INTRODUCCIÓN A PYTHON

E4

Contenido

“

Este espacio curricular permite fortalecer el espíritu crítico y la actitud creativa del futuro graduado. Lo alienta a integrar los conocimientos previos, recreando la práctica en el campo real, al desarrollar un proyecto propio.

”

Encuentro 4

Los lenguajes de programación tienen 3 grupos de instrucciones; secuenciales, decisión y repetición. En este capitulo abordaremos el uso de las instrucciones de repetición y los tipos de datos compuestos.

¡¡Así que... A disfrutar... Que, al terminar el cuatrimestre, ¡¡estaremos jugando con nuestros propios juegos!!

Introducción

En esta clase, vamos a trabajar con tipos de datos complejos y con las estructuras de repetición. El uso de estructuras de repetición es útil a la hora de desarrollar juegos, por ejemplo, cuando decimos para cada elemento del juego, comprobar si fueron alcanzados por una bala. También, puede usarse para administrar el ciclo de un juego. En teste caso, cuando decimos mientras no presione la tecla escape… sigue jugando. En el caso de Python tenemos solo dos tipos de bucles, los cuales veremos a continuación.

La instrucción For

Se trata de una instrucción que repite una línea o un conjunto de líneas (bloque). La característica es que, las repite una cantidad finita de veces. Aunque, existen formas para cortar el bucle de repetición, no lo usaremos en nuestro código fuente. Esto último, debido a que se considera una mala práctica de programación.

Esta instrucción, necesita que se le indique cuantas veces se repite el código. La cantidad se representa por una variable que oficia de iterador, y un rango de valores por recorrer. La sintaxis de la instrucción sería algo así:

```
for [iterador] in [rango]:
    bloque
```

el iterador es una variable que representa, en cada vuelta, el elemento que forma parte de la lista del rango expresado. El ciclo termina cuando no quedan elementos en la lista.

En ejemplo simple de un for, sería uno que muestre los valores del 0 al 5. El código se puede ver a continuación:

```
for i in range(6):
    print(i)
```

el resultado se ve en la siguiente imagen:

```
0
1
2
3
4
5
PS D:\salaPython>
```

En este ejemplo podemos ver que i es la variable del iterador, y que range(6) es una instrucción de Python para crear una lista de 6 números comenzando en 0.

La instrucción range tiene varias firmas, que me permiten ser mas preciso en la creación de la lista de valores. Por ejemplo, podemos indicar en que valor comienza la lista y donde termina.

```
for i in range(3,6):
    print(i)
```

Esto último no es lo único que me permite hacer la instrucción range. Por ejemplo, puedo indicar el valor de salto, o en qué sentido (ascendente o descendente) se debe crear la secuencia.

Con saltos de 2

```
#Completa el ciclo con saltos de 2
for i in range(0,6,2):
    print(i)
```

En forma descendente

```
#Completa el ciclo en forma descendente

for i in range(6,0,-1):

    print(i)
```

Pero, esto no es lo único. La instrucción for, recorre los elementos de una lista. Por lo tanto, me puedo aprovechar de esto, para recorrer las letras de una palabra.

```
for i in "Metegol":
    print(i,end='-')
```

este Código nos imprime a la salida

```
M-e-t-e-g-o-l-
```

Otra forma, tal vez un poco menos Python 😊 sería:

```
palabra="El cielo esta oscuro"


for i in range(21):

    print (palabra[i])
```

En este caso, recorremos la cadena de la variable palabra, una letra por cada vuelta.

Pero eso no es todo, todavía tenemos más para agregar a la instrucción for. Por ejemplo, podemos utilizar la instrucción else.

```
for i in "Metegol":

    print(i,end='-')

else:

    print("\nTerminó la palabra")
```

Instrucciones for anidadas

Syntaxis general del while

```
while [condicion]:

    bloque
```

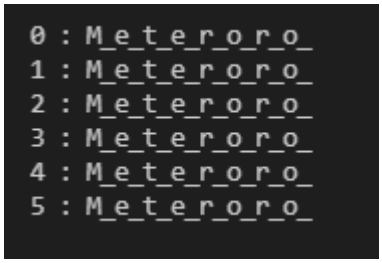
Al igual que el if, podemos programar un for dentro de un for. Considera el siguiente ejemplo que usa el ciclo for anidado

```
4  for i in range(6):
5      print(i, end=' : ')
6      for z in "Meteroro":
7          print(z, end='_')
8      print()
9
```

Analicemos cada una de las líneas:

4- comienza el primer for, que recorre una lista de números del 0 al 5
5- este print muestra el iterador i y se encuentra dentro del primer for
6- comienza el segundo for, con un iterador z que recorre todas las letras de la palabra
7- Muestra el iterador z, este se encuentra dentro del segundo for
8. hace un salto de línea, este print se encentra dentro del primero for.

El resultado de este código es que por cada vuelta del iterador i, el iterador z recorre toda la palabra.



Instrucción while

El ciclo while se usa generalmente cuando se necesita repetir un bloque de código, pero con un número desconocido de repeticiones. Por ejemplo, en un juego de dados, el programa le puede preguntar al jugador cuantos dados desea cambiar.

Veamos un ejemplo donde la cantidad de veces que se dice hola depende de un numero que ingresa el usuario.

```
1 cantidad=int(input("¿Cuántas veces digo hola? "))
2 contador=0
3 while contador<cantidad:
4     print ("Hola")
5     contador+=1
```

el código es muy sencillo, pero analicemos el código. Porque aquí tenemos una desventaja con respecto al for, no tenemos control automático de la cantidad que el ciclo repite.

La línea 1: pedimos el dato por teclado y lo asignamos en la variable cantidad

Línea 2: declaramos una variable llamada contador. Su función será contar la cantidad de vueltas que repite la instrucción while

Línea 3: es el comienzo del while. Está formado por una condición. Las condiciones se arman igual que con el if. En este caso, si la condición es verdadera (True) el ciclo se repite.

La línea 4 y 5 son parte del bloque que repite del while

Línea 4: muestra la palabra hola en pantalla

Línea 5: le incrementamos uno al contador

(*contador = contador +1*). Esta es la parte mas importante. La variable contadora, está contando las vueltas. Si esto esta mal programado, el ciclo puede ser infinito.

No siempre debemos tener un contador en el while, también podemos usar variables booleanas. Esta estrategia es muy útil para determinar el ciclo de ejecución de un juego.

```
salir=False
while not salir:
    [codigo del juego]
    op=input("Desea salir del juego? (S/N) :")
    if op=='S':
        salir=True
```

Bibliografía

Sweigart, A. (2012). Making Games with Python & Pygame: A Guide to Programming with Graphics, Animation, and Sound. Recuperado de https://inventwithpython.com/makinggames.pdf

Pygame. (s.f.). [Página oficial de pygame]. Recuperado 2 julio, 2019, de https://www.pygame.org/news