python, le variabili vengono dichiarate semplicemente assegnando loro un valore (ad esempio, un numero o una stringa).

Variabili e Input dell'Utente



```
Thonny - <untitled> @ 2:46
File Modifica Visualizza Esegui Strumenti Aiuto
<untitled> * x
1 nome = input("Come ti chiami? ")
2 print("Piacere di conoscerti, " + nome + "!")

Shell x
>>> %Run -c $EDITOR_CONTENT
Come ti chiami? Carmelo
Piacere di conoscerti, Carmelo!
>>> |

Python 3 Locale • Python di Thonny
```

- **Utilizzo della funzione `input()`:**
 - La funzione `input()` viene utilizzata per ottenere input dall'utente. Il testo tra le parentesi, racchiuso tra virgolette, viene mostrato come un prompt per l'utente.
 - Il valore che l'utente inserisce viene poi memorizzato nella variabile.

Variabili e Input dell'Utente

A screenshot of a Python shell window titled "Shell x". The window shows the execution of a Python script. The first line is a prompt ">>> %Run -c \$EDITOR_CONTENT". The script then prompts the user with "Come ti chiami? Carmelo" and prints "Piacere di conoscerti, Carmelo!". The prompt ">>> |" is shown on the next line. The bottom status bar indicates "Python 3 Locale • Python di Thonny".

```
Shell x

>>> %Run -c $EDITOR_CONTENT
Come ti chiami? Carmelo
Piacere di conoscerti, Carmelo!
>>> |

Python 3 Locale • Python di Thonny
```

nome = input("Come ti chiami? "):

- Questo chiede all'utente di inserire il proprio nome. Il testo inserito viene memorizzato nella variabile **nome**.

print("Piacere di conoscerti, " + nome + "!")

- Qui, la funzione **print()** stampa un messaggio personalizzato combinando una stringa fissa con il valore della variabile **nome**, utilizzando l'operatore di concatenazione **+** per unire le stringhe.

In questo modo, l'utente interagisce con il programma inserendo dati, che vengono poi utilizzati nel flusso dell'applicazione.

Strutture di Controllo (Condizioni (if, else))

Le condizioni in Python vengono utilizzate per prendere decisioni in base ai valori delle variabili o ai risultati di espressioni logiche. Le parole chiave `if` e `else` consentono al programma di eseguire blocchi di codice differenti a seconda della condizione.

```
if condizione:
    # Codice da eseguire se la condizione è vera
else:
    # Codice da eseguire se la condizione è falsa
```

```
età = int(input("Quanti anni hai? "))
if età >= 18:
    print("Sei maggiorenne.")
else:
    print("Sei minorenne.")
```

Punti chiave:

- **Indentazione:** In Python, il codice all'interno dei blocchi `if` e `else` deve essere indentato (di solito 4 spazi) per segnalare che appartiene a quei blocchi.
- **Operatori logici:** In questo esempio, usiamo l'operatore `>=` per confrontare il valore della variabile `età` con 18. Altri operatori includono `<`, `>`, `==` (uguale a), `!=` (diverso da).

Iterazioni (cicli for e while)

Le iterazioni permettono di eseguire un'azione più volte senza ripetere manualmente le istruzioni. Questo è utile per evitare ridondanze nel codice e semplificare operazioni ripetitive.

Il ciclo `for` è utilizzato per ripetere un'azione un numero specifico di volte. In Python, è comune usare `range()` per creare un intervallo di numeri su cui iterare.

Il ciclo `while` ripete un blocco di codice finché una condizione rimane vera. È utile quando non si sa a priori quante volte sarà necessario ripetere l'operazione.

```
for i in range(5):  
    print("Questo è il messaggio numero", i + 1)
```

```
i = 1  
while i <= 5:  
    print("Questo è il messaggio numero", i)  
    i += 1
```

Punti chiave:

- **Condizioni di Termine:** Nei cicli, è importante avere una condizione che fermi l'iterazione. Un ciclo senza una condizione che si verifichi porterà a un ciclo infinito.
- **Punti di forza:** L'uso delle iterazioni permette di scrivere codice più efficiente e compatto, riducendo la ripetizione e migliorando la leggibilità.

Spiegazione esempio ciclo for

```
for i in range(5):  
    print("Questo è il messaggio numero", i + 1)
```

for i in range(5):

Questo ciclo ripete il blocco di codice sottostante 5 volte.

La funzione range(5) genera i numeri da 0 a 4 (5 escluso).

print("Questo è il messaggio numero", i + 1)

Ad ogni iterazione, il valore di i cambia, e stampiamo un messaggio che include il numero corrente dell'iterazione.

Aggiungiamo + 1 perché vogliamo che i messaggi siano numerati da 1 a 5 anziché da 0 a 4.

Attività:

- Modificare il ciclo per stampare numeri da 1 a 10.
- Creare un ciclo che stampa solo i numeri pari compresi tra 1 e 10.

Spiegazione esempio ciclo while

```
i = 1
while i <= 5:
    print("Questo è il messaggio numero", i)
    i += 1
```

i = 1

Prima di iniziare il ciclo, inizializziamo una variabile i a 1.

while i <= 5:

Il ciclo continua fino a quando il valore di i è minore o uguale a 5.

i += 1

Ad ogni iterazione, incrementiamo il valore di i di 1 (altrimenti il ciclo non terminerebbe mai).

Attività:

- Modificare il ciclo per contare all'indietro da 10 a 1.
- Creare un programma che chiede all'utente di indovinare un numero segreto, continuando finché l'utente non indovina.

Discussione

- Il ciclo **for** è generalmente utilizzato quando **si sa quante volte eseguire un'operazione**, mentre il ciclo **while** viene utilizzato quando si continua **fino a quando una condizione specifica è soddisfatta**.
- È importante comprendere come evitare cicli infiniti, che si verificano quando la condizione del ciclo **non diventa mai falsa** (ad esempio, dimenticare di aggiornare la variabile in un ciclo while).

Approfondimenti

- *Intervallo di Numeri con range():*

range(start, stop, step):

La funzione `range()` viene spesso utilizzata nei cicli for per iterare su un intervallo di numeri. Permette di definire un inizio (start), una fine (stop, esclusa) e un passo (step).

- *Cicli Annidati:*

È possibile avere cicli all'interno di altri cicli (cicli annidati). Questo è utile per iterare su strutture più complesse, come matrici o liste di liste.

- *Uscita Anticipata dai Cicli (break e continue)*

break: Interrompe immediatamente l'esecuzione del ciclo, usato quando si verifica una condizione particolare.

continue: Salta alla prossima iterazione, utile per saltare specifiche condizioni senza interrompere tutto il ciclo.

Uso di **break** e **continue** nei Cicli

- **break**: Interrompe completamente l'esecuzione del ciclo quando si verifica una certa condizione.

```
# Esempio con break  
for numero in range(10):  
    if numero == 5:  
        break  
    print(numero)
```

Uso di break e continue nei Cicli

- **continue:** Salta l'iterazione corrente e passa alla successiva.

```
# Esempio con continue  
for numero in range(10):  
    if numero % 2 == 0:  
        continue  
    print(numero)
```

Esercizi sulle Sequenze e Variabili

- Dichiarazione di Variabili e Calcoli Semplici

```
numero1 = float(input("Inserisci il primo numero: "))  
numero2 = float(input("Inserisci il secondo numero: "))
```

```
somma = numero1 + numero2  
differenza = numero1 - numero2  
prodotto = numero1 * numero2  
quoziente = numero1 / numero2
```

```
print("La somma è:", somma)  
print("La differenza è:", differenza)  
print("Il prodotto è:", prodotto)  
print("Il quoziente è:", quoziente)
```

Esercizi sulle Sequenze e Variabili

- Conversioni e Sequenze

```
chilometri = float(input("Inserisci la distanza in chilometri: "))
metri = chilometri * 1000
centimetri = chilometri * 100000

print("Distanza in metri:", metri)
print("Distanza in centimetri:", centimetri)
```

Condizioni (If-Else)

- Condizioni di Base

```
parola = input("Inserisci una parola: ")
```

```
if len(parola) > 5:  
    print("Parola lunga")  
else:  
    print("Parola breve")
```

Condizioni (If-Else)

- Condizioni Multiple

```
numero = float(input("Inserisci un numero: "))
if numero > 0:
    print("Il numero è positivo.")
elif numero < 0:
    print("Il numero è negativo.")
else:
    print("Il numero è zero.")
```

Iterazioni: Cicli For e While

- Ciclo *For* per Sequenze

```
for i in range(1, 11):  
    print(i)
```

- Ciclo *While* per il Controllo Ripetitivo

```
parola_segreta = "python"  
tentativo = ""  
  
while tentativo != parola_segreta:  
    tentativo = input("Indovina la parola segreta: ")  
print("Complimenti, hai indovinato!")
```

- Calcolo Progressivo

```
somma = 0  
for i in range(2, 21, 2):  
    somma += i  
print("La somma dei numeri pari è:", somma)
```

Gioco "Indovina il numero"

Il computer sceglie un numero casuale tra 1 e 100 e l'utente deve indovinare. Il ciclo continua fino a quando l'utente indovina, e il programma fornisce suggerimenti ("troppo alto", "troppo basso")

```
import random
numero_segreto = random.randint(1, 100)
tentativo = None

while tentativo != numero_segreto:
    tentativo = int(input("Indovina il numero: "))
    if tentativo < numero_segreto:
        print("Troppo basso!")
    elif tentativo > numero_segreto:
        print("Troppo alto!")
    else:
        print("Hai indovinato!")
```

Cicli Annidati

- I cicli annidati permettono di eseguire iterazioni multiple, una dentro l'altra. Questo è utile per lavorare con strutture di dati come matrici o per iterare su più dimensioni.

```
matrice = [  
    [1, 2, 3],  
    [4, 5, 6],  
    [7, 8, 9]  
]  
  
for riga in matrice:  
    for elemento in riga:  
        print(elemento, end=" ")  
    print() # Nuova riga dopo ogni riga
```

- Creare una matrice 3x3 e stamparla.
- Modificare l'esercizio in modo che vengano stampati solo i numeri pari della matrice

Esempi Applicativi di Natura Pratica

- **Elaborazione di Liste:** Contare elementi, trovare il massimo/minimo, ordinare liste.
- **Ciclo su Dati:** Iterare su file di dati, leggere righe o colonne di un foglio di calcolo.
- **Simulazioni:** Ripetere calcoli o eventi fino a una determinata condizione.

Applicazioni Didattiche: Matematica

```
import numpy as np

def funzione(x):
    return x**2 + 3*x + 2

x_values = np.arange(-10, 11, 1)
for x in x_values:
    print(f"f({x}) = {funzione(x)}")
```

Applicazioni Didattiche: Matematica 2

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-10, 10, 400)
y = x**2 + 3*x + 2

plt.plot(x, y)
plt.title("Grafico della funzione")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.grid(True)
plt.show()
```

Applicazioni Didattiche: Fisica

Simulazione del Moto Uniforme

```
tempo = range(0, 11)
velocità = 5 # m/s
for t in tempo:
    distanza = velocità * t
    print(f"Al tempo {t} secondi, l'oggetto si trova a {distanza} metri.")
```

Applicazioni Didattiche: Fisica 2

Simulazione della Caduta Libera

```
g = 9.81 # accelerazione di gravità (m/s^2)
t_max = 5 # durata della caduta (secondi)

for t in range(t_max + 1):
    h = 0.5 * g * t**2
    print(f"Al tempo {t} secondi, l'oggetto ha percorso {h} metri.")
```

Applicazioni Didattiche: Economia

Calcolo dell'interesse

```
P = 1000 # Capitale iniziale
r = 0.05 # Tasso d'interesse annuale
n = 10   # Anni

for i in range(1, n+1):
    P = P * (1 + r)
    print(f"Dopo {i} anni, il capitale è: {P:.2f}")
```

Applicazioni Didattiche: Economia 2

Analisi dei Dati Finanziari

```
import pandas as pd

# Creazione di un piccolo DataFrame per mostrare l'analisi di semplici dati economici
data = {'Anno': [2018, 2019, 2020, 2021],
        'Vendite': [1000, 1200, 1100, 1300]}

df = pd.DataFrame(data)
print(df)

# Calcolo della crescita percentuale
df['Crescita'] = df['Vendite'].pct_change() * 100
print(df)
```

Applicazioni Didattiche: Chimica

Calcolo delle Moli

```
massa = float(input("Inserisci la massa (g): "))  
peso_molare = float(input("Inserisci il peso molare (g/mol): "))  
  
moli = massa / peso_molare  
print(f"Il numero di moli è: {moli:.2f}")
```

Applicazioni Didattiche: Informatica/Tecnologia



Gioco: Carta Forbice Sasso

```
import random

scelte = ["carta", "forbice", "sasso"]

utente = input("Scegli: carta, forbice o sasso? ")
computer = random.choice(scelte)

print(f"Il computer ha scelto: {computer}")

if utente == computer:
    print("È un pareggio!")
elif (utente == "carta" and computer == "sasso") or \
      (utente == "forbice" and computer == "carta") or \
      (utente == "sasso" and computer == "forbice"):
    print("Hai vinto!")
else:
    print("Hai perso!")
```

Applicazioni Didattiche: Italiano

Conteggio delle Parole in un Testo

```
testo = input("Inserisci un testo: ")  
parole = testo.split()  
print(f"Il numero di parole è: {len(parole)}")
```