SIMULAZIONE 15/05/24

Traccia Python: Studio statistico sintomatologia di un virus.

Durante un'emergenza epidemiologica, la ricerca scientifica si sta concentrando su vari aspetti per comprendere meglio la malattia e sviluppare strategie di gestione. Uno di questi aspetti riguarda l'analisi dei sintomi nei pazienti colpiti. In particolare, gli studiosi vogliono quantificare quanti pazienti hanno avuto sintomi gravi e verificare se esistono correlazioni tra l'età, il sesso dei pazienti e la gravità dei sintomi.

Vi viene quindi richiesto di creare un programma che cataloghi i *pazienti* in base a vari parametri: Nome, Genere, Anno di Nascita, gravità della malattia giorno per giorno.
SPIEGAZIONE GRAVITÀ DELLA MALATTIA GIORNO PER GIORNO: Alla fine del periodo di malattia, ogni paziente crea una lista di voti da 1(sintomi lievissimi) a 10(sintomi gravissimi) ordinata per giorno della gravità della sintomatologia.
Esempio:
Giorno 1: Gravità 3
Giorno 2: Gravità 5
Giorno 3: Gravità 6
Giorno 4: Gravità 3
Giorno 5: Gravità 1
Questa successione si trasforma nella seguente lista: [3,5,6,3,1]

Il programma deve partire con almeno i successivi pazienti:

1. Sofia Marini, Femmina, 2000, [1,2,3,5,6,2,1]
2. Filippo Cadei, Maschio, 2007, [4,8,3,1]
3. Leonardo Manfredi, Maschio, 2012, [1,4,7,1,1]

Il software, attraverso un comodo menù, deve permettere all'operatore di scegliere tra le funzioni riportare di seguito:

1. Stampaggio di tutti i pazienti in un formato comodo e funzionale;
2. Aggiunta di un paziente con i relativi dati;
3. Rimozione di un particolare paziente;
4. Ricerca del numero di pazienti Under-18 e Maschi che hanno riportato almeno una volta gravità della malattia 7;
5. Calcolo della media della gravità della malattia in uno specifico paziente. Avvertire l'operatore se la media risulta maggiore di 6;
6. Conteggiare quanti sono i maschi colpiti e femmine colpite. Dire quindi il numero di pazienti totale ed il genere che ha avuto più infettati.

FORMULARIO PYTHON

CICLO FOR:

OPZIONE 1:
Cicliamo attraverso una lista elemento per elemento.
Il valore attuale viene salvato nella variabile “elemento”.

```
for elemento in lista:
    print(elemento)#istruzione d’esempio
```

OPZIONE 2:
Cicliamo n. volte.
Nella variabile conteggio viene salvato quante volte il ciclo è stato eseguito fino a quel momento.

```
for conteggio in range(n):
    print(conteggio)#istruzione d’esempio
```

CICLO WHILE:
Ciclo infinito con possibilità di stop

```
scelta = True
while scelta:
    print(“Interno ciclo”)#istruzione d’esempio
    scelta = int(input(“Inserire 1 per mantenere il ciclo funzionante,Inserire 0 per spegnerlo”)
```

CLASSE D’ESEMPIO:

```
class Persona: #Persona nome della classe
    def __init__(self,attributo1, attributo2):
        self.attributo1=attributo1
        self.attributo2=attributo2

    def __str__(self):
        return f”””
        Attributo1: {self.attributo1}
        Attributo2: {self.attributo2}
        “””
```

```
persona1 = Persona(“valore1”,”valore2) #Creazione di una variabile con la classe Persona
```

LIBRERIA RANDOM

```
import random #da inserire all’inizio del programma
scelta_random = random.choice(lista) #Scegliere elemento a caso da una lista
numero_casuale = random.randint(1, 10) #Generazione numero casuale tra 1 e 10
```

OPERAZIONI BASE LISTE:

```
Lista = [] #Creazione lista vuota
Lista.append(elemento) #Aggiunta elemento a Lista
valore = Lista[3] #Estrarre elemento alla posizione 3 della lista e salvarla in valore
Lista.remove(elemento) #Rimuovere elemento dalla lista
lunghezza = len(lista) #numero di elementi di una lista
```

CASTING

```
immissione = int(input(“Valore da inserire”)) #Trasformazione input da stringa ad intero
```

Traccia C: Propagazione di un virus

Obiettivo:

Implementare un simulatore di contagio di un virus utilizzando il linguaggio di programmazione C. Vengono inoltre proposti dei semplici punti aggiuntivi per aumentare il voto.

Descrizione:

Si richiede di sviluppare un'applicazione console in C che permette di prevedere la gravità di un'epidemia. L'applicazione deve permettere all'utente di inserire 3 parametri:

- **Il tasso di contagio giornaliero** (numero decimale): rappresenta quante persone può infettare una singola persona. Ad esempio, con un tasso di contagio pari a 5 e una sola persona infetta iniziale si avrebbe:
  - Il 1° giorno un solo infetto
  - Il 2° giorno ci sono 6 infettati(i 5 nuovi infettati più quello di prima) = 5*1 +1
  - Il 3° giorno ci sono 36 infettati (i 30 nuovi infettati più quelli di prima) = 5*6 +6
- **Il numero di persone totale** (numero intero):è la popolazione totale su cui si vuole simulare il contagio
- **Il numero di infetti iniziale** (numero intero): è il numero che indica quante persone (comprese nel totale) sono contagiate all'inizio della simulazione.

L'applicazione deve mostrare un menù con le seguenti opzioni:

1. Calcolare quanti infetti ci sarebbero dopo un certo numero di giorni
2. Calcolare in quanti giorni si infetterebbe il totale della popolazione

Spiegazione opzioni menu

1. La prima opzione deve permettere di far inserire all'utente il numero di giorni da simulare (la simulazione non è altro che il calcolo giorno per giorno delle persone che si contagiano) dopodiché stampare quanti ammalati ci sono dopo quel numero di giorni.
2. La seconda opzione deve invece simulare in quanti giorni si avrebbe la totalità della popolazione infettata.

Come funziona il contagio

Il contagio funziona in un semplice modo: ogni giorno le persone ammalate contagiano a testa altre N persone. Nello specifico ogni persona contagia tante persone quanto indica il tasso di contagio giornaliero.
Se si hanno 2 persone iniziali e un tasso di contagio di 10 bisogna fare queste considerazioni:

- Ogni persona infetta contagia 10 persone
- Quindi poiché sono 2 persone iniziali il giorno 0 ci sono solamente 2 ammalati
- Il giorno 1 si hanno 10 persone contagiate per ogni persona ammalata che c'è: essendoci 2 persone ammalate ognuna di loro contagia 10 persone, bisogna però ricordarsi anche degli ammalati precedenti: il primo giorno ci saranno quindi 10 * 2 + 2 ammalati (i nuovi 20 contagiati più i 2 ammalati di prima, quindi un totale di 22) .
- Il giorno 2 si hanno 10 persone contagiate per ognuna delle 22 ammalate del giorno prima: 10*22 + 22 = 242 persone ammalate
- … il calcolo va avanti nello stesso modo ogni giorno…

Esempio funzionamento programma

(è un esempio!!! Deve andare anche con altri valori)

Simulazione contagio virus: *Inserisci il tasso di contagio: 5.5* *Inserisci la popolazione totale: 10000* *Inserisci gli ammalati iniziali: 3*	Richiesta dei dati iniziali
Scegli un'operazione: *0) Esci* *1) Calcolo contagi in N giorni* *2) Giorni per contagio completo* *Inserisci il numero dell'operazione che vuoi fare: 1*	Viene stampato il menu principale E viene chiesta l'opzione, viene scelto di calcolo dei contagiati in N giorni
Inserisci per quanti giorni simulare: 3 *In 3 giorni sono state contagiate 823.875000 persone*	Vengono chiesti quanti giorni simulare Vengono detti quanti contagiati ci sono stati
Scegli un'operazione: *0) Esci* *1) Calcolo contagi in N giorni* *2) Giorni per contagio completo* *Inserisci il numero dell'operazione che vuoi fare: 2*	Viene ristampato il menu principale E viene chiesta l'opzione
In 5 giorni sono state contagiate tutte le persone	In quanti giorni si ha il contagio totale
Scegli un'operazione: *0) Esci* *1) Calcolo contagi in N giorni* *2) Giorni per contagio completo*	Viene ristampato il menu principale
Inserisci il numero dell'operazione che vuoi fare: 0 *Hai scelto di uscire dal programma: arrivederci!*	Ha scelto l'opzione *0*: allora il programma termina

Per uscire dal programma bisogna scegliere per forza 0.

Requisiti minimi applicazione:

1. Il programma non deve terminare dopo aver scelto un'opzione ma solamente se viene scelto il numero corrispondente all'opzione "*Esci*".
2. L'applicazione deve essere in grado di fare i calcoli con i numeri con la virgola.
3. L'applicazione deve ripetere il processo di lettura dell'input fino a quando l'utente decide di uscire o terminare il programma.

Requisiti minimi del codice:

1. Utilizzare funzioni separate per eseguire ciascuna operazione del menu (tranne l'uscita).
2. Gestire gli errori come: tasso di contagio inserito è 0, popolazione è 0, ammalati iniziali sono 0.
3. Implementare un loop principale che consenta all'utente di fare più scelte senza dover riavviare il programma ogni volta.
4. Fornire un'interfaccia utente chiara e intuitiva, con istruzioni per l'utilizzo e il riscontro (la conferma o errori su ciò che l'utente fa) sull'input.
5. Inserire i commenti per spiegare cosa si sta facendo (per inserire i commenti su una riga usare // oppure su più righe scrivendo /* qua va il commento su più righe */).
6. Il codice deve essere il più chiaro possibile, evitare di creare variabili superflue.

Implementazioni aggiuntive:

1. *Menu in una funzione:*
   Bisogna scrivere il codice per stampare il menu e chiedere l'opzione in una funzione che non sia il "main".
2. *Guarigione delle persone:*
   Aggiungere il tasso di guarigione: all'inizio del programma chiedere anche un tasso di guarigione giornaliere (cioè quante persone al giorno guariscono).
   In questo modo quando si calcolano i giorni di contagio bisogna tenere conto anche di quante persone guariscono al giorno. (controllare che il tasso di guarigione sia < del tasso di contagio)

   La formula è: ammalati = ammalti * tasso_contagio + ammalati * (1 - tasso_guarigione)

   Ad esempio: tasso guarigione : 4.0 tasso contagio : 5.8
   ammalati_giorno_0 = 3
   *ammalati_giorno_1 = 3 * 5.8 +3* (1 – 4.0)*
   *ammalati_giorno_2 = 8.4 * 5.8 + 8.4 * (1 - 4.0)*
   *ammalati_giorno_3 = 23.52 * 5.8 + 23.52 * (1 - 4.0)*
3. *Usare le strutture:*
   Utilizzare una struttura per rappresentare i dati di: contagio, guarigione, popolazione e ammalati iniziali al posto di usare le singole variabili.

Note:

- Siete incoraggiati a implementare funzionalità aggiuntive per migliorare l'usabilità e la robustezza dell'applicazione.

- Valutare la gestione degli errori, la correttezza dell'implementazione delle operazioni e la chiarezza del codice.

Formulario C

tipo: tipo di una variabile: int, float, double, bool, char oppure una struct definita da voi
nome: nome della variabile/funzione/struttura (NO SPAZI)
condizione: condizione booleana (vero o falso) da verificare
valore: valore o espressione (può essere un numero/simbolo/variabile o un' espressione matematica)
parametri: lista di dichiarazione di parametri divisi da virgola (int param1, double param2, ...) per una funzione

Legenda termini

Simbolo	*Variabili associate*
%d	int/bool
%f	float
%lf	double
%c	char

Esempio richiesta di un numero intero all'utente

```
int variabile;
printf("Inserisci valore: ");
scanf("%d",&variabile);
```

Esempio menu

```
int opzione;
printf("Menu:\n 1) Opzione 1 \n 2) Opzione 2 \n ...");
scanf("%d",&opzione);
```

Esempio di come si stampa una variabile

```
double variabile = 9.57;
printf("Il valore è : %lf", variabile);
```

Struttura if / else if / else

```
if( condizione1 ){
        // codice se condizione1 è vera
} else if( condizione2 ){
        // codice se condizione2 è vera
} ...quanti else-if si vogliono ...
else{
        // codice se nessuna delle precedenti condiz. è vera
}
```

Struttura ciclo while

```
while( condizione ){
        // codice ripetuto finché la condizione è vera
}
```

Struttura ciclo for (esempio per eseguire il ciclo 50 volte)

```
for(int i=0; i < 50; i++){
        // codice da eseguire il numero di volte specificato
}
```

Esempio dichiarazione array

```
// dimensione è un numero o variabile intera
tipo nome[dimensione];
```

Esempio assegnamento array

```
// posizione è un numero o variabile intera compresa tra 0 e
// dimensione-1
nome[posizione] = valore;
```

Creazione e input valori in array (per l'output basta cambiare e usare la printf)

```
int esempio[10];
for(int i=0; i < 50; i++){
    printf("Inserisci valore %d nell'array", i);
    scanf("%d", &esempio[i] );
}
```

Definire una funzione:
(PRIMA DEL MAIN)
- il tipo è void (se non ritorna nulla) oppure uno tra: int, float, double, bool e char
- se non ha parametri si lasciano le parentesi vuote: ()

Def. Funzione prima di main

```
tipo nome(parametri){
   // codice
   // return se tipo non è void
}
```

Chiamare una funzione

```
nome(parmetri);
//se ha un ritorno != void
risultato = nome(parametri);
```

Definire una struttura
(PRIMA DEL MAIN)

```
typedef struct {
// definizioni variabili divise da punto e virgola
} nome;
```

Esempio definizione e uso struttura

BMI è un nuovo tipo di dato che si può usare allo stesso modo come qualsiasi altro tipo (int,float,...)

Def. prima del main

```
typedef struct {
  int altezza;
  double peso;
} BMI;
```

Utilizzo in una funzione o main

```
BMI persona1;
persona1.altezza = 50;
persona1.peso = 24.5;
```