Concetti di programmazione

(con il linguaggio Python)

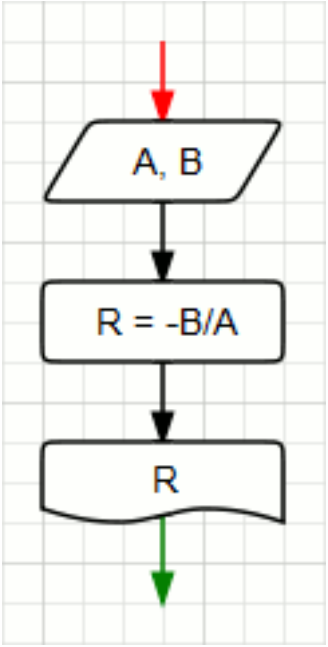
Si ricordano le tre strutture di controllo della programmazione strutturata:

1. Sequenza
2. Selezione
3. Iterazione

Nel seguito queste tre strutture verranno analizzate tramite esempi in Python.

SEQUENZA

"Il seguente diagramma mostra la struttura generale della sequenza. In rosso viene mostrato l'unico punto d'ingresso, in verde l'unico punto d'uscita." (Wikipedia)



Una **sequenza** è composta da una successione di istruzioni. Più istruzioni in sequenza costituiscono un **blocco**.

ATTENZIONE: in Python un blocco di istruzioni deve essere specificato indentando tutte le istruzioni che lo compongono, cioè rientrandole verso destra rispetto alle istruzioni esterne al blocco.

In Python per mettere in sequenza più istruzioni le si deve porre una per riga, ad esempio:

```
a = 10
b = 5
```

```
c = a + b
```

È importante che in una sequenza d'istruzioni venga rispettato l'ordine corretto. Ad esempio è necessario assegnare un valore a una variabile prima di utilizzarla in una espressione. Perché la seguente istruzione vìola questa regola?

```
x = x + 1
```

SELEZIONE

Selezione semplice

L'esecuzione di una istruzione o di un blocco d'istruzioni viene subordinata dalla verifica di una condizione logica:

```
if <condizione>:
    <istruzione/blocco>
```

Esempio 1:

```
prezzo = 120
if prezzo > 100:
    prezzo = prezzo *0.80
print(prezzo)
```

Esercizio 1: dato un numero intero (positivo o negativo) stampare il suo valore assoluto.

Esercizio 2: dato un intero positivo n controllare se è divisibile

- *per 2 (con l'operazione modulo n%2), se sì stampare "Divisibile per 2";*
- *per 3 (con l'operazione modulo n%3) se sì stampare "Divisibile per 3";*
- *per 5 (con l'operazione modulo n%5) se sì stampare "Divisibile per 5".*

Selezione con alternativa

```
if <condizione>:
    <istruzione/blocco>
else:
    <istruzione/blocco>
```

Esempio 2:

```
prezzo = 120
if prezzo > 100:
    prezzo = prezzo *0.80
    print("Sconto 80%")
else:
    print("Articolo non soggetto a sconto")
print(prezzo)
```

Esercizio 3: dato in input un anno verificare se è bisestile. Un anno è bisestile se è divisibile per 4 ma non per 100, oppure se è divisibile per 400. Se è bisestile stampare la stringa "L'anno inserito bisestile", altrimenti "L'anno inserito non è bisestile".

Selezione multipla o a casi

Questo tipo di selezione si utilizza quando le condizioni da verificare sono più di una.

```
if <condizione 1>:
    <istruzione/blocco>
elif <condizione 2>:
    <istruzione/blocco>
. . .
elif <condizione n>:
    <istruzione/blocco>
else:
    <istruzione/blocco>
```

Esercizio 4: dato un numero compreso tra 1 e 12 stampare il nome del mese corrispondente.

ITERAZIONE

L'istruzione while

In Python vi sono due tipi di costrutti linguistici per realizzare l'**iterazione**, il **while** e il **for**.
Vedremo prima l'uso del **while**, sia in caso di **iterazione indefinita** (cioè quando non si conosce a priori quante volte deve essere iterato la stessa istruzione o blocco d'istruzioni), sia per l'**iterazione definita** (anche se in questo caso il costrutto preferibile è il **for** che vedremo in seguito)

```
while <condizione 1>:
    <istruzione/blocco>
```

Esercizi con while

Esercizio 5: continuare a inserire in input una serie di numeri interi finché non se ne incontra uno divisibile per 7, allora terminare stampandolo (iterazione indefinita)

Esercizio 6: stampare i primi 10 numeri pari (iterazione definita, while con un contatore).

Esercizio 7: stampare i primi n termini della successione di Fibonacci.
(Iterazione definita)

Per sapere di più sulla stupefacente successione di Fibonacci leggi le pagine seguenti!

La successione di Fibonacci

Con il suo *Liber Abaci* (1202), il matematico italiano *Leonardo da Pisa* detto ***Fibonacci*** fu uno dei primi studiosi a introdurre nel mondo occidentale il *sistema numerico decimale* (chiamato nel libro "*modus indorum*", dato che fu utilizzato originariamente da matematici indiani). Oltre ad aver contribuito a questo fondamentale cambiamento nella storia della Matematica, in questo corposo trattato di aritmetica vengono studiate le proprietà delle quattro operazioni e alcune caratteristiche di numeri "particolari", come i numeri perfetti o i numeri primi. Nel dodicesimo capitolo, Fibonacci introduce inoltre un metodo per ricavare una successione numerica: in seguito, questa successione verrà ricordata proprio con il nome di ***successione di Fibonacci***.

La successione di Fibonacci nel *Liber Abaci*

Vediamo come, nel testo da lui scritto, Fibonacci introdusse la successione a lui intitolata.

Supponiamo di voler studiare come si evolve la popolazione di una colonia di conigli, costituita inizialmente da una sola coppia di conigli, che rispetti le seguenti condizioni:

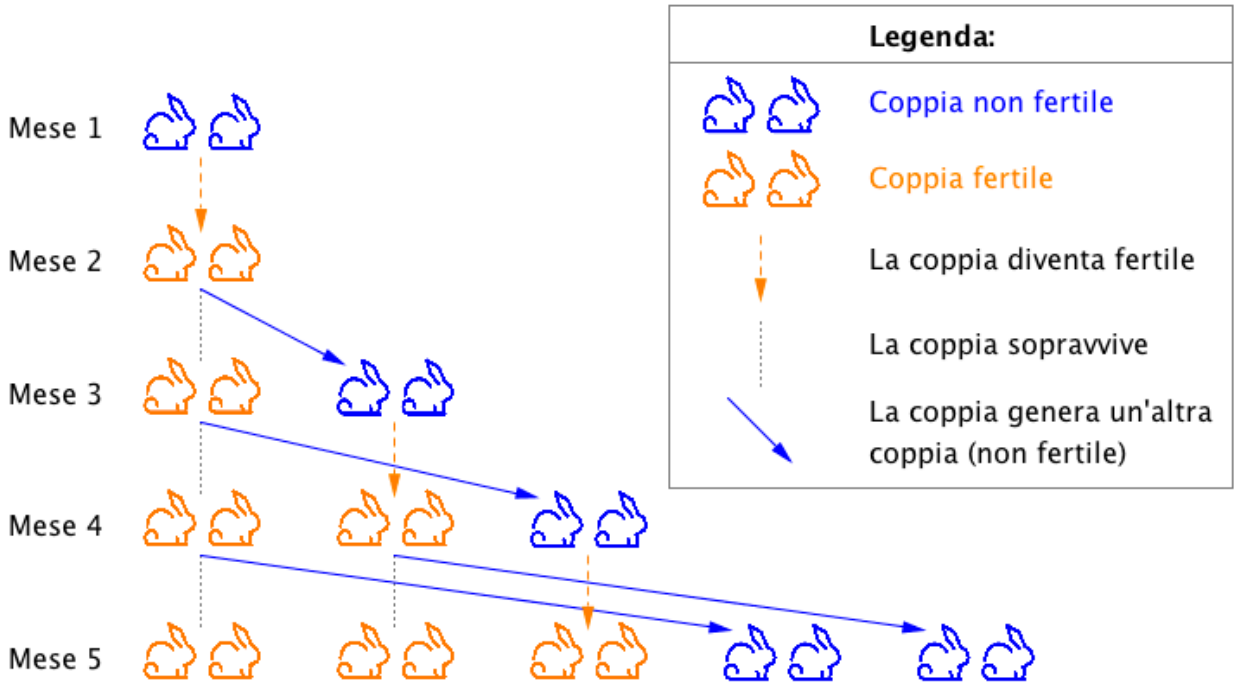
- ogni coppia di conigli genera solamente un'altra coppia di conigli alla volta;
- i conigli hanno bisogno di un mese di vita per diventare fertili;
- una volta fertile, ogni coppia continuerà a generare una coppia di conigli al mese.

Ecco cosa accade alla popolazione della colonia dei conigli:

1. Nel primo mese, abbiamo solo la coppia di partenza.
2. Nel secondo mese, abbiamo nuovamente la coppia di partenza, che nel frattempo è diventata pronta a generare un'altra coppia di conigli.
3. Nel terzo mese, la coppia di partenza ha generato una coppia di conigli, che non è ancora fertile. Quindi le coppie sono diventate **2** di cui una sola è fertile.
4. Nel quarto mese, la coppia di partenza ha generato un'altra coppia di conigli. La coppia ottenuta al passo 3 è diventata fertile, ma non ha ancora generato un'altra coppia. Quindi le coppie ora sono **3**, di cui 2 fertili.

5. Nel quinto mese, le due coppie fertili hanno generato un'altra coppia ciascuna. Quindi le coppie di conigli sono diventate **5**; le due coppie appena nate non sono fertili, mentre le altre 3 lo sono.

Riassumiamo questi cinque passaggi con la seguente illustrazione:



Potremmo andare avanti all'infinito con questo procedimento, ma quello che conta è rendersi conto di questo fatto fondamentale:

il numero di coppie di conigli al passo n, con n≥3, è uguale al numero di coppie di conigli al passo n–1 più il numero di coppie di conigli presenti al passo *n*–2. Infatti:

- al passo 3 le coppie sono 2, che è proprio uguale al numero di coppie al passo 2 (cioè una coppia) più il numero di coppie al passo 1 (che è sempre una coppia);
- al passo 4 le coppie sono 3, che è uguale al numero di coppie al passo 3 (cioè 2 coppie) più il numero di coppie al passo 2 (cioè 1 coppia);
- al passo 5 le coppie sono 5, che è uguale al numero di coppie al passo 4 (cioè 3 coppie) più il numero di coppie al passo 3 (cioè 2 coppie);

e così via. Se elenchiamo in fila il numero di coppie di conigli presente in ciascun mese, otteniamo la successione:

1,1,2,3,5,…
che prende il nome di **successione di Fibonacci**.

La successione di Fibonacci dal punto di vista matematico

Possiamo definire in maniera decisamente più astratta la successione vista prima, seguendo la definizione di successione numerica.

Consideriamo la successione $(f_n)_{n\in\mathbb{N}}$ definita nel seguente modo:

$$\begin{cases} f_1 = 1; \\ f_2 = 1; \\ f_n = f_{n-1} + f_{n-2} \quad n \geq 3 \end{cases}$$

La successione $(f_n)_{n\in\mathbb{N}}$ come si dice con un termine tecnico, è *definita per ricorsione*. Se elenchiamo i suoi elementi otteniamo proprio la successione descritta da Fibonacci:

1, 1, 2, 3, 5, 8, 13, 21, …

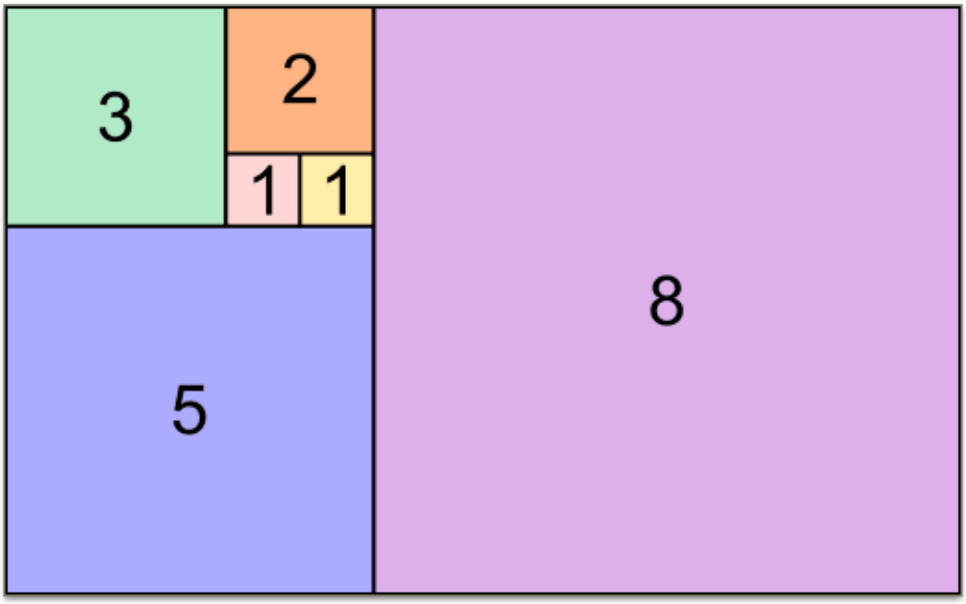
Infatti:
$f_1 = 1$
$f_2 = 1$
$f_3 = f_2 + f_1 = 1 + 1 = 2$
$f_4 = f_3 + f_2 = 2 + 1 = 3$
$f_5 = f_4 + f_3 = 3 + 2 = 5$
$f_6 = f_5 + f_4 = 5 + 3 = 8$
$f_7 = f_6 + f_5 = 8 + 5 = 13$
$f_8 = f_7 + f_6 = 13 + 8 = 21$
…
(estratto da https://library.weschool.com/lezione/successione-leonardo-fibonacci-sequenza-liber-abaci-13194.html)

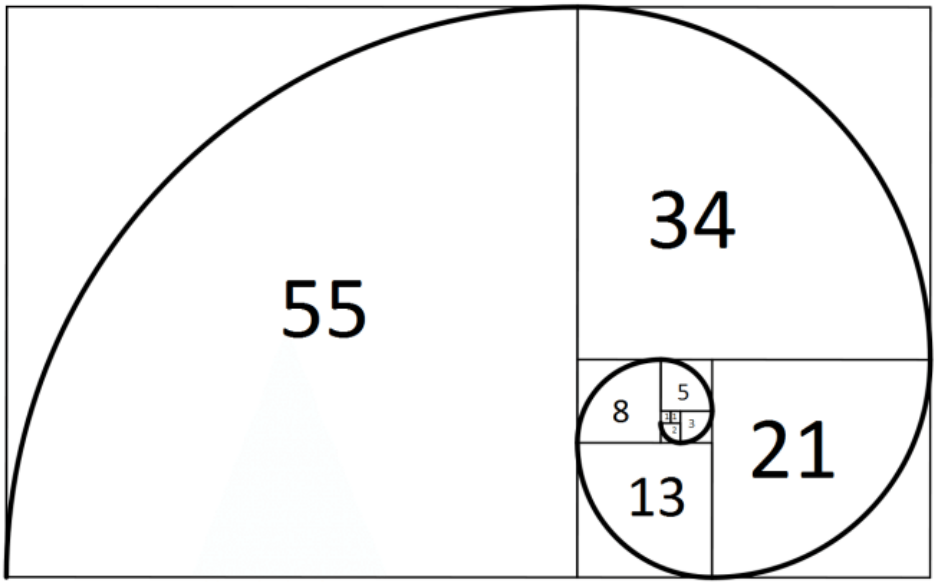
Successione di Fibonacci in natura

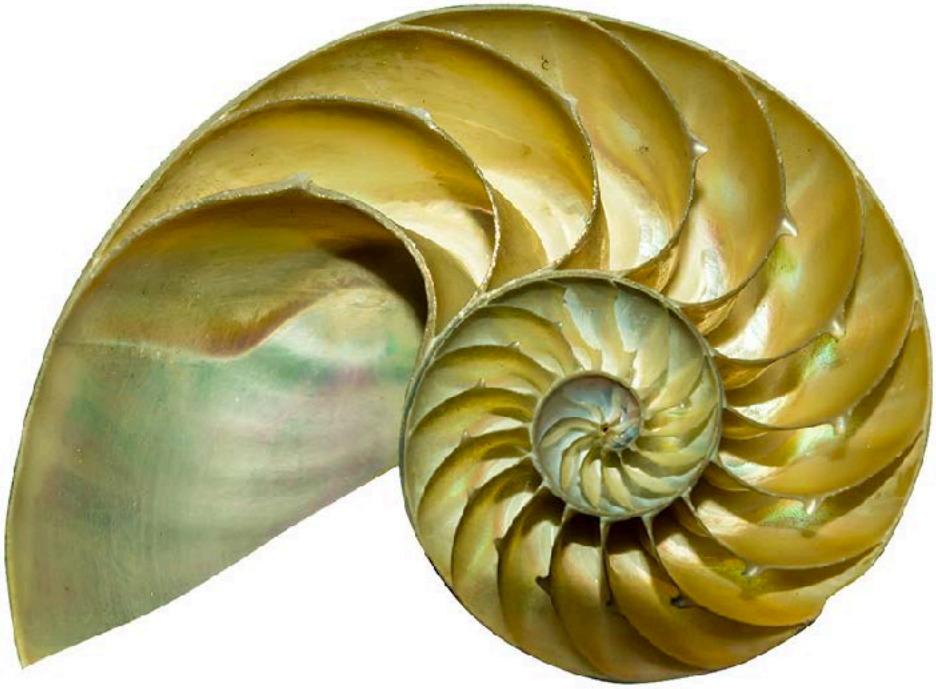
1. In quasi tutti i fiori **il numero dei petali è un termine della successione di Fibonacci**. Ne sono un esempio gigli e iris (tre petali), ranuncoli, rose canine, plumeria (5 petali) e alcune margherite (8 o 13 petali).
2. Come nel caso dei girasoli, i pistilli sulle corolle dei fiori sono spesso disposti secondo uno schema preciso, formato da spirali il cui numero corrisponde a un numero della successione di Fibonacci.

3. Sui rami degli alberi le foglie non hanno una collocazione casuale, ma sono disposte in modo da non farsi ombra l'una con l'altra e quindi ricevere la luce del sole in modo uniforme. Prendendo come punto di partenza la prima foglia di un ramo e contando il numero di foglie che ci sono fino a quella perfettamente allineata con la prima, si ottiene un numero di Fibonacci.
4. Affiancando in successione tanti quadrati, ognuno avente per lato un termine della successione di Fibonacci, si ottiene un rettangolo aureo. Partendo dalla successione di Fibonacci e con l'aiuto di riga e compasso si può disegnare una spirale che approssima molto bene la spirale aurea.

(estratto da https://www.youmath.it/domande-a-risposte/view/5950-successione-fibonacci.html)







L'istruzione for

for è l'istruzione che in Python (e in molti altri linguaggi di programmazione) permette di definire delle iterazioni. Per decidere quante iterazioni svolgere useremo il costrutto **range**:

```
for i in range(n):
    <istruzione/blocco>
```

dove **i** (il nome non è importante) è una variabile intera che assume ad ogni iterazione uno dei valori prodotti dalla funzione **range**.

La funzione range

- **range(start, stop, step)** produce la successioni d'interi che inizia con **start** e termina con il valore che precede **stop** (cioè stop è il primo intero che non fa parte della successione), procedendo con passo **step**.
  Esempio: **range(10, 20, 2)** produce **10, 12, 14, 16, 18**
- Se **range** viene applicata su due soli parametri, essi assumono il significato di start e stop. Il passo della successione in questo caso è 1.
  Esempio: **range(10, 15)** produce **10, 11, 12, 13, 14**
- Nel caso in cui in range compare un solo parametro esso assume il ruolo di stop e i valori di default di start e step sono rispettivamente 0 e 1.
  Esempio: **range(7)** produce **0, 1, 2, 3, 4, 5, 6**

Esempio:

```
for num in range(0,11,2): print(num**2)
```

stampa la sequenza: 0, 4, 16, 36, 64, 100.

N.B. È possibile contare eseguire un conto alla rovescia, ma in questo caso è comunque necessario specificare un passo negativo.
Esempio:

```
for num in range(10,-1,-1): print(num)
```

stampa la sequenza: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.

Vedremo in seguito che con **for** è possibile eseguire iterazioni su qualunque oggetto che possa essere considerato una sequenza (ad esempio una stringa):

```
for car in "ciao": print(car*3)
```

stampa: "ccc", "iii", "aaa", "ooo"

Esercizi con for

Esercizio 8: stampare numeri pari dall'n-esimo all'm-esimo, dove n ed m sono ricevuti in input.

Esercizio 9: stampare i primi n termini della successione di Fibonacci, utilizzando l'istruzione for.