Fundamentos de Programación

ESTRUCTURA DE ITERACIÓN 1
Semana 07

OBJETIVO DEL LABORATORIO

Utiliza los procesos repetitivos con el código DO / WHILE (Hacer – Mientras).

MARCO TEÓRICO

Completaremos los 3 tipos de algoritmos, viendo los de tipo repetitivo; donde su construcción incluye los secuenciales y condicionales; donde se repetirá una acción "n" veces en base a una situación en particular.

Como fuente de información, se tiene la misma plataforma del Netacad. Tanto para realizar los ejemplos propuestos en la interfaz integrada, así como para repasar lo visto en clase.

RECURSOS

a. Hardware
- Pc Pentium IV a superior
- Conexión de red

b. Software
- Sistema Operativo Windows XP a superior
- Navegador Chrome o Firefox
- Edube Sandbox de Python desde Netacad

PROCEDIMIENTO

1. EL BUCLE while

   Un bucle while permite repetir la ejecución de un grupo de instrucciones mientras se cumpla una condición (es decir, mientras la condición tenga el valor True).

   La sintaxis del bucle while es la siguiente:

```
while condicion:
    cuerpo del bucle
```

   Cuando llega a un bucle while, Python evalúa la condición y, si es cierta, ejecuta el cuerpo del bucle. Una vez ejecutado el cuerpo del bucle, se repite el proceso (se evalúa de nuevo la condición y, si es cierta, se ejecuta de nuevo el cuerpo del bucle) una y otra vez mientras la condición sea cierta. Únicamente cuando la condición sea falsa, el cuerpo del bucle no se ejecutará y continuará la ejecución del resto del programa.

   La variable o las variables que aparezcan en la condición se suelen llamar variables de control. Las variables de control deben definirse antes del bucle while y modificarse en el bucle while.

   Por ejemplo, el siguiente programa escribe los números del 1 al 3:

```
Ejemplo de bucle while 1
i = 1
```

```
while i <= 3:
    print(i)
    i += 1
print("Programa terminado")
```

RESULTADO:

```
1
2
3
Programa terminado
```

El ejemplo anterior se podría haber programado con un bucle for. La ventaja de un bucle while es que la variable de control se puede modificar con mayor flexibilidad, como en el ejemplo siguiente:

Ejemplo de bucle while 2

```
i = 1
while i <= 50:
    print(i)
    i = 3*i + 1
print("Programa terminado")
```

RESULTADO:

```
1
4
13
40
Programa terminado
```

Otra ventaja del bucle while es que el número de iteraciones no está definido antes de empezar el bucle, porque los datos los proporciona el usuario. El siguiente ejemplo pide un número positivo al usuario una y otra vez hasta que el usuario lo haga correctamente:

Ejemplo de bucle while 3

```
numero = int(input("Escriba un número positivo: "))
while numero < 0:
    print("¡Ha escrito un número negativo! Inténtelo de nuevo")
    numero = int(input("Escriba un número positivo: "))
print("Gracias por su colaboración")
```

RESULTADO:

```
Escriba un número positivo: -4
¡Ha escrito un número negativo! Inténtelo de nuevo
Escriba un número positivo: -8
¡Ha escrito un número negativo! Inténtelo de nuevo
Escriba un número positivo: 9
Gracias por su colaboración
```

2. BUCLES INFINITOS

Si la condición del bucle se cumple siempre, el bucle no terminará nunca de ejecutarse y tendremos lo que se denomina un bucle infinito. Aunque a veces es necesario utilizar bucles infinitos en un programa, normalmente se deben a errores que se deben corregir.

Los bucles infinitos no intencionados deben evitarse pues significan perder el control del programa. Para interrumpir un bucle infinito, hay que pulsar la combinación de teclas Ctrl + C. Al interrumpir un programa se mostrará un mensaje de error similar a éste:

```
Traceback (most recent call last):
  File "ejemplo.py", line 3, in <module>
    print(i)
KeyboardInterrupt
```

Por desgracia, es fácil programar involuntariamente un bucle infinito, por lo que es inevitable hacerlo de vez en cuando, sobre todo cuando se está aprendiendo a programar.

Estos algunos ejemplos de bucles infinitos:

- El programador ha olvidado modificar la variable de control dentro del bucle y el programa imprimirá números 1 indefinidamente:

```
i = 1
while i <= 10:
    print(i, end=" ")
```

RESULTADO:

```
1 1 1 1 1 1 1 1 ...
```

- El programador ha escrito una condición que se cumplirá siempre y el programa imprimirá números consecutivos indefinidamente:

```
i = 1
while i > 0:
    print(i, end=" ")
    i += 1
```

RESULTADO:

```
1 2 3 4 5 6 7 8 9 10 11 ...
```

- Se aconseja expresar las condiciones como desigualdades en vez de comparar valores. En el ejemplo siguiente, el programador ha escrito una condición que se cumplirá siempre y el programa imprimirá números consecutivos indefinidamente:

```
i = 1
while i != 100:
    print(i, end=" ")
    i += 2
```

RESULTADO:

```
1 3 5 7 9 11 ... 97 99 101 ...
```

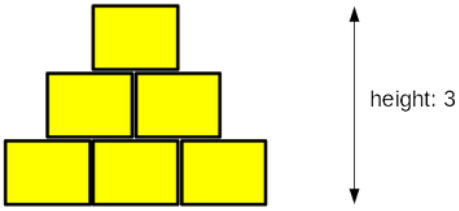
3. LAUNCH IN EDUBE

EJERCICIO 1:

- usando el bucle while;
- encontrar la correcta implementación de reglas verbalmente definidas;
- reflejando situaciones de la vida real en código informático.

Escucha esta historia: un niño y su padre, un programador de computadoras, juegan con bloques de madera. Están construyendo una pirámide. Su pirámide es un poco rara, ya que en realidad es una pared en forma de pirámide, es plana. La pirámide se apila de acuerdo con un principio simple: cada capa inferior contiene un bloque más que la capa superior.

La figura ilustra la regla utilizada por los constructores.



blocks: 6

Su tarea es escribir un programa que lea la cantidad de bloques que tienen los constructores, y genera la altura de la pirámide que puede construirse utilizando estos bloques.

Nota: la altura se mide por el número de capas completamente completadas: si los constructores no tienen un número suficiente de bloques y no pueden completar la siguiente capa, terminan su trabajo de inmediato.

Pruebe su código utilizando los datos que hemos proporcionado.

```
# Sample Solution

bloques = int(input("Introduzca el número de bloques: "))
altura = 0
inlayer = 1
while inlayer <= bloques:
      altura += 1
      bloques -= inlayer
      inlayer += 1

print("Altura de la pirámide:",height)
```

EJERCICIO 2:

- usando el bucle while;
- convirtiendo bucles verbalmente definidos en código Python real.

En 1937, un matemático alemán llamado Lothar Collatz formuló una hipótesis intrigante (aún no se ha comprobado) que se puede describir de la siguiente manera:

1. tome cualquier número entero que no sea negativo y que no sea cero y asígnele el nombre c0;
2. si es par, evalúa un nuevo c0 como c0 ÷ 2;
3. de lo contrario, si es impar, evalúe un nuevo c0 como 3 × c0 + 1;
4. si c0 ≠ 1, salta al punto 2.

La hipótesis dice que, independientemente del valor inicial de c0, siempre irá a 1.

Por supuesto, es una tarea extremadamente compleja usar una computadora para probar la hipótesis de cualquier número natural (incluso puede necesitar inteligencia artificial), pero puede usar Python para verificar algunos números individuales. Tal vez incluso encuentres el que refutaría la hipótesis.

Escriba un programa que lea un número natural y ejecute los pasos anteriores siempre que c0 sea diferente de 1. Además, agregaremos otra tarea: queremos que cuente los pasos necesarios para lograr el objetivo. Tu código también debe mostrar todos los valores intermedios de c0.
Sugerencia: la parte más importante del problema es cómo transformar la idea de Collatz en un tiempo de bucle - esta es la clave del éxito.

Pruebe su código utilizando los datos que hemos proporcionado.

```
# Sample solution

c0 = int(input("Ingrese c0: "))
if c0 > 1:
      pasos = 0
      while c0 != 1:
            if c0 %2 != 0:
                  cnew = 3 * c0 + 1
            else:
                  cnew = c0 // 2
            print(c0)
            c0 = cnew
            pasos += 1
      print("pasos =",pasos)
else:
      print("Valor de c0 incorrecto")
```

CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA

- Al momento de construir el bucle repetitivo, hay que verificar dónde se colocan las sentencias porque si no, tendremos un cálculo diferente o incluso un error.
- Se recomienda usar el Edube SandBox para realizar nuestros ejercicios sin necesidad de instalar algún software adicional.
- Así como en algún momento se solicita cálculos con números enteros, sería factible buscar realizar lo mismo; pero con números reales.

ACTIVIDAD VIRTUAL

1. Observa y analiza el siguiente video: https://www.youtube.com/watch?v=jk4hN6pef64, y responde las siguientes preguntas:

- ¿En qué momento se detiene el bucle?
- Realiza ese algoritmo según los datos proporcionados. (Desarróllalo en Python)