Clase 5 – Introducción a Ciclos con Turtle

Módulo: Ciclos + Turtle

Fundamentación

Los ciclos permiten repetir un bloque de instrucciones sin escribirlo varias veces, lo que ahorra tiempo y reduce errores. Integrar el módulo Turtle permite visualizar de forma inmediata el efecto de los ciclos, reforzando la comprensión con resultados gráficos atractivos.

Objetivo general

- Comprender y aplicar la estructura `for` en Python para realizar repeticiones controladas, utilizando Turtle para representar visualmente las iteraciones.

Objetivos particulares

1. Diferenciar entre repetición manual y repetición automática mediante ciclos.
2. Comprender la sintaxis básica de `for` con la función `range()`.
3. Usar Turtle para generar figuras repetitivas (líneas, polígonos simples).
4. Modificar parámetros del ciclo para explorar variaciones.

Contenidos

- **Conceptuales**
  - Estructura `for`
  - Función `range(inicio, fin, paso)`
  - Relación entre número de iteraciones y resultado gráfico
- **Procedimentales**
  - Escribir bucles `for` con salida en Turtle
  - Cambiar número de repeticiones para alterar el dibujo

    - Usar ángulos y movimientos repetidos
- **Actitudinales**
    - Fomentar la experimentación
    - Desarrollar observación del patrón y predicción del resultado

Desarrollo de la clase

El bucle for

En general, un bucle es una estructura de control que repite un bloque de instrucciones. Un bucle for es un bucle que repite el bloque de instrucciones un número prederminado de veces. El bloque de instrucciones que se repite se suele llamar cuerpo del bucle y cada repetición se suele llamar iteración.

```
for i in range(0, 5):
    print(i)

# Salida:
# 0
# 1
# 2
# 3
# 4
```

En Python se puede iterar prácticamente todo, como por ejemplo una cadena. En el siguiente ejemplo vemos como la `i` va tomando los valores de cada letra. Mas adelante explicaremos que es esto de los **iterables** e **iteradores**.

```
for i in "Python":
    print(i)

# Salida:
# P
# y
# t
# h
# o
# n
```

Gráficos: el módulo turtle

https://inventwithpython.com/stt/#Introduction

La "tortuga" de Logo es un cursor al que se le pueden dar órdenes de movimiento (avance, retroceso o giro) y que puede ir dejando un rastro sobre la pantalla. Moviendo adecuadamente la tortuga se pueden conseguir dibujar todo tipo de figuras.

Python incluye un módulo llamado "turtle" que permite crear gráficos de tortuga.

Para utilizar el módulo turtle sólo hace falta importarlo:

```
import turtle
```

La tortuga en Python siempre empieza “mirando” hacia la derecha y, a medida que camina, deja una línea dibujada. Si se la hace caminar hacia adelante, empezará haciendo una línea de izquierda a derecha.

Algunas órdenes básicas de Turtle (hay muchas más):

forward(x)	Caminar hacia adelante (la cantidad de pixeles indicada en el número x)
backward(x)	Caminar hacia atrás (la cantidad de pixeles indicada en el número x)
right(x)	Girar la tortuga hacia la derecha en un ángulo de x grados
left(x)	Girar la tortuga hacia la izquierda en un ángulo de x grados
begin_fill()	Las figuras cerradas dibujadas a continuación serán pintadas con un color x especificado por fillcolor(x)
end_fill()	Pinta todas las figuras cerradas que se hayan dibujado a partir de donde dice begin_fill() y hasta donde se coloca end_fill()
fillcolor(x)	Rellena, con el color indicado en el string x, la figura dibujada por las instrucciones incluidas entre begin_fill() y end_fill() Algunos ejemplos de colores: 'red', 'green', 'yellow' (deben escribirse entre comillas)
pencolor(x)	Indica que, todo lo que se dibuje, en adelante será del color indicado por el string x
circle(x)	Dibuja un círculo del radio indicado por el número x

Ejemplos:

Veamos algunos ejemplos de código para dibujar formas geométricas. Para interpretar código, una buena idea es seguirlo en papel, respetando las instrucciones en el orden en que están escritas.

Ejemplo 1:

```
from turtle import *
forward(50)
left(90)
forward(50)
left(90)
forward(50)
left(90)
forward(50)
left(90)
```

Esto dibuja un cuadrado, porque hace 4 veces lo mismo: dibuja una línea recta y gira a la izquierda en 90º. Como esas dos órdenes se repiten 4 veces, lo mejor es escribirlo con alguna herramienta que permita repetir algo. Por ejemplo, un for que itere 4 veces:

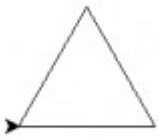
```
from turtle import *
for i in range(4):
    forward(50)
    left(90)
```

Ejemplo 2:

En este caso dibuja un triángulo, entonces las instrucciones se repiten tres veces y, en vez de girar 90º, gira 120º:

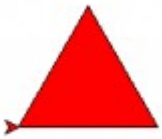
```
from turtle import *
for i in range(3):
    forward(100)
    left(120)
```

(el ángulo de 120 es porque se dividen los 360 grados totales por la cantidad de lados que tiene la figura a formar: 360/3=120).



Para hacer el mismo triángulo, pero relleno de color rojo:

```
from turtle import *
fillcolor('red')
begin_fill()
for i in range(3):
    forward(100)
    left(120)
end_fill()
```



Ejemplo 3:

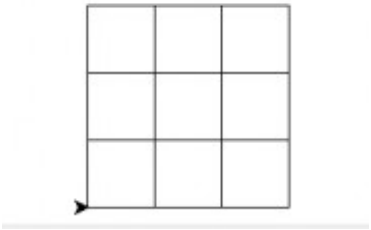
Para hacer una estrella:

```
from turtle import *
for i in range(5):
    forward(100)
    right(144)
```

Ejemplo 4:

Para dibujar una cuadrícula de 3x3 habrá que usar iteraciones anidadas: por un lado, la iteración más interna que dibuja un cuadrado (la misma vista en ejemplo 1), por otro lado la que forma las columnas y por otro la que forma las filas:

```
from turtle import *
for fila in range(3):   #qué quiero hacer en cada fila
    for columna in range(3):   #qué quiero hacer en cada columna
        for cuadrado in range(4):   #dibujo un cuadrado
            forward(50)
            right(90)
        forward(50)   #para no empezar otra vez en el mismo punto al terminar una fila, me corro a la
derecha en el ancho de 1 cuadradito
    backward(50*3)   #vuelvo 3 cuadraditos para atrás (cada uno mide 50 de ancho) para quedar nuevame
nte en la primera columna
    right(90)
    forward(50)
    left(90)   #con las últimas 3 líneas me vuelvo a posicionar para empezar a dibujar la siguiente f
ila
```



Si quisiera que cada cuadrado quede pintado de un color, alternando entre 2 colores, entonces tengo que usar alguna estrategia: por ejemplo, empiezo contando del 1 en adelante cada cuadrado que dibujo; si el número es par pinto de un color y si es impar pinto de otro. Para contar los cuadrados uso una variable:

PYTHON INICIAL

```
from turtle import *
recuadro = 1   #número de cuadrado
for fila in range(3):
    for columna in range(3):
        begin_fill()   #lo que quiero pintar es lo que se dibuja a continuación: un cuadrado
        for cuadrado in range(4):
            forward(50)
            right(90)
        if recuadro % 2 == 0:   #si el número es par
                fillcolor('red')   #pinto de rojo
        else:   #si el número no es par
                fillcolor('green')  #pinto de verde
        end_fill()   #terminé de dibujar un cuadrado, entonces lo pinto
            recuadro += 1;   #paso a contabilizar el siguiente cuadrado
            forward(50)
    backward(150)
    right(90)
    forward(50)
    left(90)
```