C O R S O D I L O G O

MANUALE DI RIFERIMENTO
ED ESERCIZIARIO, LEZIONE N.2

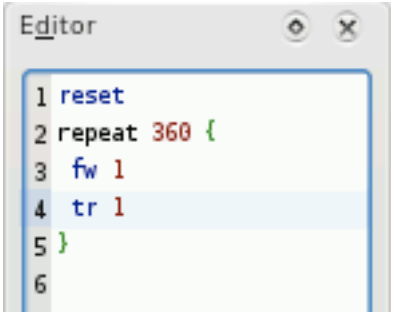
LEZIONE N.2

versione 1.0.1 18/02/2009

Lezione n.2

Facciamo un cerchio

Proviamo ora a fare un cerchio, utilizzando solo le istruzioni **forward, turnright** e **repeat** che abbiamo imparato nella lezione precedente:



Analizziamo riga per riga il programma scritto:

reset

Come al solito, cancelliamo tutto quanto fatto ad ogni nuova esecuzione

repeat 360 {

Questo comando dice alla tartaruga di ripetere per 360 volte il blocco contenuto fra parentesi graffe. Si noti come il blocco è stato spostato verso destra (con uno spazio) per migliorare la leggibilità e capire a colpo d'occhio l'inizio e la fine del blocco stesso. Questa tecnica è molto utile e si chiama *indentazione*.

forward 1

Andiamo avanti di un passettino alla volta

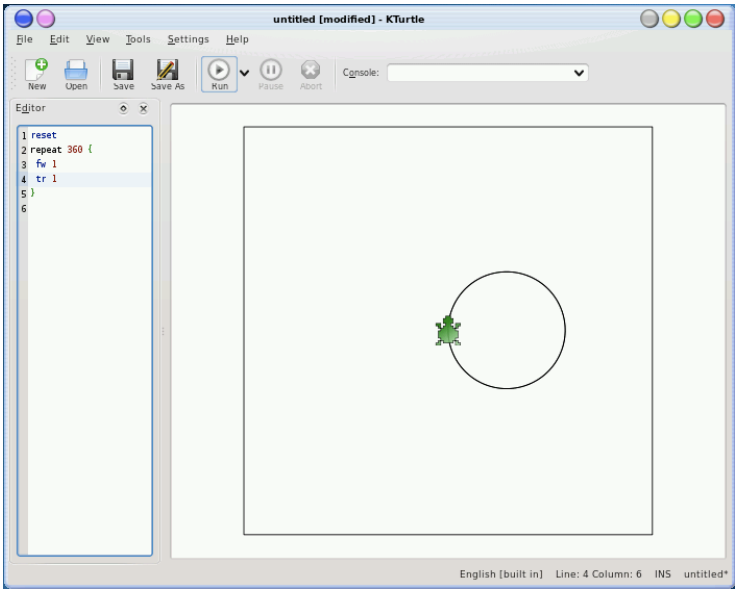
turnright 1

giriamo a destra di un solo grado. Ripetendo il tutto 360 volte, gireremo di 360° ed intanto disegneremo dei pezzettini di linea.

}

chiudiamo il blocco di istruzioni da ripetere.

Spunti e riflessioni:



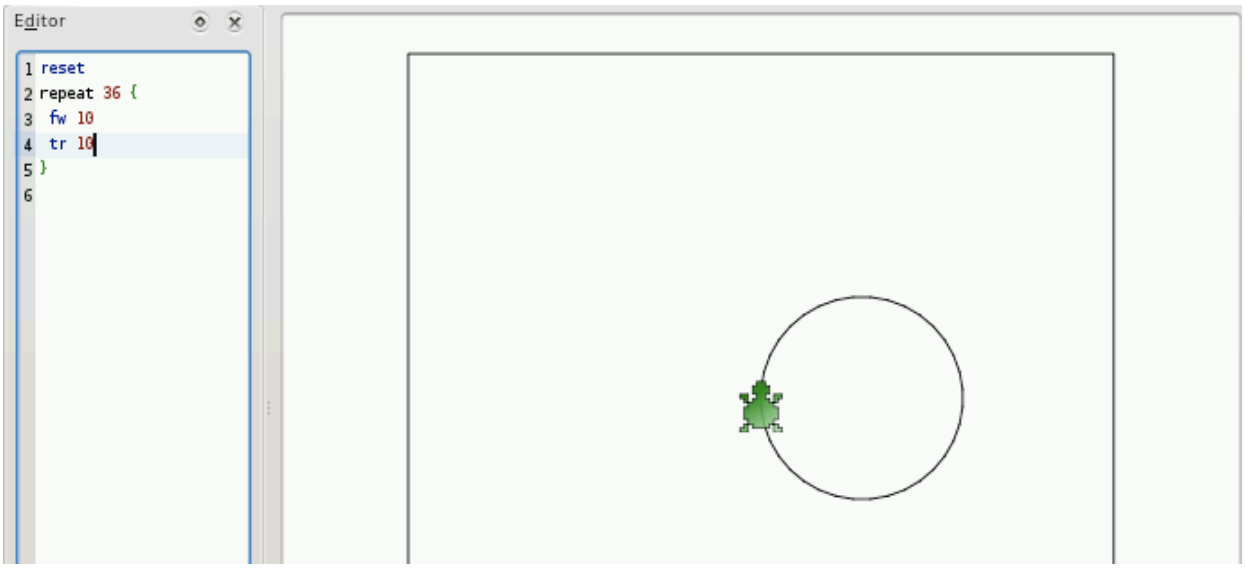
• La tartaruga fa 360 volte la stessa cosa, per cui sullo schermo si vede chiaramente la fase di disegno. Osservare che kturtle evidenzia in giallo l'istruzione che sta eseguendo, nello stesso momento in cui la sta eseguendo: questo può essere molto utile per trovare un'eventuale problema

• NON modificare il programma mentre lo si sta eseguendo, poichè potrebbe bloccarsi e dover essere riavviato. Usare prima il tasto termina per fermarlo.

• Anche qui come nei precedenti esempi, la tartaruga torna esattamente al punto di partenza.

Miglioriamo il cerchio

Il cerchio è un po' lento a disegnarsi: per velocizzare, osserviamo che forse non c'è bisogno di disegnare 360 righe ruotando di un grado. Proviamo a ripetere solo 36 volte, girando però di 10° (36 x 10 = 360) e disegnando righe leggermente più lunghe.

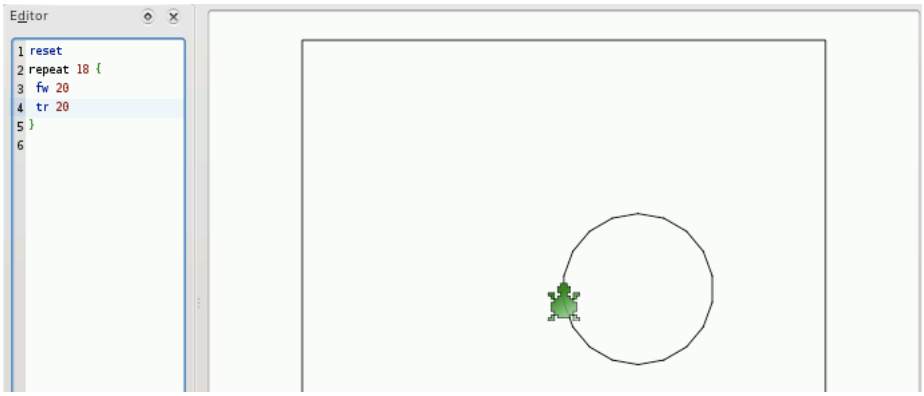


In questo modo il cerchio è molto più veloce a disegnarsi, poichè la tartaruga deve ripetere solo 36 volte il blocco di istruzioni invece di 360.

Spunti e riflessioni:

• Non abbiamo veramente disegnato un cerchio, ma un poligono di 36 lati, e prima un poligono di 360 lati. *Sembra* un cerchio, poichè i lati sono molto piccoli. In matematica si dice che abbiamo fatto un'approssimazione.

• Proviamo a sostituire 36 con 18 (meno lati) e osserviamo come ora sia necessario ruotare di 20° per disegnare un "cerchio". Con 18 lati, si dovrebbe notare molto più chiaramente che la figura non è un cerchio ma un poligono di 18 lati!.



Sperimentiamo altri valori, ricordando che l'importante è che il numero di lati moltiplicato per l'angolo faccia 360.

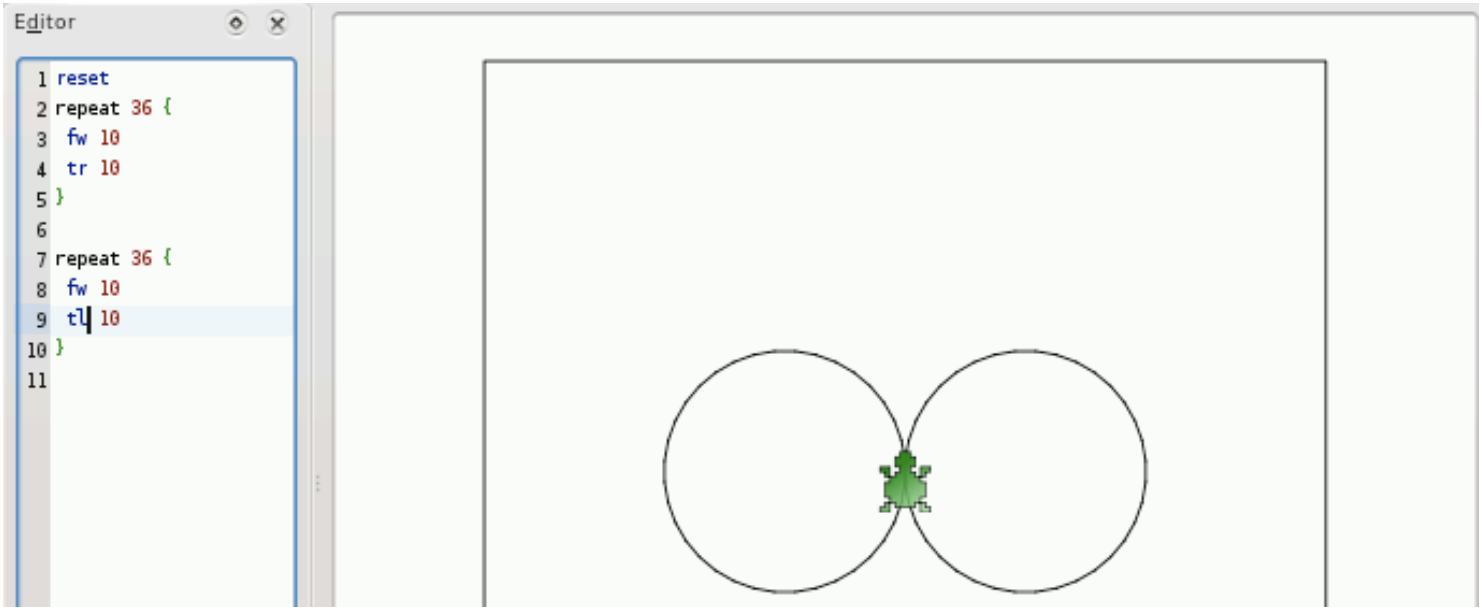
• Il parametro di forward (la lunghezza del segmento da disegnare) deve per forza essere uguale all'angolo?

• Cosa succede con 4 lati? Non è simile a qualcosa di già visto?

Disegnamo due cerchi vicini

Introduciamo una nuova istruzione, turnleft (tl).

Questa istruzione ruota a sinistra (left) invece che a destra (right). La figura ottenuta sembra una pista per automobili, oppure un otto appoggiato su un fianco. In matematica, questo 8 "sdraiato" è il simbolo dell'infinito!



Spunti e riflessioni:

- E' possibile ottenere lo stesso risultato senza usare turnleft? (suggerimento: ruotare di 180° dopo il primo cerchio)
- Notare come le seguenti istruzioni siano equivalenti, e verificarlo con la tartaruga:

ISTRUZIONE A	ISTRUZIONE B
turnright 90	turnleft 270
turnright 360	turnright 0
turnleft 180	turnright 180
turnright 45	turnleft 315

Le variabili

Le variabili sono un concetto importantissimo e potentissimo. Grazie alle variabili si possono fare disegni molto più belli e complessi.

Una variabile è praticamente una scatola che contiene un valore: ci mettiamo dentro una cosa che ci vogliamo ricordare e poi la riprendiamo da li'. In kturtle le variabili si indicano con una singola lettera preceduta dal simbolo del dollaro (che sulla tastiera si ottiene premendo shift + 4).

Ad esempio, analizziamo le seguenti due semplici istruzioni:


```
$P = 5
```

Assegna a $P il valore 5. $P è la variabile (scatola), mentre 5 è il valore che contiene.

```
forward $P
```

La tartaruga va avanti di $P pixel, ossia 5 pixel, poichè la scatola $P contiene il numero 5.

Se ora eseguiamo l'istruzione:


```
$P = 6
```

Assegneremo a $P il valore 6. La scatola è sempre la stessa ($P), il valore contenuto invece è diverso (6).

Quindi ora la stessa istruzione:

```
forward $P
```

andrà avanti di 6 pixel e non di 5.

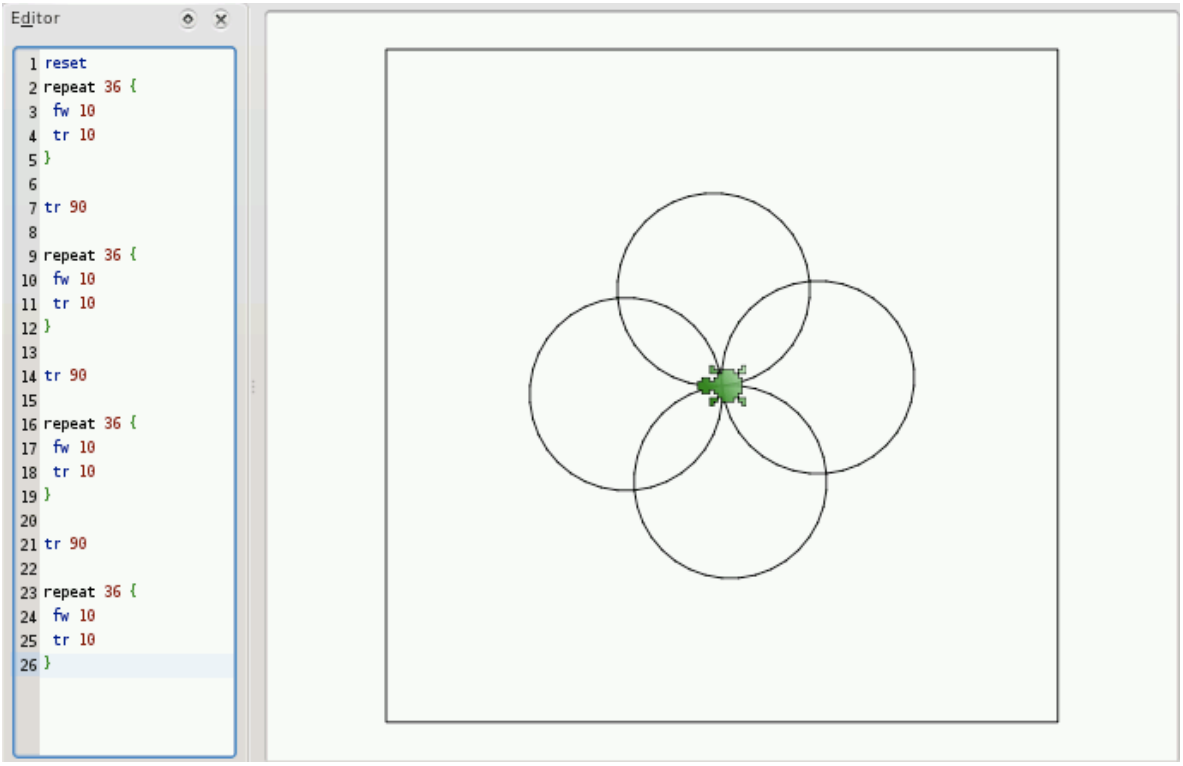
Notiamo intanto che il quadrato ed il cerchio hanno praticamente le stesse istruzioni, poichè cambiano solo i parametri utilizzati. In figura sono mostrati i due programmi l'uno vicino all'altro proprio per metterne in evidenza le similitudini. Fra poco potremo usare le variabili per fare un solo programma invece di due!

```
1 reset
2
3 repeat 4 {
4  fw 90
5  tr 90
6 }
7
```

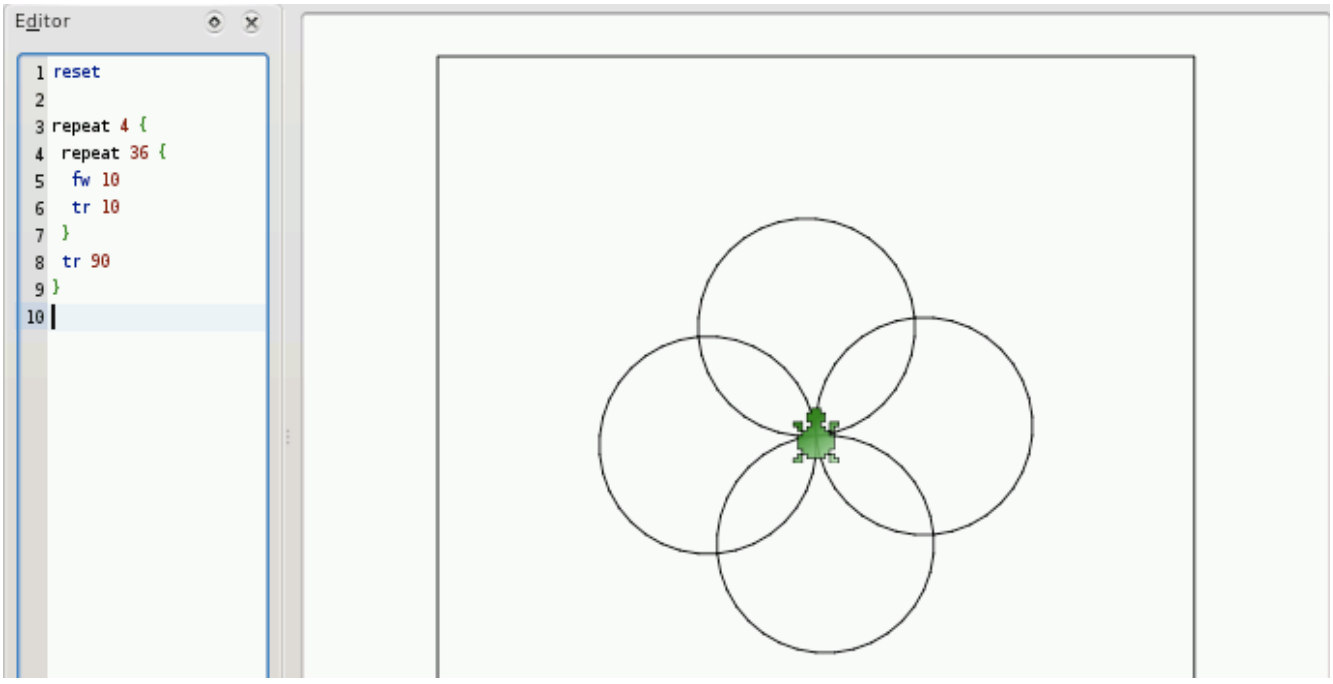
```
1 reset
2
3 repeat 36 {
4  fw 10
5  tr 10
6 }
7
```

Disegnamo un fiore

Per disegnare un disegno simile ad un fiore, possiamo ripetere più volte un cerchio (o un poligono qualsiasi), come nel seguente esempio.



E' bastato ripetere quattro volte il disegno del cerchio ruotando di 90° fra un cerchio e l'altro per ottenere una figura più complessa. Ma possiamo fare di meglio, poichè sappiamo che "ripetere" è un qualcosa che la tartaruga sa fare da sola, se opportunamente istruita, e senza bisogno di ricopiare le istruzioni!



Ripetendo 4 volte - ma stavolta con repeat - le istruzioni che disegnano un cerchio, siamo riusciti a scrivere del codice molto più breve e compatto di prima, ed il risultato è identico.

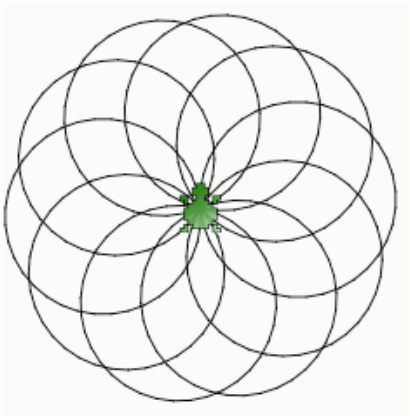
Spunti e riflessioni:

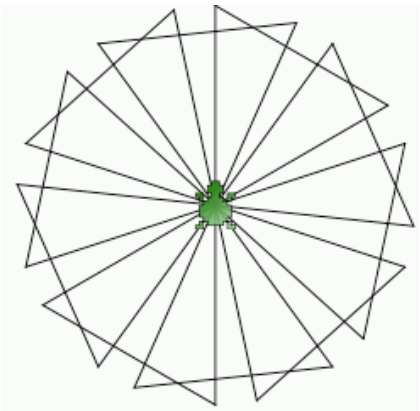
• I due cicli uno dentro l'altro sono insidiosi: attenzione alle parentesi aperte e a quelle chiuse, poichè ci si sbaglia facilmente!

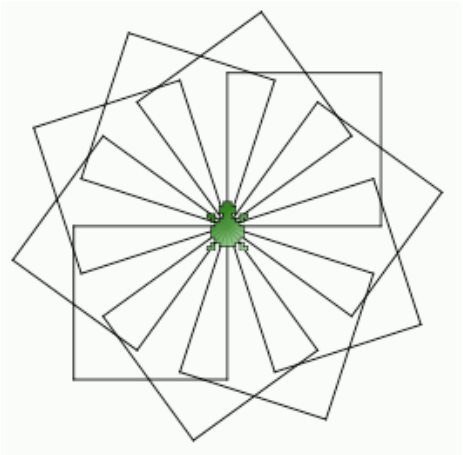
• In questi casi la corretta *indentazione* del codice aiuta moltissimo, notare come il repeat più interno è spostato verso destra con uno spazio.

• Quando un ciclo è dentro un altro ciclo, si dice che i cicli sono *nidificati*.

• Agendo sui parametri (i valori numerici), è possibile disegnare figure molto carine, alcune delle quali sono mostrate nelle immagini successive. Notare che il programma è sempre lo stesso, cambiano solo i valori numerici!







• Sperimentare autonomamente con valori diversi ed osservare i risultati. Provare ad esempio a disegnare 100 triangoli, o 30 cerchi, ruotando del giusto numero di gradi per fare un giro completo. Nota: Il numero di repeat moltiplicato per l'angolo corrispondente deve sempre fare 360°!

Insegnamo una volta per tutte!

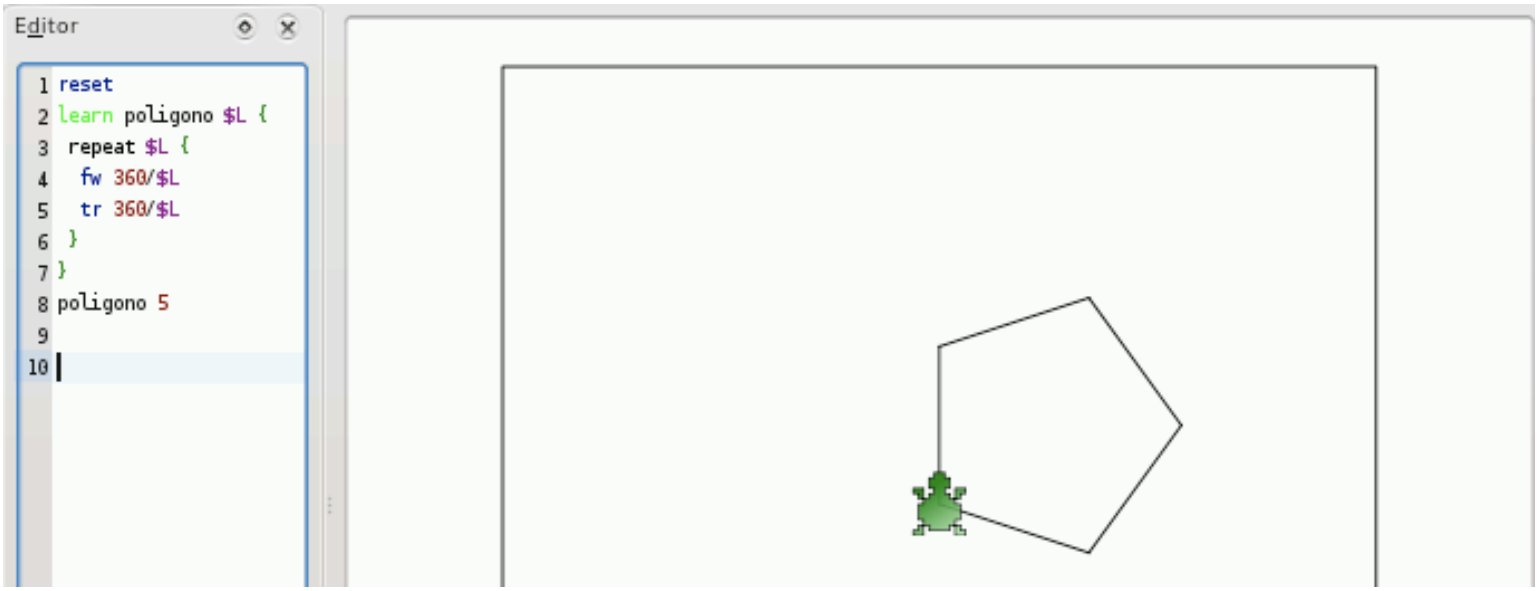
Ora siamo pronti per imparare il comando più importante di tutti, il comando learn (impara). Tramite questo comando e le variabili che abbiamo appena studiato è possibile migliorare ancora i programmi appena scritti.

Con il comando learn possiamo raggruppare delle istruzioni e dargli un nome (in informatica questa cosa si chiama procedura, o funzione) in modo da poter richiamare quelle istruzioni in qualsiasi punto senza doverle ripetere ogni volta. Un esempio chiarirà meglio:

```
1 reset
2 learn quadrato {
3  repeat 4 {
4   fw 90
5   tr 90
6  }
7 }
8 quadrato
```

In questo modo insegnamo alla tartaruga cos'è un quadrato. A questo punto possiamo scrivere direttamente quadrato, come se fosse una nuova istruzione!

Ma è combinando learn con l'uso delle variabili che si ottiene il massimo.



Con questo programma possiamo disegnare un triangolo semplicemente scrivendo poligono 3, oppure un quadrato scrivendo poligono 4, o anche un "cerchio" scrivendo poligono 40. Notare come la variabile $L, che rappresenta il numero dei lati, sia usata anche per calcolare di quanto la tartaruga deve ruotare per fare un giro completo.

Se infatti l'angolo moltiplicato per il numero dei lati deve essere 360, vuol dire che 360 diviso per il numero dei lati darà come risultato l'angolo! Analizziamo il codice riga per riga:

learn poligono $L {

Impara una uova istruzione, poligono, che accetta un parametro, il cui valore assegneremo alla variabile $L

```
  repeat $L
```

Ripeti per $L volte, ossia il numero dei lati del poligono

```
    fw 360/$L
    tr 360/$L
```

Ruota di un angolo pari a 360° diviso per il numero dei lati. Il carattere / si ottiene con shift + 7 ed indica al computer di fare la divisione.

```
  }
```

Chiudi il ciclo repeat

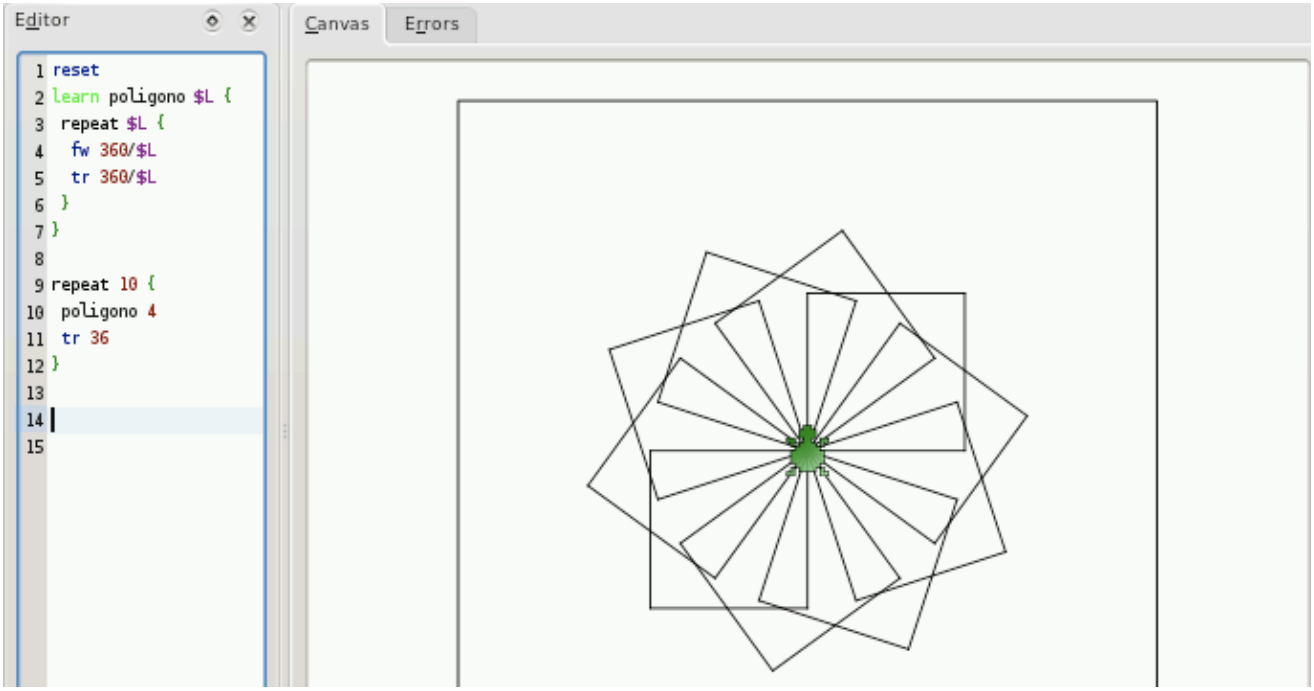
```
}
```

Chiudi l'istruzione learn

```
poligono 5
```

Disegan un poligono di 5 lati, ossia un pentagono.

A questo punto è possibile ottenere disegni come il fiore molto più semplicemente, senza *nidificare* i cicli



Questo programma è molto più leggibile, e si può scomporre in due sezioni diverse. Nella prima insegnamo alla tartaruga a disegnare un poligono, ed il codice è identico all'esempio precedente e dunque non necessita di ulteriori spiegazioni.

Nella seconda invece effettuiamo una rotazione di 360° mentre disegnamo dei poligoni:

```
repeat 10 {
```

Ripeti per 10 volte

```
poligono 4
```

Disegna un poligono regolare di 4 lati, ossia un quadrato

```
tr 36
```

Ruota di 36° (36 per 10 fa 360°)

```
}
```

Chiudi il ciclo.

Spunti e riflessioni:

• Avere insegnato a fare i poligoni una volta per tutte vuol dire poter riusare il codice senza doverlo ricopiare ogni volta, il che implica anche che eventuali errori si devono correggere in un punto solo e non in tutti i punti in cui abbiamo lo stesso codice duplicato.

• I cicli non sono più nidificati, ma continuano ad esistere tutti e due, separati. Ora il codice è semplicemente più semplice da leggere, ma come numero di cicli eseguiti è equivalente.

Esercizi

Ispirandosi alle figure mostrate in basso:

- Disegnare dei quadrati sempre più grandi, mantenendo ferma l'origine
- Come nell'esercizio precedente, ma ruotando ad ogni iterazione