Labforweb

nerd academy

Introduzione alla programmazione

Cos'è la “programmazione”?

La programmazione è il processo di scrivere istruzioni che un computer (più in generale un dispositivo elettronico) può seguire e interpretare al fine di svolgere dei compiti.

Affinché un computer possa interpretare correttamente i nostri ordini è necessario utilizzare un linguaggio di programmazione a esso comprensibile.

Poiché esistono diverse tipologie di dispositivi elettronici, esistono anche linguaggi di programmazione diversi, anche se nella maggior parte dei casi i linguaggi tutti hanno in comune la maggior parte delle regole logiche e sintattiche.

Cosa si può fare la programmazione?

- **Automatizzare compiti ripetitivi:** la programmazione permette di creare programmi che eseguono attività noiose o ripetitive al posto delle persone.
- **Controllare dispositivi:** gestire elettrodomestici (lavatrici, forni, luci intelligenti) o macchinari complessi (ad esempio quelli di una fabbrica).
- **Creare software:** applicazioni per computer, smartphone o tablet che aiutano a risolvere problemi o a intrattenere.
- **Elaborare dati**: Analizzare grandi quantità di informazioni, fare calcoli complessi o cercare modelli utili.
- **Comunicare tra dispositivi**: permettere a diverse macchine di *parlare* tra loro, come nei sistemi di smart home o nei server web.

Esiste un linguaggio di programmazione migliore di un altro?

Non esiste un linguaggio di programmazione "migliore" in assoluto, ma ci sono linguaggi che sono più adatti a specifici compiti o contesti. Ogni linguaggio ha i suoi punti di forza e di debolezza, e la scelta dipende da ciò che si vuole realizzare.

Perché nessun linguaggio è il migliore?

Specializzazione: Alcuni linguaggi sono progettati per specifici settori. Ad esempio:

- JavaScript: ideale per lo sviluppo di siti web e applicazioni interattive.
- Python: ottimo per analisi dati, intelligenza artificiale e apprendimento automatico.
- C/C++: per sistemi operativi e software ad alte prestazioni.

Comunità e risorse: linguaggi con una grande comunità (come Java, JavaScript e PHP) hanno molte risorse di supporto, ma la "grandezza" non li rende sempre la scelta migliore per un progetto specifico.

Ci sono linguaggi più importanti?

Alcuni linguaggi sono considerati fondamentali o strategici perché sono:

- **Ampiamente utilizzati**: come JavaScript per il web o Python per la scienza dei dati.
- **Richiesti sul mercato**: molti lavori tech richiedono conoscenza di Java, PHP, Python, C#, o JavaScript.
- **Versatili**: linguaggi come Java o Python possono essere usati in campi diversi.

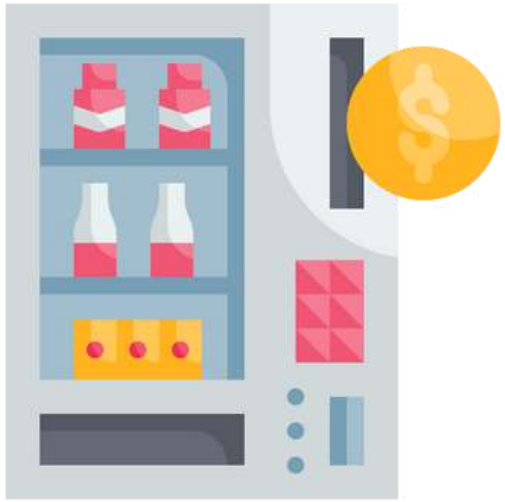
In sintesi

Imparare le basi della programmazione è più importante del linguaggio specifico. Una volta che capisci come pensare da programmatore è più facile imparare nuovi linguaggi.

Com’è strutturato un programma?

Rispondere a questa domanda in un solo colpo è impossibile, ma per semplificare il più possibile il concetto di programma lo sintetizziamo in **3 step**:

Input



Scelgo la bevanda

Logica

La vending machine prepara la bevanda scelta

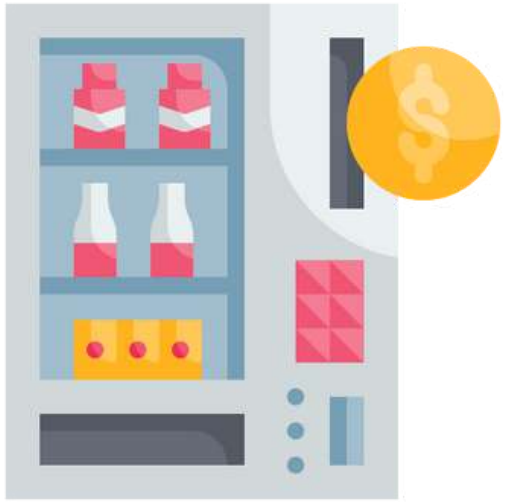
Output

Bevo (utilizzo) ciò che la macchina ha prodotto

Com’è strutturato un programma?

In questo caso il programmatore ha il compito di **scrivere le istruzioni** (logica) necessarie a rendere autonoma la vending machine di **preparare la bevanda** (output) sulla base della **scelta fatta dal consumatore** (input) subito dopo avere inserito la moneta

Input



Logica

Output

Com’è strutturato un programma?

Di solito, l’input (la scelta dell’utente) e l’output (il risultato prodotto) sono le parti più semplici da gestire in un programma. **La vera sfida è nella logica**, dove il programmatore collega l’input all’output definendo regole, condizioni e processi. È qui che si concentra il maggior impegno creativo e tecnico, richiedendo tempo e attenzione.

Logica

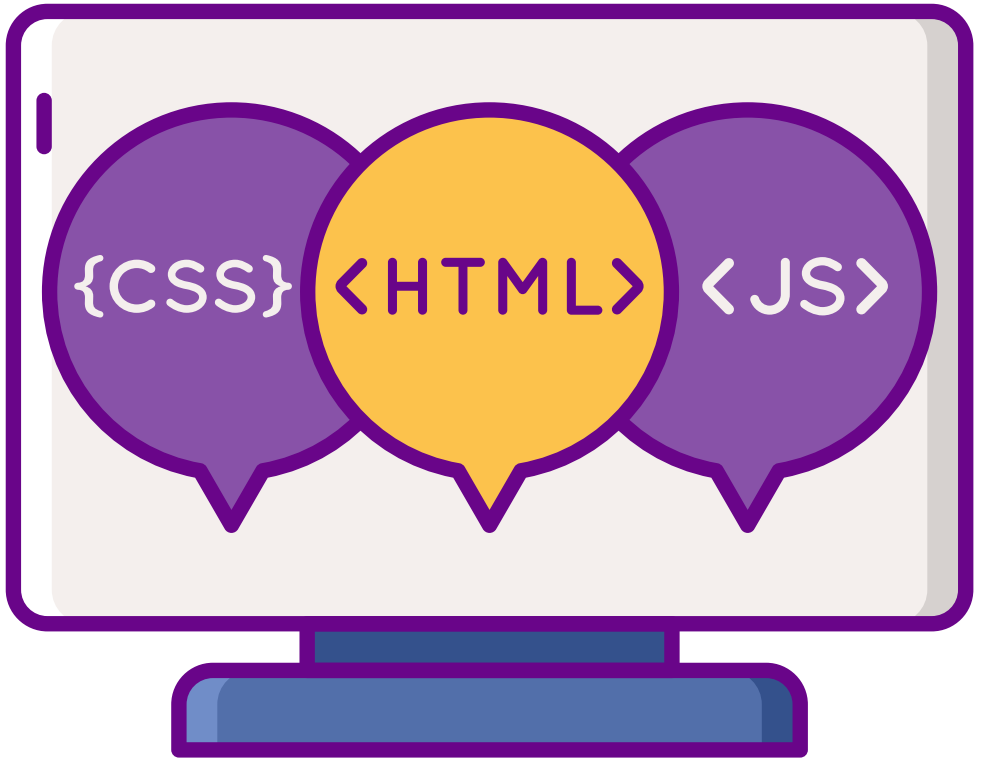
Il codice che serve a gestire una vending machine deve tener conto di diversi parametri che in un primo momento potrebbero non essere evidenti.
La moneta inserita è sufficiente a comprare la bevanda selezionata? Anche qualora fosse sufficiente, **la bevanda scelta è disponibile**? Il latte, ad esempio, potrebbe essere esaurito rendendo impossibile preparare il cappuccino

Cos'è la programmazione per il web?

Il web è fatto di pagine interattive accessibili tramite browser.

I tre pilastri del web

1. **HTML**: Struttura delle pagine.
2. **CSS**: Stile e layout.
3. **JavaScript**: Interattività e logica.



Frontend e Backend: le 2 facce del WEB

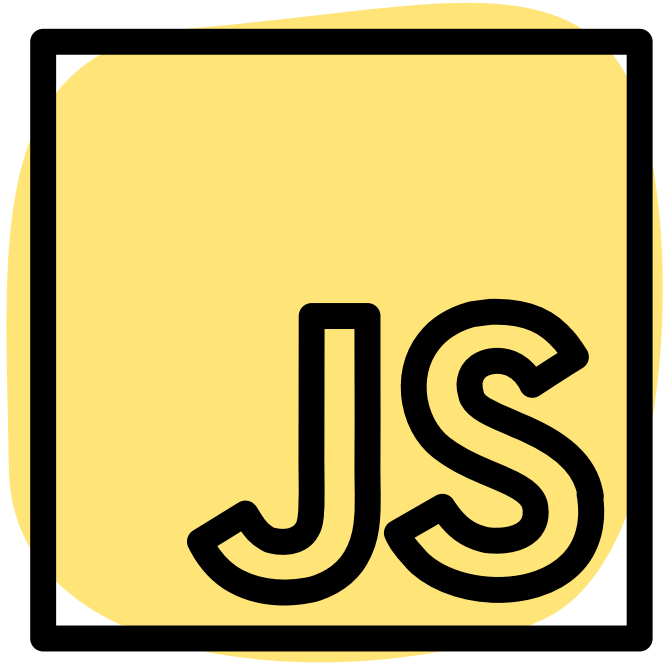
Frontend: cosa vedono e con cui interagiscono gli utenti.
I linguaggi sono HTML, CSS, JavaScript coadiuvati da librerie e framework come Angular o React.js

Backend: cosa succede dietro le quinte (database, logica di business).
Accanto a database come MySQL e Oracle i linguaggi utilizzati sono Java, Python, PHP, Node.js, C# e altri. Alcuni tra i framework più utilizzati sono Spring, Laravel, .NET, Django e altri.

JavaScript, il linguaggio Web

Non esiste un linguaggio di programmazione per antonomasia, tuttavia in questo modulo abbiamo la necessità di fare alcuni esempi e la nostra scelta ricade su JavaScript, il linguaggio Web più utilizzato.

JavaScript è il linguaggio dei Browser e ha lo scopo di aggiungere interattività alle interfacce dei siti web. Grazie a JavaScript possiamo amplificare le potenzialità dei tag HTML.



Un esempio di codice in 3 step

Primo step: scriviamo il codice HTML

```html
<h1>Benvenuto nel mio sito</h1>
<p>Io sono il primo paragrafo del paragrafo.</p>
<button id="cambia">Cambia colore</button>
```

Secondo step: scriviamo il codice CSS

```css
body{
  background:#eee;
  color:#333;
}
```

Un esempio di codice in 3 step

Terzo step: scriviamo il codice JavaScript

```
document.getElementById('cambia').addEventListener('click', function() {
    body.style.cssText="background:#333;color:#eee";
});
```

In questo esempio, al click sul TAG <button> presente nel codice HTML (**input**), vedremo modificarsi lo stile della pagina (**output**).
La **logica**: al “click” il browser ottiene le informazioni necessarie a modificare il tag body. In particolare il colore di sfondo diventerà di un grigio scuro e il testo di un grigio chiarissimo .

Introduzione allo pseudocodice

Descrizione

Lo **pseudocodice** è una modalità semplice per descrivere un algoritmo **usando frasi e parole comprensibili**, senza seguire le regole rigide di un linguaggio di programmazione. Serve a pianificare il funzionamento di un programma prima di scrivere il codice vero e proprio.

Vantaggi

Facile da capire per chiunque.

Permette di concentrarsi sulla logica piuttosto che sulla sintassi.

Come scrivere lo pseudocodice

Descrizione:

Usa un linguaggio semplice e descrittivo.

Segui una struttura logica chiara:

- **Inizio**: indica dove comincia l'algoritmo.
- **Passaggi**: descrivi i passaggi in sequenza.
- **Condizioni**: usa "Se", "Allora", "Altrimenti".
- **Cicli**: usa "Ripeti" o "Per ogni" per le ripetizioni.
- **Fine**: indica dove termina l'algoritmo.

Algoritmo per fare una spremuta d'arancia (con psudocodice)

```
Inizio
  Prendi un bicchiere
  Prendi alcune arance
  Per ogni arancia:
    Taglia l'arancia a metà
    Spremi l'arancia
    Versa il succo nel bicchiere
  Se il bicchiere è pieno:
    Ferma il processo
  Mescola il succo
  Servi la spremuta
Fine
```

Da Pseudocodice a Codice

Pseudocodice

```
Inizio
  Chiedi all'utente la sua età
  Se l'età è maggiore o uguale a 18:
    Mostra "Puoi entrare"
  Altrimenti:
    Mostra "Accesso negato"
Fine
```

Codice

```
let eta = prompt("Inserisci la tua età:");

if (eta >= 18) {
  console.log("Puoi entrare");
} else {
  console.log("Accesso negato");
}
```

Pseudocodice: fornisce la logica in linguaggio naturale per determinare se una persona può entrare in base all'età

Codice: implementa la stessa logica ma con JavaScript, utilizzando un prompt per ottenere l'età e una condizione if-else per determinare l'output.

I blocchi fondamentali della programmazione

Le variabili

Una variabile è un **contenitore che memorizza un valore.**

Ad esempio in una variabile possiamo memorizzare un nome o un numero, che può essere utilizzato e modificato durante l'esecuzione di un programma.

```
Inizio
  Dichiara variabile Nome
  Assegna a Nome il valore "Mario"
  Mostra "Ciao, " + Nome

  Dichiara variabile Età
  Assegna a Età il valore 25
  Mostra "Hai " + Età + " anni"

  Cambia il valore di Età a 26
  Mostra "Buon compleanno! Ora hai " + Età + " anni"
Fine
```

Le variabili

```
Inizio
  Dichiara variabile Nome
  Assegna a Nome il valore "Mario"
  Mostra "Ciao, " + Nome

  Dichiara variabile Età
  Assegna a Età il valore 25
  Mostra "Hai " + Età + " anni"

  Cambia il valore di Età a 26
  Mostra "Buon compleanno! Ora hai " + Età + " anni"
Fine
```

Spiegazione dello pseudocodice qui accanto in 4 passi

1. **Dichiarazione** di due variabili: Nome ed Età.
2. **Assegnazione** di un valore
3. **Utilizzo** della variabile
4. **Modifica** della variabile: il valore di Età cambia, da 25 a 26.

Le variabili e i tipi di dato

Le variabili possono contenere testi, numeri, ma anche altre tipologie di dati.
Di seguito **i tipi di dato più utilizzati in JavaScript**:

Numerici

- **Interi:** 2, -10, 28 o **Decimali**: 3.14, -0.5, 2.718
- **Valori speciali**: Infinity, -Infinity e NaN (Not-a-Number)

Stringhe: il termine che in programmazione identifica i testi, più in generale una sequenza di caratteri. Le stringhe si identificano con l'uso delle virgolette: *"Mario"*, *"Ciao, mi chiamo Anna e ho 25 anni!"*

Booleani: possono avere solo due valori, **true** o **false**. Si utilizzano per le condizioni e la logica.

Undefined: identifica una variabile dichiarata ma a cui non è stato assegnato alcun valore.

Null: rappresenta l'assenza intenzionale di un valore.

Perché sono importanti i tipi di dato

I tipi di dato in un linguaggio di programmazione sono come **contenitori speciali** che servono per tenere diversi tipi di cose, proprio come nella vita reale usiamo diversi contenitori per diversi oggetti o scopi.

Un blocco di appunti o un quaderno servono a contenere dei testi così come una variabile per una stringa può contenere il tuo nome

Il sacchetto dei numeri della tombola può rappresentare una variabile che contenga un numero

L'interruttore della luce, con le sue 2 uniche posizioni può rappresentare una variabile che contenga un valore booleano (true o false)

Un scatola vuota può identificare una variabile che contenga un valore di tipo null

Perché sono importanti i tipi di dato

Aiutano il computer a capire cosa fare: il computer deve sapere che tipo di contenuto sta maneggiando per trattarlo correttamente.

Esempio: non puoi sommare una parola come "Mario" con un numero come 5. Il computer, come del resto anche *noi*, non sapremmo dare una risposta valida.
Il computer deve sempre essere in grado di distinguere tipi di dato.

Ti permettono di fare le cose giuste: se vuoi fare una somma, servono due o più numeri.
Se vuoi scrivere un messaggio, serve usare del testo.

Evitano confusione: proprio come nella vita reale non puoi mettere un bussolotto della tombola in un quaderno e non puoi usare l'interruttore della luce per prendere appunti :)

Un esempio con i tipi di dato

Programmiamo un robot che deve scrivere un biglietto di compleanno, questi gli step per scrivere il codice con i tipi di dato da utilizzare:

- Nome della festeggiata: è una stringa ("Michela").
- Anni della festeggiata: è un numero (29).
- È il suo compleanno oggi? È una risposta sì o no, quindi è un booleano (true o false).

```
Inizio
  Dichiara variabili Nome, Età, Compleanno
  Assegna a Nome il valore "Michela"
  Assegna a Età il valore 29
  Assegna a Compleanno il valore true

  Se (Compleanno è true)
    Mostra "Buon compleanno " + Nome +" per i tuoi " + Età + " anni"
Fine
```

Un altro esempio con i tipi di dato

Nel mio esempio definisco due variabili **x** e **y** e assegno loro due valori in questo modo

x=10 (valore di tipo numerico) **y="20"** (valore di tipo stringa)

Con JavaScript adesso provo a sommare il valore delle due variabili **risultato=x+y**
Per via dei valori non entrambi numerici, scoprirei che il valore assegnato a risultato è "**1020".** L'output produce una concatenazione, un'unione, tra il numero 10 e la stringa "20" producendo un'ulteriore stringa composta dal 10 che si unisce al testo "20"

Diverso sarebbe il caso in cui assegnassi alle due variabili i valori in questo modo

x=10 (valore di tipo numerico) **y=20** (valore di tipo numerico)

In questo caso l'output produrrebbe il valore di tipo numerico **30**

Introduzione alle decisioni e ai cicli

Le decisioni (if-else)

Quando programmiamo spesso dobbiamo dire al computer cosa fare in base a certe condizioni. Questo si chiama **decisione**.

Immagina di essere davanti alla macchinetta delle bevande vista prima: se hai abbastanza soldi puoi prendere un caffè, altrimenti non puoi.

In programmazione, questo si scrive con if-else, che significa "Se succede una cosa, fai questo; altrimenti, fai qualcos'altro".

Ad esempio, se fuori fa freddo (if temperatura < 10), possiamo dire al computer di mostrare "Metti il cappotto!", altrimenti (else), "Puoi uscire senza giacca".

Introduzione alle decisioni e ai cicli

I cicli (for)

A volte, invece di prendere una decisione una sola volta, vogliamo ripetere un'azione più volte, ed è qui che entrano in gioco i cicli.

Un ciclo serve per far fare al computer la stessa cosa più volte senza dover scrivere lo stesso comando molte volte.

Ad esempio, se dobbiamo contare da 1 a 10, possiamo usare un ciclo for che dice al computer di ripetere un'azione finché arriva a 10.

È come quando devi mettere in fila 10 sedie: invece di dire "Metti una sedia" dieci volte, dici "Ripeti questa azione 10 volte".

Esempi sulle decisioni e i cicli

If/Else - Controllo dell'Età

Un semplice controllo per verificare se un utente è maggiorenne.

```
Inizio
  Dichiara variabile Età
  Assegna a Età il valore 17

  Se (Età >= 18):
      Mostra "Puoi entrare"
  Altrimenti:
      Mostra "Accesso negato"
Fine
```

Esempi sulle decisioni e i cicli

If/Else Annidati - Controllo della Temperatura

Un controllo più complesso per verificare più condizioni.

```
Inizio
  Dichiara variabile Temperatura
  Assegna a Temperatura il valore 5

  Se (Temperatura > 30):
      Mostra "Fa molto caldo!"
  Altrimenti se (Temperatura >= 15 E Temperatura <= 30):
      Mostra "Il clima è piacevole"
  Altrimenti se (Temperatura >= 0 E Temperatura < 15):
      Mostra "Fa freddo"
  Altrimenti:
      Mostra "Fa molto freddo!"
Fine
```

Esempi sulle decisioni e i cicli

Ciclo For - Contiamo le fette di pizza che mangia Gaia

Lasciamo che Gaia mangi una fetta di pizza per volta, finché non finiscano.

```
Inizio
  Dichiara variabile FettePizza
  Assegna a FettePizza il valore 8
  Dichiara Nome
  Assegna a Nome il valore "Gaia"

  Per ogni Fetta da 1 a FettePizza:
      Mostra Nome + " sta mangiando la fetta numero " + Fetta
      Se (Fetta == 4):
          Mostra Nome + " è a metà della pizza!"
      Se (Fetta == FettePizza):
          Mostra Nome + " ha finito la pizza! 😋"
Fine
```

Esempi sulle decisioni e i cicli

Spiegazione dell'esercizio precedente:

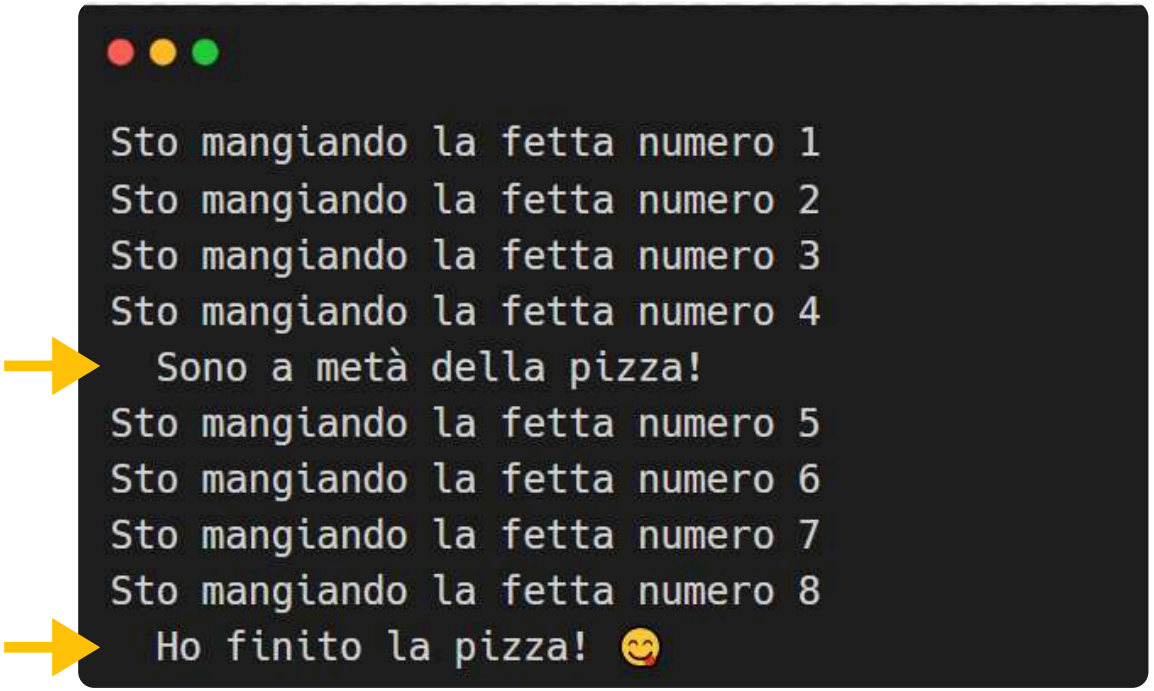
Il ciclo for conta ogni fetta di pizza mentre viene mangiata.

Quando arriva alla quarta fetta, mostra un messaggio speciale.

Alla fine, quando l'ultima fetta viene mangiata, appare un messaggio che dice che la pizza è finita!

Qui accanto è rappresentato l'output dell'esercizio

Le 2 frecce evidenziano le due stringhe prodotte come conseguenza delle decisioni (if/se)



Introduzione alle funzioni

Cos’è una funzione?

Descrizione:

- Una funzione è un blocco di codice che esegue un'operazione specifica.
- Serve per organizzare il codice e riutilizzarlo più volte senza ripeterlo.

Vantaggi:

- Rende il codice più ordinato e leggibile.
- Evita ripetizioni inutili.
- Facilita la manutenzione e le modifiche

Se torniamo all’esempio della *vending machine*, possiamo scomporre l’intero algoritmo in una serie di funzioni, in cui ognuna delle quali si occupa di una operazione specifica (una funzione che controlli la moneta inserita, una la scelta della bevanda, una la quantità di zucchero, ecc...)

Introduzione alle funzioni

Quali Funzioni servono per una Vending Machine?

Per far funzionare una vending machine, scriviamo diverse funzioni che gestiscono ogni fase dell'operazione e che, insieme, contribuiscono alla realizzazione del software.

Una lista di funzioni utili alla realizzazione del codice:

- **VerificaCredito():** Controlla se l'utente ha inserito abbastanza denaro.
- **SelezionaProdotto(codice)**: Legge il codice della bevanda scelta.
- **VerificaDisponibilità(prodotto)**: Controlla se la bevanda è disponibile.
- **ErogaProdotto(prodotto)**: Attiva il sistema per preparare e servire la bevanda.
- **RilasciaResto(creditoInserito, prezzoProdotto)**: Calcola e restituisce il resto.
- **MostraMessaggio(messaggio)**: Mostra informazioni all'utente sul display.

Come si scrive una Funzione?

Una funzione ha tre parti principali:

- **Dichiarazione**: il nome e i parametri della funzione.
- **Corpo**: il codice che esegue l'operazione.
- **Valore Restituito** (opzionale): il risultato della funzione.

```
Inizio Funzione Somma(a, b)
    Risultato = a + b
    Restituisci Risultato
Fine Funzione
```

In questo esempio

1. si dichiara un funzione dal nome somma che prevede due parametri (a,b).
2. Nel corpo della funzione si esegue la somma aritmetica dei 2 parametri e il risultato viene salvato in una variabile.
3. Il valore restituito è quello della variabile “Risultato”

Come si usa una Funzione?

Una funzione non viene eseguita automaticamente ma **deve essere "chiamata"**. Quando la chiamiamo, possiamo passare dei valori (argomenti) che saranno elaborati.

```
Variabile risultato = Somma(5, 3)
Mostra risultato  // Output: 8
```

Funzioni: parametri e valori di ritorno

Come le funzioni scambiano dati

- Le funzioni possono ricevere parametri (dati in ingresso) per personalizzare il loro comportamento.
- Possono restituire un valore in uscita, che possiamo salvare o usare direttamente.

```
Inizio Funzione Moltiplica(x, y)
    Restituisci x * y
Fine Funzione

Variabile risultato = Moltiplica(4, 2)
Mostra risultato // Output: 8
```

Funzioni con e senza valore di ritorno

1. **Funzioni senza valore di ritorno**: eseguono un'azione ma non restituiscono nulla.

```
Inizio Funzione Saluta()
    Mostra "Ciao!"
Fine Funzione
```

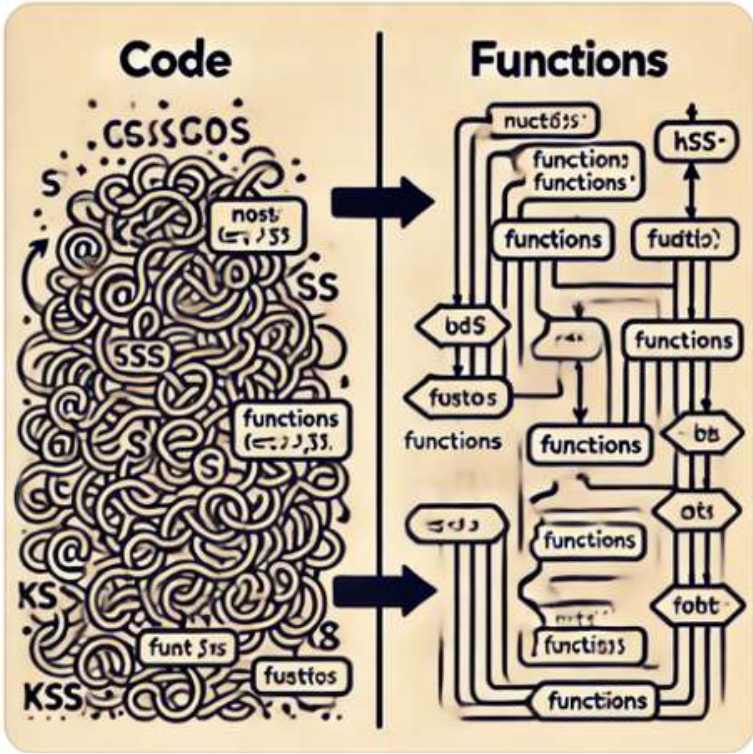
2. **Funzioni con valore di ritorno**: calcolano e restituiscono un valore

```
Inizio Funzione CalcolaQuadrato(n)
    Restituisci n * n
Fine Funzione
```

Benefici dell'uso delle funzioni

Perché le funzioni sono utili?

- Permettono di riutilizzare codice senza riscriverlo.
- Rendono il codice più modulare e organizzato.
- Facilitano la risoluzione dei problemi e la collaborazione in team.



JavaScript, il linguaggio del web

JavaScript nei Siti Web

- JavaScript è il linguaggio che permette di rendere interattivi i siti web.
- Viene usato per modificare contenuti, rispondere a eventi e creare animazioni.

Come aggiungere JavaScript a una pagina HTML:

- **Metodo interno**: inserire codice JavaScript all'interno di **<script>** in un file HTML.
- **Metodo esterno**: collegare un **file .js** con **<script src="script.js"></script>**.

```
<script>
    console.log("Hello world");
</script>
```

```
<script src="path/file.js">
</script>
```

Manipolare l'HTML con il DOM

JavaScript è in grado di "manipolare" i tag HTML: grazie al DOM è possibile individuare qualsiasi tag della pagina HTML e modificarne qualunque contenuto o attributo.

Il **DOM (Document Object Model)** è lo *strumento* che permette di accedere e modificare gli elementi HTML con JavaScript.

Possiamo, ad esempio, usare **document.getElementById("idElemento")** per selezionare un tag e modificarne il contenuto

```
document.getElementById("titolo").innerText = "Nuovo testo!";
```

Manipolare l'HTML con il DOM

Grazie al DOM, JavaScript **può rilevare e rispondere agli eventi** generati dall'utente.

Un evento è qualsiasi azione compiuta sulla pagina, come:

- il click su un pulsante
- lo scroll della pagina
- la pressione di un tasto sulla tastiera

```
document.getElementById("mioBottone").addEventListener("click",
function() {
    alert("Hai cliccato il bottone!");
});
```

Esercizio pratico

Cambiare il colore della pagina grazie a un pulsante interattivo

Creiamo un pulsante che cambia il colore dello sfondo della pagina quando viene cliccato.

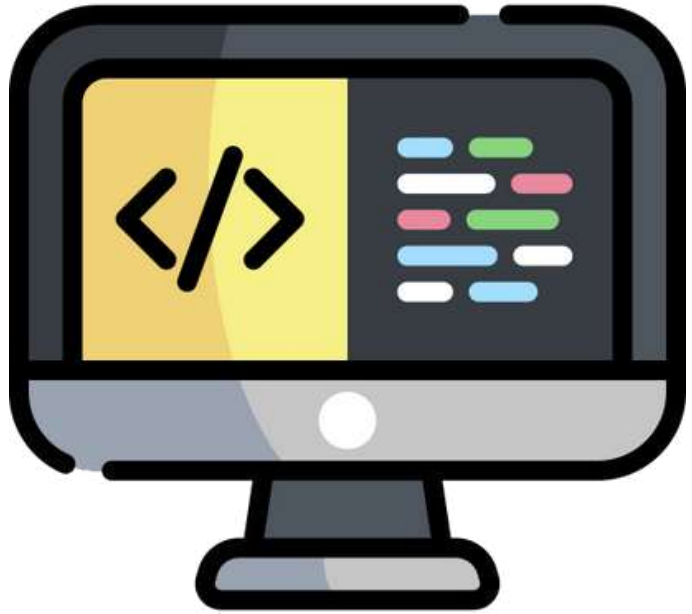
```html
<button onclick="cambiaColore()">Cambia Colore</button>
<script>
  function cambiaColore() {
    document.body.style.backgroundColor = "lightblue";
  }
</script>
```

Conclusioni e Riepilogo

Cosa Abbiamo Imparato?

Durante questo percorso abbiamo trattato i concetti fondamentali della programmazione:

- Cosa significa programmare?
- I blocchi fondamentali della programmazione.
- Introduzione ai cicli e alle decisioni.
- Funzioni e organizzazione del codice.
- Introduzione e JavaScript: manipolare l'HTML con il DOM




Labforweb
nerd academy