PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC1103- INTRODUCCIÓN A LA PROGRAMACIÓN

Laboratorio 3
Control de flujo: Ciclos `while`

Segundo semestre, 2015

Objetivo

Este laboratorio es para que practiques con estructuras de control de flujo para producir ciclos en el programa, usando `while`, reforzando además lo visto anteriormente. Además, se incluye la primera parte de la creación de números aleatorios.

1. Ciclos `while`

Como habrás visto en clases, en muchos programas es necesario ejecutar una acción en forma repetida. Una de las instrucciones que dispone Python para realizar esta tarea es `while`. El siguiente código muestra la estructura del lenguaje usando `while`:

```
while <condicion> :
    <codigo que se ejecuta mientras se cumple condicion>
```

La condición debe generar un valor booleano `True` o `False`. Cuando es `True`, se logra que el programa entre al código en la línea siguiente. Al final del código que se ejecuta "dentro" del `while`, se vuelve a evaluar la condición (en el ejemplo, `condicion`), y si es verdadera (`True`) el programa vuelve a ejecutar el código dentro del ciclo `while`. Dentro del código de un ciclo `while`, pueden usar todas las estructuras vistas hasta ahora, como variables, operadores, controles de flujo `if/elif/else`, e incluso otros ciclos `while`.

Recuerda que en tus programas en Python, debes respetar la estructura del lenguaje, como:

- Usar dos puntos (:) después de cada condición que viene después de `if` y `elif`, y después de `else`. Ahora, tambien se debe usar después de la condición después de `while`.
- Indentar. IDLE y pycharm, entre otros, lo hacen automáticamente después de los dos puntos (:). La regla general es que la indentación sean cuatro espacios.
- Colocar el código dentro de cada control de flujo en el mismo nivel de indentación. Recuerda que también puedes anidarlos, es decir, poner un control de flujo dentro de una parte de otro control de flujo.

Dentro de condición, puedes poner la expresión que quieras mientras genere un valor `True` o `False`, o también podrá tomarse como `True` cualquier valor no nulo (distinto de cero), mientras que como `False` también podrá interpretarse un valor cero (0).

1.1. Ejemplo resuelto

Escriba un programa que imprima en pantalla los múltiplos de 3 entre el 1 y el 100 que además no sean múltiplos de 9.

Solución

El siguiente programa resuelve lo pedido. Notar que es **muy importante** la instrucción final para incrementar `numero` en 1; si no estuviera esta instrucción, el programa no podría salir del ciclo ya que siempre la condición `numero <= 100` sería verdadera.

```
numero = 1

while numero <= 100 :
    if numero%3 == 0 and numero%9 != 0 :
            print("Encontrado nuevo numero:",numero)
    numero = numero + 1
```

2. Generación de números aleatorios

Python provee de útiles funciones para el manejo de números aleatorios (al azar). Hoy practicaremos con una función que genera números aleatorios enteros `int`. Para poder usar las funciones, primero debes ejecutar `import random`, que es una instrucción que carga las funciones de la librería `random`. Puedes pensar en una librería como un conjunto de funciones que te permiten extender las funciones básicas de Python (como `input`, `print` o `str`, que también son funciones).

Para obtener números aleatorios enteros entre los números `i` y `j`, dispones de la función `random.randint(i,j)`. Ejecuta el siguiente código para que explores su funcionamiento:

```
import random
numero = 1
while numero <= 20 :
    num_azar = random.randint(1,10)
    print(num_azar, end=" ")
    numero = numero + 1
```

El programa te imprimirá en pantalla una lista de 20 números entre 1 y 10, separados por un espacio. El argumento `end` de la función `print` te permite Ejecuta el código varias veces, y nota que en cada vez cambian los números generados. Un posible resultado de la ejecución del ejemplo sería

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

3. Ejercicios

1. Escribe un programa que simule el resultado del lanzamiento repetitivo de dos dados de seis caras, y que lance los dados hasta que:

   a) La suma de los dos dados sea 7.

   b) Se obtenga un doble 6.

   c) Se obtenga un 6 y un número impar.

   En cada caso, imprime en pantalla el número de intentos que le llevó al programa obtener el resultado deseado.

2. Los números binarios son aquellos que se representan en base 2. Por ejemplo, el número 29 es posible escribirlo como $2 \times 10^1 + 9 \times 10^0$ en base 10, el cual al representarlo en base 2 sería $1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$, por lo que el número 29 en representación binaria sería 11101. Como puedes notar, cualquier número en base n se puede escribir sólo con cifras entre 0 y $n - 1$. Escribe un programa que permita obtener la representación en número binario de un número positivo en base 10, que ingrese un usuario del programa, como el siguiente ejemplo:

   ```
   Ingrese un numero: 1234
   El número 1234 en representacion binaria es 10011010010
   ```

3. Usando números aleatorios, programe un mini-Blackjack numérico, usando números en vez de cartas, y jugado entre un usuario y el computador. El objetivo es acercarse lo más posible a 21, sin pasarse. Gana el usuario si su número es mayor que el del computador. Los pasos del juego son:

   - El usuario inicia el juego con saldo de $1000.
   - El usuario apuesta una cantidad entre 1 y 100.
   - El computador le entrega un número entre 2 y 21.
   - El usuario puede pedir otro número entre 1 y 10 para aumentar el número inicial. Esto lo puede hacer varias veces. Si se pasa de 21, se le resta el dinero de la apuesta y el juego termina.
   - Si el usuario no quiere aumentar más el número, el computador imprime su número (entre 2 y 21). Si el número es mayor que 16, el computador no puede elevar más su número. Si el número es menor o igual a 16, el computador está obligado a generar un nuevo número (entre 1 y 10) y sumarlo al anterior, y repite el proceso mientras su número sea menor o igual a 16. Si el computador se pasa de 21, entrega al usuario como premio el doble del dinero de la apuesta. Si el número del computador es mayor o igual al del usuario, gana el computador.
   - El usuario puede optar al final seguir jugando o salir del programa.

   Se muestra una posible salida del programa:

   ```
   Tu saldo es 1000. ¿Cuanto deseas apostar?: 100
   Tu apuesta es 100. Tu numero es 14.
   ¿Deseas aumentar el numero? (si / no): si
   Tu aumento es de 6. Tu nuevo numero es 20.
   ¿Deseas aumentar el numero? (si / no): no
   ```

```
El computador tiene el numero 12.
El computador aumenta el numero en 6. Ahora tiene el 18.
El computador no puede aumentar más.
Gana usuario (20>18). Su nuevo saldo es 1100.
¿Deseas continuar jugando? (si / no): no
Gracias por jugar!!!
```

4. Bonus

1. Modifica el primer ejercicio para que se ejecute varias veces seguidas, y obtén la razón (fracción) entre el número de intentos exitosos y el número de intentos totales. Verifica que estas razones son cercanas a $1/6$, $1/36$ y $1/12$, respectivamente.

2. Modifica el segundo ejercicio para que el usuario además ingrese la base que quiere expresar el número, donde la base puede ser entre 2 y 9.