**18/09/2023**

**INTRODUZIONE**

Cosa è un **algoritmo**?

Un algoritmo, definito semplicisticamente, sarebbe una sequenza finita di istruzioni: formato quindi da un insieme di step, comprensibili ed eseguibili.

1° ALGORITMO 🡪 2° PROBLEMA 🡪 3° SUDDIVIDO I PROBLEMI 🡪 4° RISOLVO

Le proprietà **fondamentali** degli algoritmi sono:

EFFETTIVITA’: garantire la semplicità nell’utilizzo ed eseguibile meccanicamente.

FINITEZZA: poiché deve avere una sequenza finita di istruzioni.

OSSERVABILITA’: l’effetto prodotto deve essere osservabile, quindi deve garantire un risultato.

RIPRODUCIBILITA’: nella stessa situazione e nello stesso contesto di operazione deve sempre garantire lo stesso risultato.

Mentre le proprietà **non fondamentali** sono:

CORRETTEZZA: riesca a risolvere il problema senza produrre errore in output.

EFFICIENZA: risolvere il problema garantendo il massimo risultato con il minimo sforzo.

Se dovessimo trovare una vera e propria definizione di **algoritmo** sarebbe :”formalizzazione astratta della soluzione di un problema, che viene realizzato da esseri umani”.

**Il programma** invece risulta essere differente a livello di definizione, infatti :”sarebbe la descrizione dettagliata delle istruzioni presenti in un algoritmo e leggibili attraverso un calcolatore”.

**L’esecuzione**: “applicazione delle istruzioni ai dati svolta in automatico dal calcolatore”.

PROBLEMA

↓

ANALISI 🡪 SOLUZIONE INFORMALE

↓

FORMALIZZAZIONE 🡪 ALGORITMO

↓

PROGRAMMAZIONE 🡪 PROGRAMMA

↓

ESECUZIONE 🡪 RISULTATO

Tutta la parte che comprende la parte relativa al processo di programmazione e la creazione del programma, fanno parte **dell’Ambiente di Sviluppo.**

Per quanto riguarda l’esecuzione e il risultato fanno parte **dell’esecutore**.

**20/09/2023**

**CENNI STORICI**

Facciamo dei piccoli cenni storici che ci aiutano a capire come siamo arrivati fino a questo punto.

**Ada Lovelace** fù una matematica inglese che partecipò ad una festa organizzata da **Charles** **Babbage** (Babbage originated the concept of a digital programmable computer) per la presentazione della **macchina differenziale** (la macchina differenziale apre la possibilità di accedere a una vasta gamma di calcoli matematici).

**Ada** fu la traduttrice in inglese del primissimo articolo scritto da **Luigi Menabrea** (uno dei più grandi scienziati del XIX secolo), che poi lei decise di arricchire con svariate sue riflessioni.

Tra le varie annotazioni scritte da **Ada**, ci fu anche la descrizione di come utilizzare la **macchina** **analitica** per calcolare i numeri di **Bernoulli**.

Questo viene riconosciuto come l’instaurazione del **primo programma** **nella** **storia**, anche se la **macchina analitica** non era ancora una **macchina reale**; pertanto, questo fa di **Ada** “la prima programmatrice della storia”. (scienza astratta delle operazioni = informatica)

**Che linguaggio usare per definire un algoritmo?**

I programmi utilizzati sono vari, il tutto si basa sulle proprie conoscenze e sulle preferenze del programmatore in questione.

Noi ovviamente vedremo **python**, che di solito risulta essere quello più diffuso nella maggior parte dei contesti lavorativi.

Un altro fattore molto importante prima di definire gli algoritmi è l’applicazione del cosiddetto **pseudocodice**: un linguaggio informale per descrivere i passi di un algoritmo istituiti dalla mente umana.

Anche qua, come nel caso dei programmi, esistono vari tipi di **pseudocodice**, noi ci concentreremo nell’utilizzo dei **diagrammi di flusso**: per descrivere un algoritmo.

**TABELLA Diagramma di flusso:**

**Linea di flusso**: la freccia è usata per definire le operazioni che vengono compiute all’interno del diagramma, inizia in un terminale e finisce in un altro.

**Terminale**: utilizzato per indicare l’inizio e la fine di un algoritmo.

**Processo**: utilizzata per esprimere un’istruzione che viene eseguita e utilizzata per cambiare lo stato corrente di qualche variabile usata nell’algoritmo.

**Decisionale**: utilizzato per esprimere una condizione, creando due possibili rami, uno seguito se la condizione è vera; e un altro se la condizione che si presente non si verifica.

**Input/Output**: permette di specificare possibili input o output che vengono usati o restituiti dall’algoritmo solamente all’inizio o alla fine della sua esecuzione.

Quando all’interno di un **diagramma** si ripetono le operazioni all’interno della **condizione**, fino quando questa si verifica, esso viene chiamato “**ciclo di reiterazione**”.

**25/09/2023**

**LINGUAGGI DI PROGRAMMAZIONE**

Per comunicare con un **pc** (detto anche **nodo di rete**) si usa un **linguaggio di programmazione**:

formale, obbliga a seguire delle regole sintattiche che sono sviluppate per evitare istruzioni ambigue; una sola istruzione, in modo che tutte le frasi possano trasmettere un solo possibile significato.

Esistono *tanti linguaggi di programmazione*, si distinguono per **basso** e **alto** livello di astrazione da un linguaggio comprensibile da un calcolatore elettronico, il computer capisce un linguaggio che per noi è difficile da comprendere.

**Ci sono 3 insiemi**:

*linguaggio macchina* 🡪 un insieme di istruzioni che possono essere eseguite direttamente dal computer, attraverso l’utilizzo della CPU (è un processore di regolazione del computer che esegue i calcoli), il nostro computer comprende ciò che gli stiamo chiedendo unicamente se il linguaggio è binario. **Codice** sinonimo di **testo del programma**.

Una funzione in questo è definibile come un qualcosa che **prende** un **input** e **produce** un **output**.

*linguaggio di programma di basso livello* 🡪 forniscono un *livello di astrazione* **sopra** il **livello** **macchina**, sono più vicini al nostro linguaggio naturale; e serve per facilitare i programmatori a scrivere un programma (questo perché prima si scriveva unicamente in **binario,** e quindi rubava molto tempo nella stesura del codice). **Assembly** è un esempio di **linguaggio basso livello**. **Ogni linea** corrisponde a una **riga di istruzione** di un **linguaggio macchina**.

Tutti e due sono abbastanza difficili, dato che i comandi che si possono applicare con questi programmi sono abbastanza **primitivi e semplici**. Prima, infatti, si risolvevano tutti i problemi scomponendoli in piccoli passettini, anche per fare cose semplici, allungando di tantissimo il processo.

Inizia a esserci un certa attenzione **all’efficienza**, per questo i programmi erano fatti per usare le poche risorse che avevano questi computer, difficili e noiosi da usare.

linguaggi di programmazione di alto livello 🡪 qua abbiamo un **forte livello di astrazione**, ci avviciniamo quasi del tutto al linguaggio umano. Si possono usare parole per definire costrutti specifici, in modo che siano più comprensibili. **Sono quelli utilizzati** per svolgere e per scrivere programmi. Le righe del nostro programma sono più **complesse**, ma semplicemente perché fanno più cose, e sono anche più **flessibili**, ma allo stesso tempo non sono direttamente comprensibili da un calcolatore.

**Il computer** comprende **solo** il **binario**, pertanto utilizzeremo delle piattaforme che **traducono** il nostro linguaggio in **binario**.

Ce ne sono tanti, perché hanno caratteristiche specifiche utilizzabili in specifici contesti, il primo programma di questo tipo utilizzabile è stato **Fortran**.

**La genesi del web** è collegata a una motivazione molto minima, ossia per evitare i prestiti interbibliotecari e il passaggio o richiesta di riviste scientifiche (postino consegna).

**Il 90 % usa linguaggi di alto livello.**

Ci sono due modi per fare questa traduzione:

**linguaggi interpretati** 🡪 scriviamo il nostro **codice sorgente** (**programma**), dobbiamo produrre un codice che sia comprensibile dal nostro computer; per questo, grazie all’interprete, si riesce a tradurre il linguaggio, da programma a binario (**python** è un esempio di interprete).

**linguaggi compilati** 🡪 partiamo sempre da un **codice sorgente** con in mezzo un altro soggetto che viene chiamato **compilatore**, questo ha il compito di far passare da **codice sorgente** a **codice oggetto** (che sarebbe il codice unicamente leggibile dal nostro calcolatore), per poi utilizzare un altro programma, detto **esecutore**, che lo esegue applicandolo al calcolatore.

Il secondo caso ha più passi ma è più veloce nell’esecuzione totale.

**PYTHON**

Si chiama così perché chi l’ha creato gli piacevano i comici dei **monti python**, con simbolo del pitone.

Noi utilizzeremo **Python versione 3**.

È un chiaro esempio di **linguaggio interpretato**, semplice da usare, molto potente ed estendibile con nuove funzionalità:

**Pandas** 🡪 viene adoperato per fare **analisi** di dati, **caricare** i dati e fare **piccole** **operazioni** di statistica descrittiva.

**Seaborn** 🡪 ha le funzionalità di **data visualization** (possibilità di istituire grafici come vogliamo noi e molto più editabili rispetto a excel).

**Scikit-learn** 🡪 per fare operazioni di **machine** **learning** (è un sottoinsieme dell'intelligenza artificiale, che si occupa di creare sistemi che apprendono o migliorano le performance in base ai dati che utilizzano).

**L’ambiente Colab** è quello che noi utilizzeremo, ed è una piattaforma per **data scientist free** che ci permette di eseguire il **codice python** direttamente sul **cloud**, utilizzando i server di Google.

**INTRODUZIONE ALLA PROGRAMMAZIONE PYTHON**

È una sequenza di istruzioni che specifica come svolgere una computazione, le istruzioni devono essere scritte riga per riga e la **computazione** può riguardare un *problema matematico*, un *problema grafico*, *un problema simbolico*….

**Il cancelletto iniziale** serve per delimitare dei commenti che non verranno letti da parte di python.

**27/09/2023**

**print (“”)** 🡪 ha la funzione di stampare il messaggio che viene inserito all’interno delle virgolette.

**I linguaggi di programmazione** hanno molte differenze. Prima di tutto dobbiamo sapere le differenze tra questi concetti:

**input** 🡪 istruzioni per poter inserire file in ingresso, generalmente dalla tastiera.

**output** 🡪 istruzioni effettuate per dati in uscita, su schermo o file.

**condizionali** 🡪 istruzioni che verificano se specifiche condizioni accadono.

**di ripetizione(cicli)** 🡪 istruzioni per svolgere ripetutamente azioni.

Questi, come potremo aver già visto, sono anche gli elementi principali di un **diagramma di flusso**.

**I programmi** e **algoritmi** sono parametrici: il comportamento dipende da una serie di parametri di partenza; e descrivono la soluzione non di un singolo problema ma di una serie.

**Le istruzioni** che noi inseriamo fanno riferimento a **variabili:** sono dei valori che possono cambiare a seconda delle situazioni di lavoro e dell’esecuzione del programma:

variabili che contengono dati di input

varabili che contengono dati intermedi

variabili per salvare il risultato finale

**Variabili** 🡪 sono un nome che si riferisce ad un valore.

**Il nome**: può essere lungo a piacere, può essere composto sia da lettere che da numeri (deve iniziare con una lettera e non contenere spazi).

**Il valore** a cui la variabile fa riferimento può essere di diverso tipo: numero intero, razionale o stringa (mix alfanumerico o solo lettere).

L’operazione principale che riguarda le variabili è **l’assegnamento** 🡪 nome = valore

**Se la variabile non esiste**: si crea un nuovo nome e vi si associa un valore.

**Se la var esiste già**: il nuovo valore sostituisce quello vecchio.

**I nomi e i valori** che assegniamo all’inizio, poi alla fine vengono distrutti, in funzione di ciò che deve essere visualizzato.

**“\n”** 🡪 significa new line e serve per andare a capo nella visualizzazione finale.

**“\t”** 🡪 spazia le parole nella stampa.

Quando ripetiamo negli input l’assegnazione dello stesso valore al nome prenderà quello più recente.

Se metto **“”** non mi prende il calcolo, ma lo visualizza come un messaggio di testo, questo perché qualsiasi cosa scritta all’interno è definibile da python come un messaggio .

**Script sinonimo di programma**.

**Cos’è un’espressione?**

È una combinazione di variabili, valori ed operatori.

**OPERATORI ARITMETICI** 🡪 3+2 x – 5 (x+y) \* 6 14/3

**ELEVAMENTO A POTENZA** 🡪 x\*\*2

**MODULO** 🡪 quante volte ci sta questo numero in un altro e quanto lascia di resto: x % 2

**“”” “””** Tre apici all’inizio e alla fine del programma indicano la possibilità di inserire su un print più parole, una a capo all’altra.

Il comando **type** serve per stampare il tipo di un’espressione come:

**type (2)**

stamperebbe il risultato: **classe int**, perché è un numero intero.

Quando i programmi sono lunghi e complicati diventano difficili da leggere, ed è difficile ritornare a quel punto. Pertanto, è utile utilizzare i commenti, che verranno segnalati con **“# cosa vogliono inserire”**. Servono per spiegare degli aspetti non ovvi del programma, allo stesso tempo è importante inserire commenti solo utili al lettore.

09/10/2023

Per quanto riguarda gli **spazi bianchi**, non sono importanti nella formattazione, tranne quelli all’inizio delle righe: perché **python** li usa per capire a che livello logico stiamo scrivendo.

Servono per **organizzare sequenze di istruzioni comuni** sullo stesso punto iniziale, infatti, se noi scrivessimo più linee di codice con un inizio spostato rispetto al primo, questo ci darà **errore**.

**Tutte le istruzioni devono iniziare con gli stessi spazi iniziali.**

Per la **lettura dell’input**, ossia valore in ingresso, si usa la funzione **input**.

**var = input (“stringa”)**

Quello che c’è tra parentesi tonde lo stampa su schermo, rimanendo in attesa che l’utente scriva qualcosa e prema invio. Dopo di ciò il valore viene interpretato e visualizzato come **stringa**, creando una finestra dove io compilo il dato che voglio inserire. Per questo **python** rimarrà in attesa che io inserisca il valore.

Se invece vogliamo farci **operazioni matematiche**, bisognerà convertire i valori da **caratteri** nel corrispondente **valore numerico** e viceversa.

**Int (“”)** 🡪 il valore numerico che inserisco all’interno verrà interpretato come valore intero, prende solo i numeri interi, di fatto, se metto dei numeri razionali, comunque mi prenderà solo le cifre prima della virgola.

**Str() 🡪** il valore di tipo numerico me lo stampa sottoforma di stringa.

**Float (“”)** 🡪 mi converte un numero, non in numero intero ma in numero razionale.

Ovviamente tutte queste formule posso unirle in formule più complesse nella stessa riga, per esempio:

**numero = int(input(“Inserisci un numero: “))**

**print (numero +1)**

**ore = int(input(“Inserisci ora: “))**

**minuti = int(input(“Inserisci minuti: “))**

**secondi = int(input(“Inserisci secondi: “))**

**risultato = ore \* 3600 + minuti \* 60 + secondi**

**print (risultato)**

**11/10/2023**

**ESPRESSIONI BOOLEANE**

Siamo già a conoscenza e abbiamo già effettuate esercizi sulle espressioni numeriche e su stringhe.

Queste **espressioni** vengono utilizzate per realizzare le **condizioni**. Sono espressioni che seguono **l’algebra di Bool** e il risultato sono degli output **true** o **false**.

**Operatori razionali**:

**==** uguale **!=** diverso **<** minore **>** maggiore **<=** minore uguale **>=** maggiore uguale

!!!Non confondiamo l’operatore singolo “uguale” rispetto al “doppio uguale” delle booleane!!!

Tutte queste operazioni, che abbiamo appena visto, possono anche essere combinate con **operatori** **logici**, tra cui:

**1 and 2** 🡪 è vera se sia 1 che 2 sono vere.

**1 or 2** 🡪 è vera se uno dei due è vera.

**not 1** 🡪 è vera se è falsa, falsa se non è vera.

**Condizioni** **in** **PYTHON** 🡪 sono eseguibili attraverso l’ausilio di **IF** ed **ELSE**, ricordiamoci che ogni **IF** ed **ELSE** hanno bisogno di chiudersi con i due punti **“:”**

**IF** ed **ELSE** devono essere sulla stessa linea logica come anche le operazioni successive, questo si chiama **indentazione**.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Da quanto potremo aver notato la condizione porterà a delle modifiche del programma.

Questo che ho fatto su **python** viene chiamata **esecuzione alternativa**. Se noi mettiamo solo un **IF** allora sarà una **condizione singola**.

Possiamo anche costruire **IF** con **alternative multiple**, ma in questo caso si utilizza il comando **ELIF** che mi aprirà una condizione dentro un’altra condizione.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Ricordiamoci che i dati possono essere inseriti sia tramite finestra di input sia con assegnazione tramite uguale.

Conviene utilizzare la condizione **ELIF**, anche se funzionerebbe con delle **IF staccate**, così il programma non è obbligato a fare tutte le verifiche, ma solo una annidata.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Per utilizzare delle condizione con **pari e dispari si usa %**. Ed è sempre meglio inserire le operazioni tra parentesi, anche se poi il calcolo verrebbe lo stesso: è più una questione di logicità.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Queste che abbiamo appena visto sono delle **condizioni annidate**, poiché sono prese più condizioni all’interno della stessa condizione.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamenteImmagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Questi due sono chiari esempi di come **l’indentazione** possa variare l’output finale. Nel **primo** la **ELSE** sarà all’interno della **condizione annidata**, mentre nella **seconda** la **ELSE** sarà all’interno della **prima condizione**.

**16/10/2023**

**CICLI DI REITERAZIONE PYTHON**

I pc si usano per compiere in maniera automatica dei compiti ripetitivi, questi si chiamano cicli o reiterazione.

Serve per automatizzare compiti ripetitivi.

Il tipo più semplice è il seguente:

CICLO FOR 🡪 for n in range (1,11):

print (n)

Prende due valori separate da virgola, produce la lista di tutti i numeri compresi nell’elenco, fermandosi prima del secondo numero.

Questi valori sono validi nel corpo del ciclo, seguendo la stessa indentazione.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamenteImmagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamenteImmagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

|  |  |  |
| --- | --- | --- |
| Nel primo caso stamperà dieci volte 10. | In questo caso stamperà i numeri da 1 a 10. | In questo caso stamperà i numeri da 1 a 10 e dieci volte 10. |

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Questo algoritmo ci dà la possibilità di stampare i numeri da 1 a n, dove nIntero è un’assegnazione che converte l’input in intero. La variabile i che inserisco dentro il ciclo è un ulteriore variabile che serve per fare il print dei valori che vorremo stampare.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

CICLO WHILE: permette di fare forme più complesse di iterazione:

n = 1

while n < 11:

print (n)

n = n + 1

print (“Ho contato da 1 a 10”)

Determina se la condizione (n <11) è vera o falsa.

Se è falsa mi farà tutte le operazione all’interno del ciclo fino a quando arrivo a 10.

Alla fine di tutto mi stamperà l’ultimo print separato da indentazione.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Il corpo del ciclo while deve cambiare il valore delle variabili ad ogni iterazione, in modo tale che la condizione prima o poi diventi falsa, altrimenti il ciclo si ripete all’infinito.

La variabile n viene usata come un contatore: viene inizializzata con valore 1 e a ogni ripetizione del ciclo il valore aumenta di un valore fisso.

ACCUMULATORE: Un altro modo di usare le variabili è utilizzarle come degli accumulatori, dentro quella variabile continuo a sommare valori al suo interno. !!!Devo inizializzarla prima del ciclo!!!

È utilizzata per calcolare totali e somme.

Viene inizializzata prima dell’esecuzione del ciclo.

Ad ogni valore sommato, non sostituisce quello precedente ma viene comunque sommato fino alla fine.

USCITA ANTICIPATA DAL CICLO: serve per interrompere l’istruzione del ciclo e si utilizza il comando break.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente