Linguaggi di programmazione

Un linguaggio di programmazione è un insieme di parole e simboli che servono a scrivere le istruzioni per un computer. Spesso si arriva a un compromesso tra la semplicità d'uso del linguaggio e la sua efficacia.

Linguaggi ad alto e a basso livello

I linguaggi di programmazione ad alto livello sono studiati per chi non ha una comprensione approfondita dell'hardware. Spesso usano parole simili al linguaggio umano e gestiscono automaticamente alcuni aspetti del computer; in genere, questi programmi funzionano su vari hardware. I linguaggi a basso livello, al contrario, offrono ai programmatori un controllo granulare sul computer, ma richiedono conoscenze minuziose del suo funzionamento; i programmi scritti in linguaggi a basso livello possono non essere compatibili con hardware diversi.

LINGUAGGI AD ALTO LIVELLO
- Relativa rapidità di scrittura
- Relativa facilità di comprensione
- Spesso la velocità di esecuzione è abbastanza alta
- Possono essere usati su hardware diversi
- Non richiedono la conoscenza dell'hardware

Istruzione per visualizzare un testo

```
print("Hello!")
```

Python
Linguaggio ad alto livello molto diffuso, Python è facile da leggere e da scrivere. L'istruzione visualizza il messaggio "Hello!" sullo schermo.

LINGUAGGI A BASSO LIVELLO
- Controllo diretto delle funzioni hardware
- Possono aumentare la velocità di codici performance-sensitive
- Richiedono una comprensione dell'hardware
- I programmi girano solo su processori simili

È una notazione esadecimale, un sistema numerico molto usato in informatica

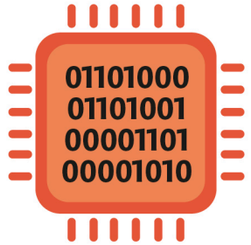
```
MOV AX, 66H
```

Linguaggio assembly
Riflette più fedelmente il codice macchina del processore. L'esempio sceglie un numero e lo inserisce nella parte del processore detta accumulatore.

Codice macchina

Il codice a basso livello che rappresenta il modo in cui vengono impartiti i comandi all'hardware e al processore è detto codice macchina. Si tratta di un insieme di cifre binarie (composte da 1 e 0), lette e interpretate dal processore. Le istruzioni in codice macchina sono formate da un opcode (codice operativo) e da uno o più operandi: il primo dice al computer cosa fare, i secondi quali dati usare.

Microprocessore
Il microprocessore è il "cervello" del computer e controlla la maggior parte delle operazioni. Esegue i comandi e segue le istruzioni del codice.

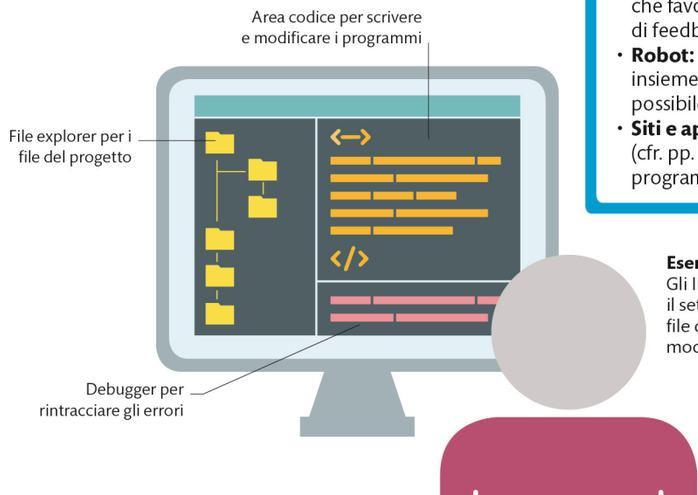


Come il computer comprende il linguaggio di programmazione

Alla fine, tutti i programmi sono ridotti al codice macchina. Molti, pur utilizzando linguaggi facili da capire, per poter essere compresi dal processore devono essere tradotti in termini di bit. L'interprete trasforma ed esegue le istruzioni durante il funzionamento del programma, mentre il compilatore traduce il programma in una sola volta, prima dell'avvio.

Usare un ambiente di sviluppo integrato (IDE)

Un IDE è un insieme di strumenti a disposizione dei programmatori. Include un editor di codice per la scrittura dei programmi ed è spesso dotato di funzioni che aumentano la produttività, come l'autocompletamento per le istruzioni e un codice cromatico per favorire la leggibilità. Alcuni prevedono un debugger per rintracciare gli errori e un compilatore o un interprete per testare ed eseguire i programmi.



Esempio di configurazione di un IDE
Gli IDE spesso consentono agli utenti di configurare il set-up. Nell'esempio, il programmatore sfoglia i file del progetto sulla sinistra, inserisce il codice e lo modifica sulla destra, ed esegue il debug in fondo.

APPLICAZIONI

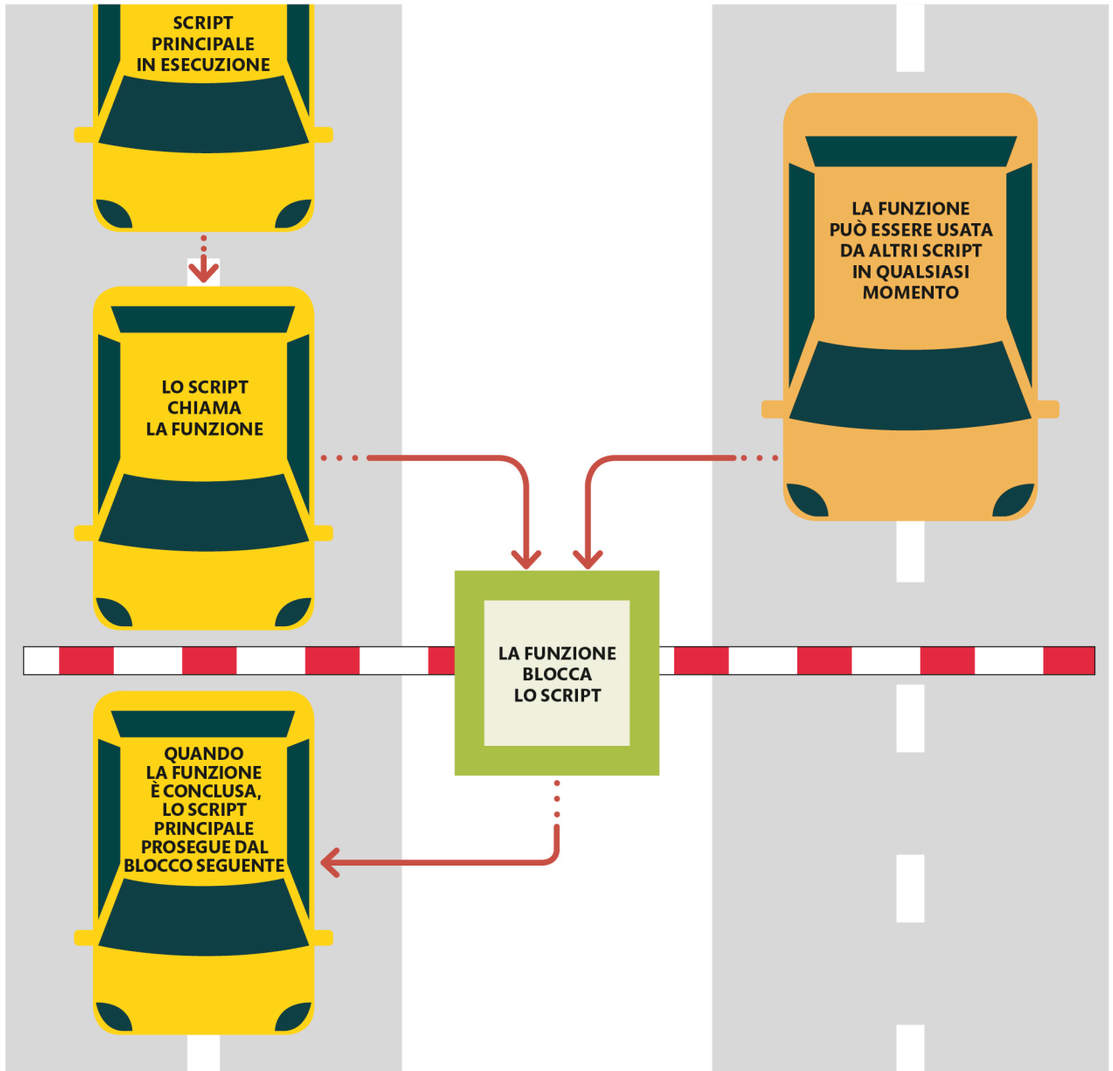
Dopo aver imparato a programmare, le competenze elencate possono essere usate per un ampio ventaglio di progetti utili e creativi.
- **Domotica:** l'applicazione all'ambiente domestico, per esempio per gestire luci o tende
- **Videogiochi:** i videogiochi sono un modo fantastico per sperimentare con il coding, che favorisce la condivisione e il ricevimento di feedback (cfr. pp. 80-91, 178-203)
- **Robot:** usando schede Arduino o Raspberry Pi insieme a kit o componenti elettronici, è possibile programmare il proprio robot
- **Siti e app Web:** usando HTML, CSS e JavaScript (cfr. pp. 210-343) possono essere creati programmi che girano su ogni Web browser

Usare le funzioni

Le funzioni sono parti di un programma che possono essere riutilizzate, svolgono compiti determinati e semplificano la lettura, la scrittura e la verifica del codice. In Scratch, ogni blocco è una funzione e gli utenti possono crearne di nuovi.

Come procede il programma

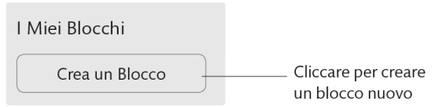
Quando viene eseguito uno script, Scratch attiva un blocco di istruzioni alla volta, procedendo dall'alto verso il basso. Se si tratta di una funzione, ricorda la sua posizione nello script e passa a eseguirne le istruzioni. Una volta concluso il processo, riprende lo script principale da dove lo aveva interrotto. Le funzioni possono essere usate da vari script e consentono di processare informazioni.



Definire i propri blocchi

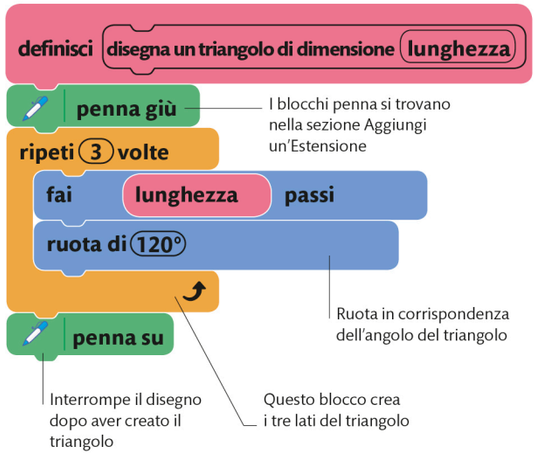
Per evitare di ripetere parti di codice più volte, Scratch consente agli utenti di creare i propri blocchi, ognuno dei quali può essere composto da varie istruzioni. L'esempio in basso illustra come creare una funzione per tracciare un triangolo, poi la usa per disegnare triangoli di tre dimensioni diverse uno sopra l'altro. Il risultato ha l'aspetto di un abete.

1 Creare un nuovo blocco
Selezionate il pulsante Crea un Blocco nella sezione I Miei Blocchi. Nominate il blocco "disegna un triangolo di dimensione".

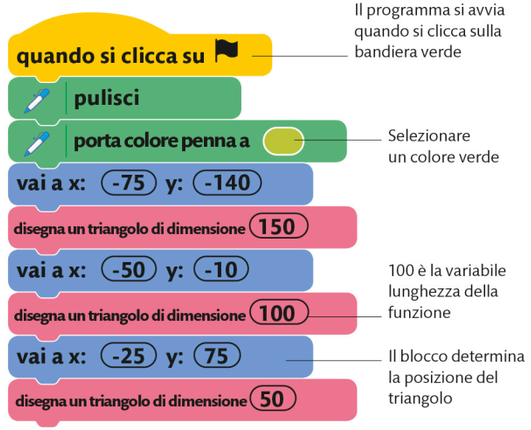


2 Aggiungere e denominare un input
Selezionate l'opzione per aggiungere un argomento. Comparirà un'altra area di input in cui digitare "lunghezza". Cliccate ora su OK.

3 Definire lo script
Aggiungete le seguenti istruzioni al blocco **definisci**. Il valore assegnato alla funzione si colloca nel blocco **lunghezza**. Per usarla nello script, trascinate la variabile dal blocco **definisci** al blocco **fai 10 passi**.



4 Usare il blocco nuovo
Ora è possibile usare il nuovo blocco in un programma. Aggiungete lo script nell'area Codice e, dopo aver cliccato sulla bandiera verde, Scratch disegnerà tre triangoli.



PERCHÉ USARE LE FUNZIONI?

Quasi tutti i linguaggi di programmazione fanno uso di funzioni. Ecco alcuni vantaggi:

- Una volta scritta, una funzione può essere riutilizzata in altri programmi.
- Se ogni funzione ha un nome chiaro, i programmi sono più facili da leggere.
- Riutilizzando le funzioni, i programmi sono più brevi.
- È più facile scrivere e testare molte piccole funzioni che un grande programma.

L'esempio riportato sopra sarebbe più lungo, complesso e difficile da capire se ogni occorrenza di "disegna un triangolo di dimensione" fosse sostituita da tutti i blocchi che compongono la funzione.

Messaggi di errore

Benché siano lo strumento più immediato per aiutare i programmatori con il debugging, quelli di Python tendono a essere piuttosto criptici e possono dare l'impressione di aumentare il mistero invece che chiarirlo. La tabella elenca alcuni dei messaggi più comuni con il loro significato. In genere, i programmatori imparano presto a familiarizzare con gli errori e a trovare la soluzione.

MESSAGGI DI ERRORE

Messaggio di errore	Significato
EOL found while scanning string literal	Apice di chiusura mancante per una stringa nella riga
unsupported operand type(s) for +: 'int' and 'str'	L'operatore + richiede che i valori su entrambi i lati siano dello stesso tipo
Expected an indented block	Il corpo di un ciclo o una condizione non è indentato
Unexpected indent	L'indentazione della riga è eccessiva
Unexpected EOF while parsing	Parentesi mancante poco prima della fine del programma
Name [name of variable or function] is not defined	In genere causato da un errore nel nome di una variabile o di una funzione

TROVARE L'ERRORE

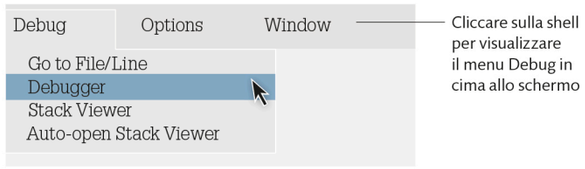
Un problema comune del debugging è causato dal fatto che l'errore può trovarsi appena prima del luogo indicato dal messaggio. Cercate l'errore prima del luogo indicato, nella stessa riga o in quella precedente.

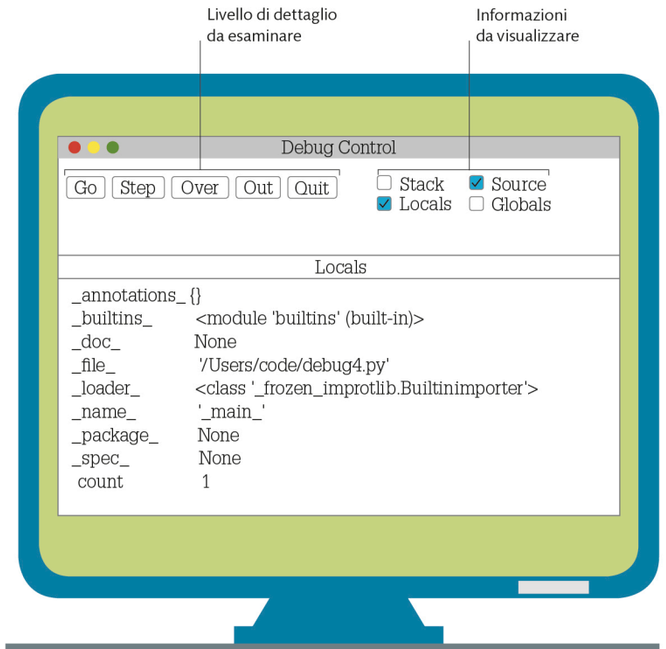
```
print(hello " + "world")
```

La mancanza dell'apice di apertura genera l'errore "Invalid syntax"

Debugger

IDLE include inoltre uno strumento chiamato debugger, che consente ai programmatori di passare in rassegna il codice, eseguendo riga dopo riga e mostrando il contenuto delle variabili per ogni passaggio del programma. Il debugger si avvia dalla shell, che include un menu Debug. Selezionando Debugger, si avvia il processo non appena si lancia il programma. Selezionandolo una seconda volta, viene disattivato.





Debugger di IDLE
Quando si esegue un programma, il debugger visualizza le relative informazioni, inclusi i valori correnti delle variabili. Cliccando sull'opzione "Step", si visualizzerà il codice eseguito "dietro le quinte", in genere nascosto.

Colore del testo

Come la maggior parte degli IDE (cfr. pp. 208-209) e dei programmi dedicati per la modifica del codice, IDLE impiega colori diversi, facilitando l'identificazione degli errori. Per esempio, parole chiave come "for", "while" e "if" sono arancio, mentre le stringhe sono verdi (cfr. p. 98). Se una parte del codice non appare del colore giusto, potrebbe indicare un errore di sintassi, mentre varie righe di codice in verde è in genere un segno che manca l'apice di chiusura di una stringa.

Colore sbagliato
Nell'esempio sono presenti quattro errori. Gli apici mancanti e l'errore nella digitazione di "while" contrassegneranno il codice con il colore sbagliato.

Apice di apertura mancante

La parola chiave "while" non è digitata correttamente

Errore di battitura e apice mancante

```
answer = input(Scegli un numero")
whle answer != 7:
    pritn(Non è il numero giusto")
    answer = input("Scegli un numero")
```

Checklist per il debugging

Quando compare un errore ma non capite perché, potete controllare vari aspetti. Pur non risolvendo tutti i problemi, molti errori sono generati da sviste banali e facili da correggere. Ecco una lista di possibili cause:

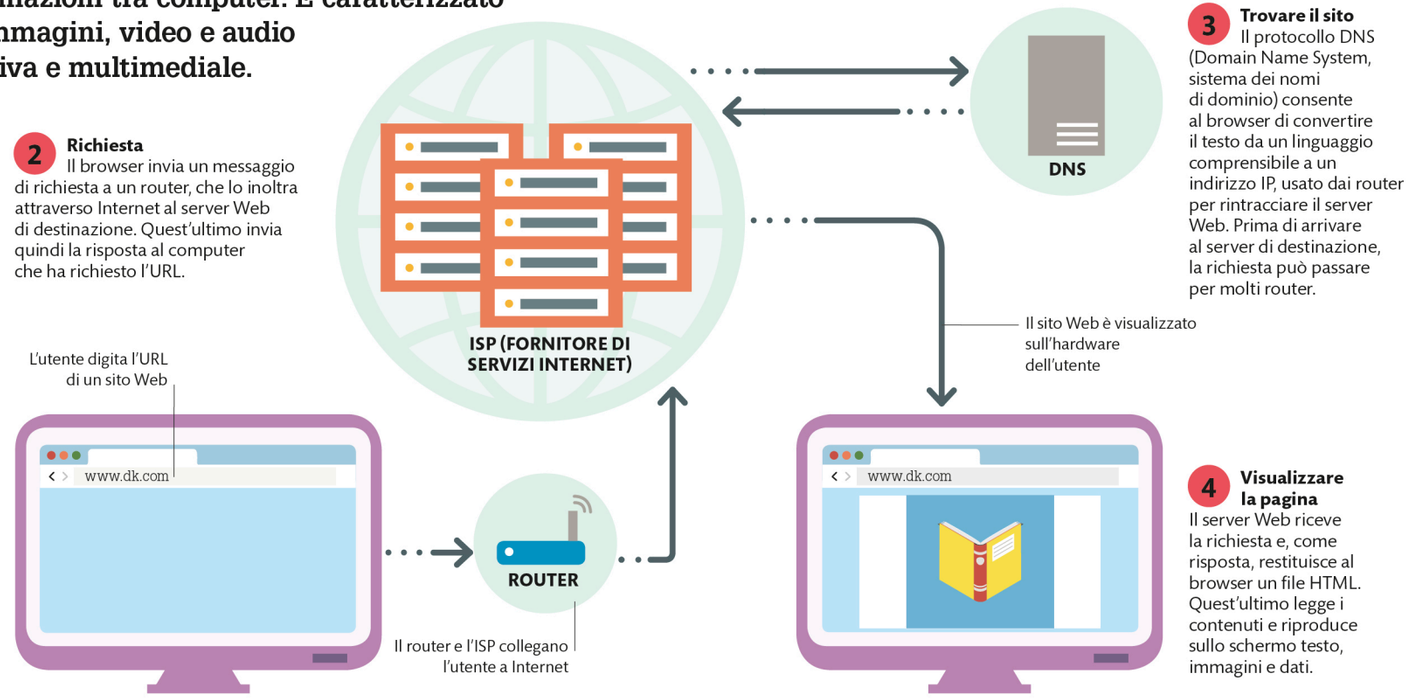
- [x] L'ortografia è corretta
- [x] Uso corretto di maiuscole e minuscole
- [ ] Le stringhe hanno apici all'inizio e alla fine
- [ ] Le parentesi sono aperte e chiuse
- [ ] Il codice è stato salvato dopo l'ultima modifica
- [ ] Non sono stati confusi numeri e lettere, per es. O e 0, l e 1
- [ ] Non sono presenti spazi inutili all'inizio di una riga
- [ ] Variabili e funzioni sono dichiarate prima dell'uso

Come funziona il Web

Il World Wide Web è un insieme di tecnologie che, attraverso Internet, consente la condivisione di informazioni tra computer. È caratterizzato da una combinazione di testo, immagini, video e audio per offrire un'esperienza interattiva e multimediale.

Connettersi a un sito Web

Il Web si basa su un modello client/server. Il browser è il client che richiede una pagina Web a un server, il quale risponde inviando un file HTML. Il contenuto di ogni richiesta è determinato dal protocollo di comunicazione adottato. Hypertext Transfer Protocol (HTTP) è il più usato su Internet, una rete globale che connette miliardi di dispositivi.



1 Inserire l'URL di una pagina Web
Il processo inizia quando un utente inserisce un URL (Uniform Resource Locator) nella barra degli indirizzi di un browser Internet. L'URL contiene l'indirizzo della pagina richiesta e può essere usato per localizzare il server Web che ospita il sito.

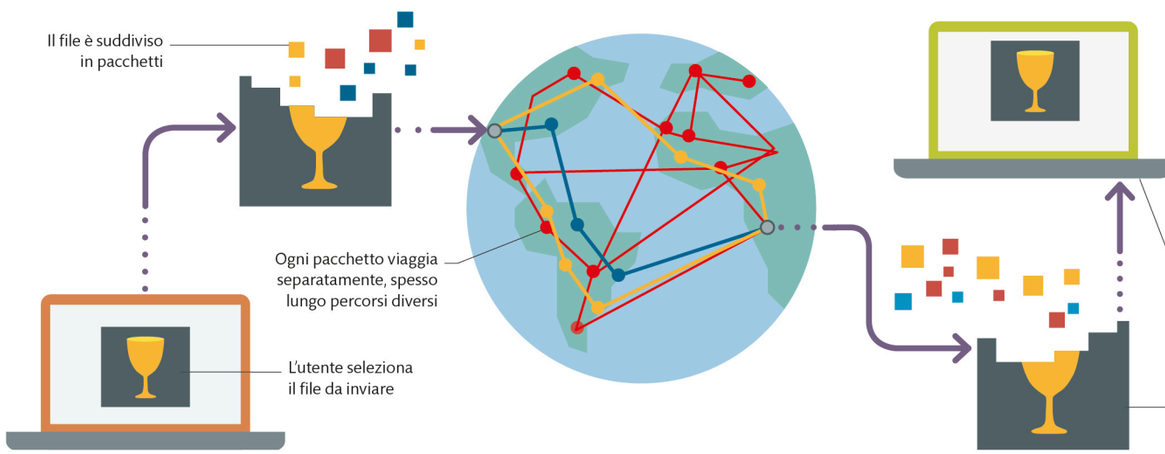
2 Richiesta
Il browser invia un messaggio di richiesta a un router, che lo inoltra attraverso Internet al server Web di destinazione. Quest'ultimo invia quindi la risposta al computer che ha richiesto l'URL.

3 Trovare il sito
Il protocollo DNS (Domain Name System, sistema dei nomi di dominio) consente al browser di convertire il testo da un linguaggio comprensibile a un indirizzo IP, usato dai router per rintracciare il server Web. Prima di arrivare al server di destinazione, la richiesta può passare per molti router.

4 Visualizzare la pagina
Il server Web riceve la richiesta e, come risposta, restituisce al browser un file HTML. Quest'ultimo legge i contenuti e riproduce sullo schermo testo, immagini e dati.

Pacchetti e routing IP

Tutta la comunicazione sul Web avviene dividendo la richiesta in piccoli segmenti di dati, detti pacchetti, che vengono "instradati" dalla fonte alla destinazione, dove sono riassemblati per comporre il messaggio originario. Le reti che trasportano i pacchetti sono dette "reti a commutazione di pacchetto". Questi sono composti da due parti: l'informazione, che indica dove e come inviare i dati, e i dati, il contenuto che si vuole recapitare.



Perché i pacchetti
Per aumentare l'efficienza dei canali di comunicazione, immagini, testo e perfino richieste HTTP basiche sono scomposti e trasferiti pezzo dopo pezzo, ognuno dei quali contiene la sequenza del pacchetto che indica al server ricevente come riassemblare l'informazione.

Protocolli

I protocolli sono insiemi di regole che governano la comunicazione tra due entità, e quelli del Web hanno la funzione di gestire la comunicazione tra il browser client e il server Web. I protocolli di rete sono strutturati come una serie di livelli, ognuno studiato per uno scopo specifico e presente sia sull'host che invia il messaggio, sia su quello che lo riceve.

Livello di applicazione
Definisce come un'applicazione formatta i dati per comunicare con altre applicazioni. Per esempio, HTTP e File Transfer Protocol (FTP) stabiliscono come un browser Web comunica con un server Web.

Livello di accesso alla rete
Avvalendosi di router per trovare il computer di destinazione e consegnare il messaggio, definisce come inviare i dati da una rete a un'altra.

Livello di trasporto
Definisce la gestione delle comunicazioni, mantenendo sessioni tra i computer di origine e di destinazione e riassemblando i pacchetti ricevuti nell'ordine corretto.

Protocolli Web
Il Transmission Control Protocol (TCP) gestisce le sessioni e l'ordinamento dei pacchetti ricevuti dal browser. L'Internet Protocol (IP) si occupa dell'instradamento dei pacchetti di dati tra client e server, mentre HTTP/FTP/UDP (User Datagram Protocol) definiscono i messaggi scambiati tra browser e server.

HTTP

HTTP è un protocollo a livello applicativo che descrive le modalità di formattazione e di invio di un messaggio di richiesta di un client e della risposta del server.

- Il metodo GET recupera i dati
- Il metodo POST li aggiorna
- Il metodo PU li crea
- Il metodo DELETE li rimuove

2.6 PAGINA HTML

Visual Studio crea il file "index.html" con il codice minimo richiesto per una pagina HTML valida. Se state usando un altro ambiente di sviluppo, digitate il codice riportato a lato nel nuovo file indice.

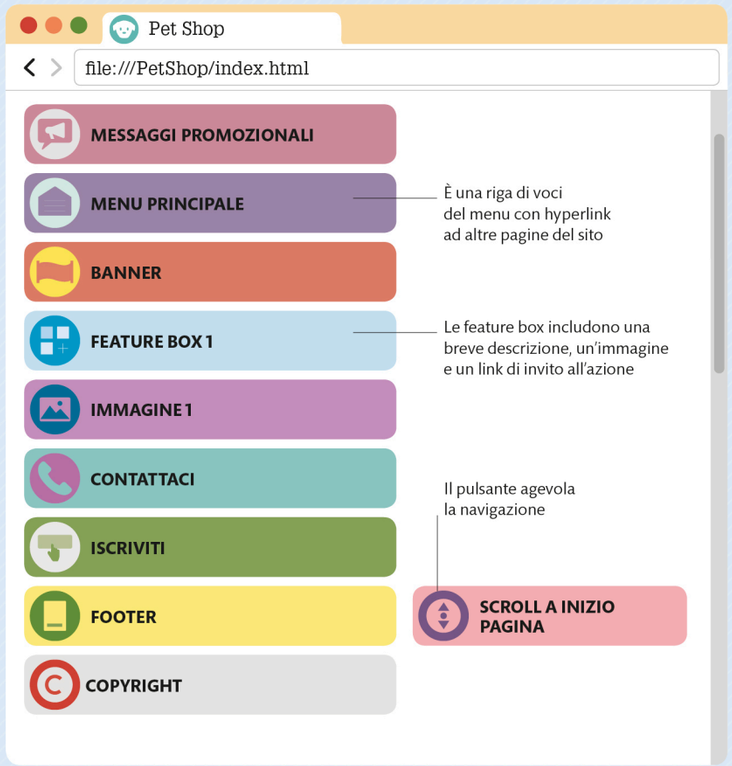
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>

</body>
</html>
```

- `<!DOCTYPE html>`: Dichiarazione del tipo di documento
- `<meta charset="utf-8" />`: L'attributo "charset" specifica la codifica dei caratteri per il documento HTML, cioè comunica al computer come interpretare i dati binari in caratteri reali
- `<body>`: Il tag contiene l'insieme di testo, dati e immagini visibili sulla pagina
- `<title></title>`: Il tag è un contenitore per il testo che comparirà nel browser come il titolo della pagina
- `</html>`: Tag esterno per il documento HTML

3 Strutturare la homepage

In HTML, una homepage è composta da una serie di livelli orizzontali collocati uno sopra l'altro. Il primo conterrà una riga animata di messaggi promozionali, seguita dal "Menu principale" e da un banner con un'immagine grande, il logo aziendale e un pulsante di invito all'azione. Compariranno poi una feature box e un'immagine di grandi dimensioni. Seguiranno altri livelli che alterneranno feature box e immagini grandi. I dati di contatto, un link per la registrazione, hyperlink e l'avviso di copyright saranno in fondo. Aggiungerete quindi un pulsante di scroll a inizio pagina per facilitare la navigazione.



3.1 AGGIUNGERE IL NOME DEL SITO

Prima di inserire nella pagina testo, immagini e dati, aggiungete il nome del sito nel tag <head> digitando la definizione del titolo di pagina nel tag <title>.

```
<head>
    <meta charset="utf-8" />
    <title>Pet Shop</title>
</head>
```

- `<head>`: Il tag <head> carica i metadati prima di visualizzare la pagina
- `<title>Pet Shop</title>`: Il testo compare nel browser come il titolo della scheda

3.2 AGGIUNGERE LA DEFINIZIONE DELLA FAVICON

Per aggiungere l'icona al sito, inserite il tag <link> sotto il tag <title>. L'attributo "href" indica il file favicon nella cartella "images" del sito.

```
<title>Pet Shop</title>
<link rel="icon" type="image/png"
href="images/favicon.png">
```

Il libro usa la freccia per suddividere il codice su due righe

FAVICON

La favicon è una piccola immagine che compare nella scheda del browser accanto al titolo della pagina. È un'icona quadrata che facilita la ricerca della scheda nel browser. Il fondo può essere a tinta unita o trasparente, e il formato è .png o .ico.

3.3 AGGIUNGERE IL TESTO

Potete ora cominciare ad aggiungere il testo, i dati e le immagini all'interno del tag <body>, in modo che siano visualizzati all'apertura del documento HTML nel browser. Per inserire i messaggi promozionali aggiungete il tag esterno , seguito dai tag figli che contengono i messaggi. Eccetto il primo, tutti i div dei messaggi devono includere un attributo "style" che comunica al browser di non visualizzarli: per ora ci limitiamo a un solo messaggio promozionale; in seguito, JavaScript visualizzerà ciclicamente i vari messaggi.

```
<body>
    <div id="promo" >
        <div>Spedizione gratuita</div>
        <div style="display:none;">Nuovi giocattoli per cuccioli</div>
        <div style="display:none;">Acquista 5 giocattoli e risparmia il 30%</div>
        <div style="display:none;">Spedizione il giorno stesso</div>
    </div>
</body>
```

- `<div id="promo" >`: Il tag contiene un gruppo di elementi che condividono lo stesso stile
- `<div>Spedizione gratuita</div>`: I tag figli sono indentati sotto il tag genitore "promo"
- `<div style="display:none;">Nuovi giocattoli per cuccioli</div>` … `<div style="display:none;">Spedizione il giorno stesso</div>`: Messaggi promozionali