L’ARTE DI USARE I CICLI FOR

I cicli FOR sollevano il programmatore da molte fastidiose ripetizioni, all’interno dei programmi.

Quando, ad esempio, vogliamo che il computer ripeta una serie di operazioni un numero prefissato di volte, si usa un ciclo FOR.

Come vedremo, i cicli FOR si possono anche usare per "dipingere" col computer, ma sono anche utili nei giochi e nei programmi applicativi.

COS'E' UN CICLO FOR

In Python, un ciclo FOR è una particolare struttura per far ripetere al computer più volte una stessa operazione.

Supponiamo per esempio, di voler calcolare i quadrati di tutti i numeri da 1 a 100. Potremmo scrivere:

```
print(1, 1**2)
print(2, 2**2)
print(3, 3**2)
```

e via dicendo (il simbolo ** indica l'operazione di elevazione a potenza).

Il computer, ad ogni richiesta così formulata, visualizzerebbe il risultato. Un simile sistema, però, è alquanto faticoso.

Per un uso più efficiente delle capacità del computer si scriva invece:

```
for i in range(1,101):
    print(f"il quadrato di {i} è {i**2}")

print("Ecco fatto")
```

Il programma fa sì che il computer visualizzi, automaticamente, il numero 1 e il suo quadrato, poi il numero 2 e il suo quadrato e così via, fino ad arrivare al numero 100.

Come ciò possa accadere è presto detto. Quando il Python incontra l'istruzione 'for', sa che le successive linee di programma indentate sotto la FOR devono essere ripetute un certo numero di volte (nel nostro esempio devono essere ripetute 100 volte), ed ogni volta la variabile 'i' assume i valori forniti da range().

La 'for' quindi è un ciclo, come la 'while' vista nella precedente lezione, ma mentre la 'while' si ripete finchè si verifica la condizione indicata, la 'for' si ripete finchè si esaurisce il numero di valori forniti da range().

LA FUNZIONE range()

range() è una funzione che tira fuori una lista prefissata di numeri secondo quanto specificato nei parametri.

Facciamo un po' di prove; apriamo la console di python e digitiamo 'list(range())' e mettiamo dentro le parentesi uno, due o tre numeri separati da virgola:

```
$ python3
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(3,11))
[3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(1,21,2))
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
>>> quit()
```

Nel primo caso (cioè con un solo parametro) range ha tirato fuori esattamente 10 numeri come specificato, a partire da 0 fino ad arrivare a 9.
Nel secondo caso (con due parametri) ha tirato fuori 8 numeri, a partire dal primo specificato (3) fino ad arrivare all'ultimo specificato (11) ESCLUSO (quindi arriva a 10).
Nel terzo caso (con 3 parametri) anche si ottiene una lista di numeri a partire dal primo (1) fino ad arrivare all'ultimo specificato (21) escluso, ma ogni volta avanza di 2 anzichè di 1 come nei precedenti. Il terzo parametro è quindi il 'passo' o 'step'. In questo caso abbiamo quindi tirato fuori tutti i numeri dispari.

E' possibile tirare fuori anche i numeri in ordine inverso specificando un passo negativo

```
$ python3
>>> list(range(10,0,-1))
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> quit()
```

MicroTip: usiamo il ciclo giusto

L’uso dei cicli FOR è assai frequente, ma altrettanto frequenti sono I casi nei quali è bene evitarne l’uso.
La regola generale è:
Quando si desidera far ripetere un gruppo di istruzioni un numero prefissato di volte, senza interruzioni di sorta, allora deve essere impiegato un ciclo FOR.
Quando, invece, si desidera far ripetere una certa sequenza fino al verificarsi o meno di una condizione, è preferibile utilizzare il ciclo WHILE.

LA FUNZIONE time.sleep()

Il computer è abbastanza veloce ed i risultati del programma vengono forniti in maniera molto rapida, e se vengono stampate molte informazioni a schermo, l'utente non fa in tempo a leggerle. Talvolta è quindi utile inserire qualche pausa. Questo si può fare con la funzione time.sleep(secondi)

```
import time

for i in range(1,11):
    print(f"il quadrato di {i} è {i**2}")
    time.sleep(0.5)

print("Ecco fatto")
```

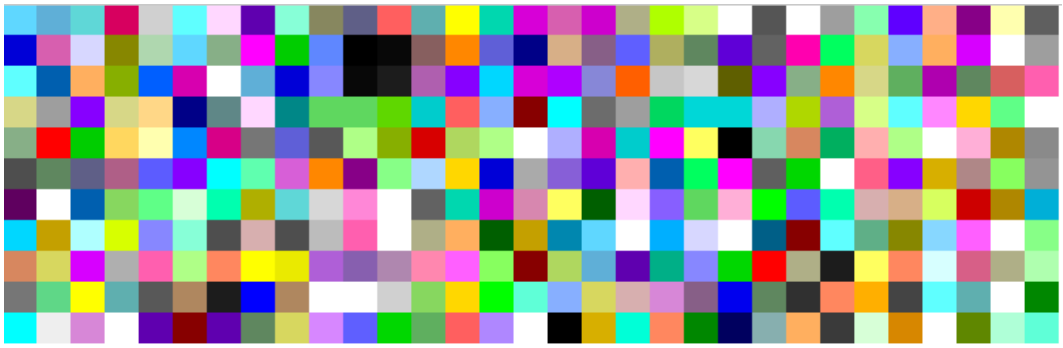
In questo caso l'elenco dei quadrati precendenti viene fornito con una pausa di mezzo secondo tra ognuno.

DIPINGERE COI NUMERI

Un po' per gioco e un po' per imparare qualcosa ancora sui cicli, vediamo come ottenere degli effetti grafici utilizzando le FOR. Ecco un esempio:

```
 1.  import screen
 2.  from colors import color
 3.  import random
 4.  import time
 5.
 6.  screen.clearScreen()             # cancella lo schermo
 7.  screen.cursorHide()              # nascondiamo il cursore
 8.
 9.  for i in range(1000):            # ripetiamo le operazioni 1000 volte
10.      colonna = random.randint(0, 30)        # scegliamo una colonna a caso
11.      riga =    random.randint(0, 10)        # una riga a caso
12.      colore =  random.randint(0, 255)       # un colore a caso
13.      screen.cursorTo(colonna*3, riga)       # portiamo il cursore nella riga e colonna
14.      print(color("   ", bg=colore), end="") # stampiamo un blocco colorato
15.      time.sleep(0.01)                       # piccola pausa di 10 millisecondi
16.
17.  screen.cursorTo(0, 11)          # riportiamo il cursore al suo posto
18.  screen.cursorShow()             # e mostriamolo
```

Se non ci sono errori questo dovrebbe essere il risultato:



Nelle righe 10-12 scegliamo a caso dove stampare un blocco e il suo colore. Nella riga 13 portiamo il cursore nella posizione dove vogliamo stampare il blocco e nella riga 14 stampiamo il blocco (nelle virgolette sono 3 spazi), del colore specificato. Nella riga 15 mettiamo una piccola pausa per mostrare una animazione.

La riga importante è la 9. Con questa gli diciamo di ripetere questi comandi 1000 volte

Le righe 6,7,17,8 servono per inizializzare e terminare lo schermo.

Le righe 1,2,3,4 indicano quali moduli di programma utilizzeremo.

Sostituendo la riga 15 con queste righe.

```
15.         screen.cursorTo(colonna*3, riga)        # portiamo il cursore nella riga e colonna
16.         time.sleep(0.1)                         # piccola pausa di 100 millisecondi
17.         print(color("   ", bg=0), end="")       # cancelliamo il blocco precedente
```

otterremo una simpatica animazione dove prima far comparire il blocco successivo, il precedente scompare

Variazioni sul tema

E' anche possibile mettere due cicli uno dentro l'altro.

Sostituiamo le righe l'intero ciclo con questo

```
 9.   for riga in range(0,11):          # ripetiamo il ciclo per ogni riga
10.       for colonna in range(0,31):           # ripetiamo il ciclo per ogni colonna
11.           colore = random.randint(0, 255)       # un colore a caso
12.           screen.cursorTo(colonna*3, riga)      # portiamo il cursore in posizione
13.           print(color("   ", bg=colore), end="") # stampiamo un blocco colorato
14.           time.sleep(0.01)                      # piccola pausa di 10 millisecondi
```

Si otterrà lo stesso risultato del primo codice ma i blocchi anzichè essere stampati in modo casuale verranno stampate in modo sequenziale, una riga per volta.

Il blocco dentro la seconda for viene eseguito la prima volta con riga=0 e colonna=0, poi viene eseguito con riga=0 e colonna=1, eccetera fino a riga=0 e colonna=30; poi si ricomincia dalla prima for, così riga diventa 1 e colonna ritorna a 0, quindi il blocco viene eseguito con riga=1 e colonna=0, poi riga=1 e colonna=1, ecc, e poi si ricomincia fino a che riga=10 e colonna=30.

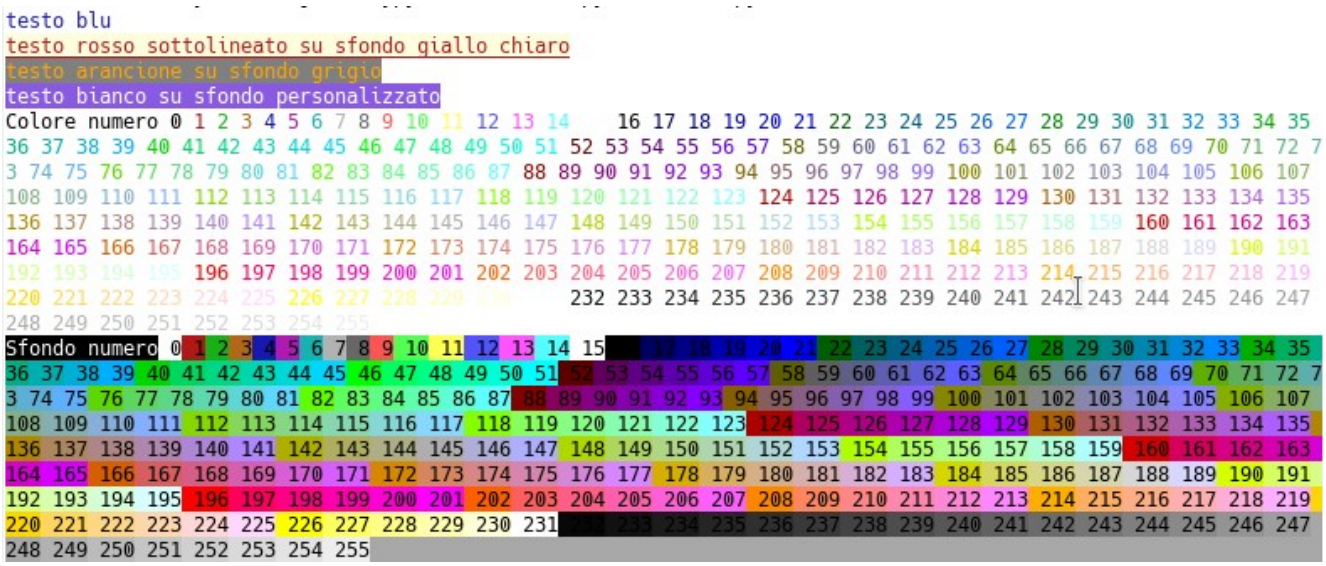
NUMERI COLORATI

Nell'esercizio precedente abbiamo utilizzato un po' di colori a caso. Possiamo utilizzare un ciclo for anche per farci mostrare quali sono tutti i colori che il sistema ci mette a disposizione

```
1.  from colors import color
2.
3.  print(color('testo blu', fg='blue'))
4.  print(color('testo rosso sottolineato su sfondo giallo chiaro',
5.              fg='red', bg='lightyellow', style='underline'))
6.  print(color('testo arancione su sfondo grigio', 'orange', 'gray'))
7.  print(color('testo bianco su sfondo personalizzato', 15, '#8a5be2'))
8.
9.  print(f"Colore numero", end="")
10. for i in range(256):
11.     print(color(f" {i}", fg=i), end="")
12.
13. print(color(f"\nSfondo numero", fg=15, bg=16), end="")
14. for i in range(256):
15.     print(color(f" {i}", bg=i), end="")
16. print("\n")
```

Qualche nota:
- 'fg' è un acronimo di 'foreground', ovvero 'colore del testo'
- 'bg' è un acronimo di 'background' ovvero 'colore dello sfondo'
- 'style' è lo stile di scrittura, in questo caso il sottolineato
- le lettere in riga 7 dopo il '#' compongono un colore; provare a cambiarle inserendo a piacere numeri da 0 a 9 e lettere da 'a' ad 'f'
- il "\n" in riga 13 e 16 significa 'vai a capo'
- il end="" dice a print di non andare a capo

Ogni colore è identificato da un numero. Con la prima for cicliamo su 256 numeri, da 0 a 255, e questa è la traduzione:



N.B.: 0 non significa nero ma un colore automatico, dipendente dal terminale.

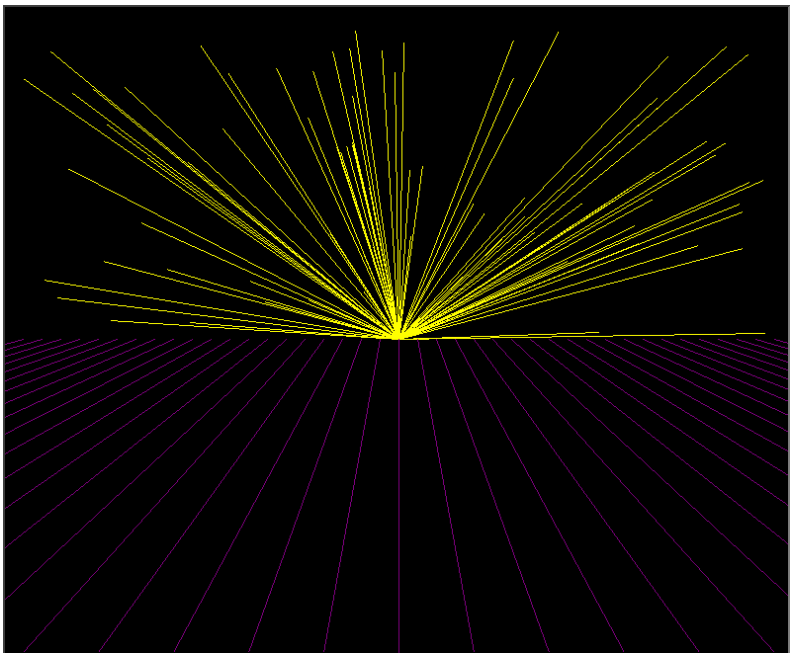
UN'AURORA MULTICOLORE

Questo programma usa un ciclo for per generare l'immagine di una aurora.

```
 1.  import draw
 2.  import random
 3.
 4.  draw.start()
 5.
 6.  # impostiamo lo sfondo nero dello schermo
 7.  draw.paper("black")
 8.
 9.  # tracciamo 80 linee casuali gialle nella parte superiore dello schermo
10.  draw.ink("yellow")
11.  for i in range(80):
12.      draw.line(0, 0, random.randint(draw.minx, draw.maxx), random.randint(0, draw.maxy))
13.
14.  # tracciamo 41 linee viola nella parte inferiore dello schermo
15.  draw.ink("purple")
16.  for x in range(draw.minx, draw.maxx + 1, (draw.maxx - draw.minx) // 40):
17.      draw.line(x, 0, x * 4, draw.miny)
18.
19.  draw.end()
```

La riga 12 tracia una linea dal centro dello schermo ad una posizione casuale nella parte superiore. Il ciclo for di riga 11 gli fa ripetere l'operazione esattamente 80 volte.

La riga 17 traccia una linea nella viola nella parte inferiore dello schermo. In un piano di solito una variabile chiamata 'x' sta ad indicare la posizione orizzontale, mentre 'y' quella verticale. Tralasciando la formula utilizzata per calcolare il punto esatto dove tracciare la linea, la for in riga 16 consente di stampare la liena 41 volte in 41 posizioni ben determinate facendo assumere ad 'x' 41 valori ben precisi che indicano la posizione, dalla parte sinistra dello schermo (primo parametro di range) alla parte destra dello schermo (secondo parametro di range), con una distanza esatta calcolata (terzo parametro di range).



Se tutto va a buon fine il risultato dovrebbe essere simile a questo:

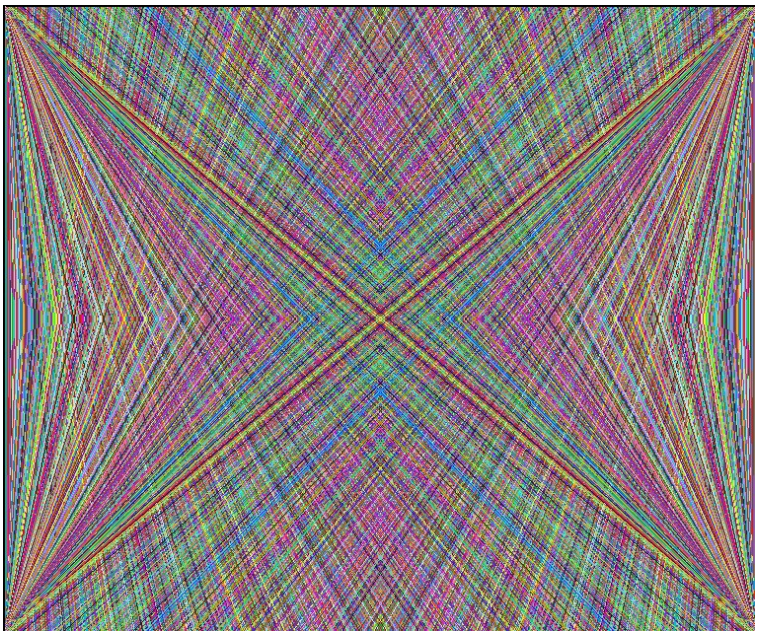
RICAMI... ISTANTANEI

Infine, ecco un programma per tirare fuori un effetto veramente spettacolare.

```
 1.  import draw
 2.  import random
 3.
 4.  draw.start()
 5.
 6.  d = {}
 7.  draw.start()
 8.  draw.paper("black")
 9.
10.  for distanza in range(20, 0, -1):
11.      for x in range(draw.minx, draw.maxx, distanza):
12.          if x in d:
13.              continue
14.          d[x] = True
15.
16.          draw.ink((random.random(), random.random(), random.random()))
17.          draw.line(draw.minx, draw.miny,  x, draw.maxy)
18.          draw.line(draw.maxx, draw.miny, -x, draw.maxy)
19.          draw.line(draw.minx, draw.maxy,  x, draw.miny)
20.          draw.line(draw.maxx, draw.maxy, -x, draw.miny)
21.
22.  draw.end()
```

Tralasciamo l'uso della variabile 'd' in riga 6,12,13,14 che è solo un escamotage per rendere più veloce il programma, le linee 16-20 disegnano 4 linee agli angoli dello schermo, sencondo un colore specificato in riga 16. Notare che a differenza dei precedenti programmi non abbiamo usato 'random.randint()' che tira fuori un numero intero a caso, ma abbiamo usato 'random.random()' che tira fuori un numero intero decimale da 0 a 1.

La variazione di posizione di queste linee è determinato dalla for in riga 11 in cui la range tira fuori tutte le posizioni orizzontali da parte a parte dello schermo con una distanza specificata nel terzo parametro, 'distanza'.



La variabile 'distanza' è variata nella for di riga 10 con la funzione range, che tira fuori tutti i numeri da 20 a 1. Il disegno delle linee avviene quindi 20 volte, ed ogni volta le linee disegnate sono più fitte, fino a riempire l'intero schermo fino ad ottenere il seguente effetto: