Clase 07 - Control de flujo cíclico (I)

IIC1103-07 - Introducción a la Programación

Cristian Ruz – cruz@ing.puc.cl

Jueves 29-Agosto-2019

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Contenidos

Contenidos

Laboratorios: Lab 1 y Lab 2 disponibles

- Laboratorios evaluados automáticamente (*hackerrank*)
- Pueden ir a cualquier laboratorio (Lunes a Jueves)
- Pueden hacerlos **fuera de la sala** (Lunes a Domingo)
- No doy puntos por asistencia, pero les recomiendo ir
  - SIN LAPTOP
    - Lunes a Jueves, mód 5 y 6: Lab San Agustín (piso 2)
  - CON LAPTOP
    - Lunes, mód 5 y 6: CS204
    - Martes, mód 5 y 6: K200, A5, B12
    - Miércoles, mód 5 y 6: **B23**, CS203, C203
    - Jueves, mód 5 y 6: B13, K204, CS101
- **¡Este es un curso práctico! Aprender haciendo**
- **¡¡LABS 1 y 2 ya están disponibles!!**
- **Lunes 23-Septiembre, 18:00**, se recolecta puntaje de Labs 1 al 4.

Contenidos

Hackerrank de Contol de Flujo Condicional

Hasta el viernes 30 a las 20:00.

300 puntos de participación.

Contenidos

¿En qué vamos?

1. **Variables y expresiones**
2. **Control de flujo** $\Leftarrow$ *Aquí vamos*
3. Funciones y recursión $\Leftarrow$ *I1: 24-Sept, 18:30*
4. Strings
5. Listas
6. Tipos de datos personalizados (objetos)
7. Ordenación y búsqueda
8. Archivos

Control de flujo: `if`, `elif` y `else`

`if`: Ejecuta un trozo de código *si* se cumple una condición.

Sintaxis `if`

```
if condicion:                #debe entregar bool
  bloque_de_codigo_if
  bloque_de_codigo_if
  ...
  bloque_de_codigo_if
bloque_de_codigo_fuera_de_if
```

```
a = int(input("Ingrese número del 0 al 9: "))
if not((0 <= a) and (a <= 9)):
  print("¡Error! número mal ingresado")
print("¡Gracias!")
```

Control de flujo: `if`, `elif` y `else`

`else`: Ejecuta un trozo de código *si* no se cumple la condición de un `if` anterior.

Sintaxis `if-else`

```
if condicion:          #debe entregar bool
  bloque_de_codigo_if
  ...
  bloque_de_codigo_if
else:                  #si la condicion de if es False
  bloque_de_codigo_else
  ...
  bloque_de_codigo_else
bloque_de_codigo_fuera_de_if_else
```

Control de flujo: `if`, `elif` y `else`

`elif`: ejecuta un trozo de código *si* se cumple una condición y no se ha cumplido ningún `if` o `elif` anterior.

Sintaxis

```
1  if condicion:          #debe entregar bool
2    bloque_de_codigo_if ...
3    bloque_de_codigo_if
4  elif condicion:        #si if es False y elif es True
5    bloque_de_codigo_elif ...
6    bloque_de_codigo_elif
7  else:                  #si if es False y elif es False
8    bloque_de_codigo_else ...
9    bloque_de_codigo_else
10 bloque_de_codigo_fuera_de_if_elif_else
```

- (1) debe existir un `if`
- (2) pueden haber varios `elif`
- (3) podría no haber `else`

Código de marcador de fútbol

```
local = 0          # goles del local
visita = 0         # goles de la visita
# [1: local, 2: visita, otro: fin partido]
gol = int(input("¿Qué equipo anotó el gol?"))
print("Gol del equipo: ", gol)

local = local + 1      # gol local
visita = visita + 1    # gol visita

print("Local", local, "-", visita, "Visita")
# Ahora veo quien gana
if local > visita:
  print("¡Ganó el local!")
else if local < visita:
  print("¡Ganó la visita!")
else:
  print("¡Empate!")
```

Contenidos

Ciclos (loops): `while`

Instrucción `while`

Permite ejecutar varias veces la misma sección de código

```
1  while condicion:                # Siempre debe estar
2    bloque_de_codigo_while        # Este bloque se repite
3    ...                           # MIENTRAS se cumpla la
4    ...                           # condicion
5    ...
6    ...                           # Despues de esto se vuelve
7    bloque_de_codigo_while        # a comprobar la condicion
8
9  codigo_fuera_de_while           # Esto se ejecuta cuando
10 codigo_fuera_de_while           # la condicion
11                                 # deja de cumplirse
```

Ciclos (loops): `while`

¿Cuántas veces se imprime?

```
1 i = 0
2 while i < 10:      # loop que se ejecuta 10 veces
3   i += 1
4   print(i)
```

¿Y aquí?

```
1 i = 0
2 while True:        # loop infinito
3   i += 1
4   print(i)
```

Ciclos (loops): `while`

¿Cuándo dejamos de leer números?

```
1 a = int(input("Ingresa numero entre 0 y 9: "))
2 condicion = (0<=a) and (a<=9)
3
4 while condicion:
5   print("Gracias por ingresar",a)
6
7 print("No hay mas numeros :(")
```

Si no modificamos la condición, el ciclo **`while`** se ejecutará sin término (**¡loop infinito!**)

Ciclos (loops): `while`

¿Cuándo dejamos de leer números?

```
1  a = int(input("Ingresa numero entre 0 y 9: "))
2  condicion = (0<=a) and (a<=9)
3
4  while condicion:
5    print("Gracias por ingresar",a)
6
7    a = int(input("Ingresa numero entre 0 y 9: "))
8    condicion = (0<=a) and (a<=9)
9
10 print("No hay mas numeros :(")
```

Contenidos

Ejercicio 1

Código que pide un número n al usuario, e imprime todos los números entre 1 y n

(5 líneas de código)

Ejercicio 1

Código que pide un número n al usuario, e imprime todos los números entre 1 y n

```
1 n = int(input("Ingrese un numero entero: "))
2 i = 1
3 while i<=n:
4   print(i)
5   i = i + 1
```

Ejercicio 2

Código que pide un número n al usuario, e imprima todos los números *pares* entre 1 y n

(al menos dos maneras distintas)

Ejercicio 2

Código que pide un número *n* al usuario, e imprima todos los números *pares* entre 1 y *n* (1 línea más)

```
1 n = int(input("Ingrese un numero entero: "))
2 i = 1
3 while i<=n:
4   if (i%2)==0:    #Unica linea extra
5     print(i)      #Recuerden indentar esta linea
6   i = i + 1
```

Ejercicio 2

Código que pide un número *n* al usuario, e imprima todos los números *pares* entre 1 y *n* (alternativa)

```
1 n = int(input("Ingrese un numero entero: "))
2 i = 2              #Empieza con un numero par
3 while i<=n:
4   print(i)         #Imprimo todos
5   i = i + 2        #Pero ahora sumo 2
```

Ejercicio 3

Código que pide un número n al usuario, e indique si el número es primo. El programa debe terminar si el usuario ingresa 1

Ejercicio 3

Código que pide un número n al usuario, e indique si el número es primo. Debe terminar si $n \leq 1$

```
#Primera idea
n = int(input("Ingrese un numero entero:"))
if n<=1:
  print(n,"no es primo ni compuesto")
else:
  esPrimo = True  #Supongamos que n es primo
  if esPrimo:
    print(n, "es primo")
  else:
    print(n, "es compuesto")
```

¿Cuándo n es primo? Cuando n es divisible **SOLAMENTE** por 1 y por n.

Ejercicio 3: ¿Cuándo un número es primo?

Cuando *n* es divisible **SOLAMENTE** por 1 y por *n*.

```python
#Segunda idea
n = int(input("Ingrese un numero entero:"))
if n<=1:
  print(n,"no es primo ni compuesto")
else:
  esPrimo = False  #Supongamos que n es primo
  if (n%1)==0 and (n%n)==0:
    esPrimo = True
  if esPrimo:
    print(n, "es primo")
  else:
    print(n, "es compuesto")
```

¿Qué problema tiene esta solución?

Ejercicio 3: ¿Cuándo un número es primo?

Necesitamos que **NO SEA DIVISIBLE** por algún i desde 2 a n[1]

```python
#Tercera idea (la vencida)
n = int(input("Ingrese un numero entero: "))
if n<=1:
  print(n,"no es primo ni compuesto")
else:
  esPrimo = True  #Supongamos que n es primo
  i = 2
  while i<n:
    if (n%i)==0:  #Si es divisible por i
      esPrimo = False  #ya no es primo
    i = i+1
  if esPrimo:
    print(n, "es primo")
  else:
    print(n, "es compuesto")
```

[1] Puede ser solo hasta $\sqrt{n}$

Ejercicio 3: Para pensar ...

¿Puede ser más rápido?

```
#Version reducida
n = int(input("Ingrese un numero entero: "))
if n<=1: print(n,"no es primo ni compuesto")
else:
  esPrimo = True  #Supongamos que n es primo
  i = 2
  while i<n:
    if (n%i)==0:  #Si es divisible por i
      esPrimo = False  #ya no es primo
    i += 1
  if esPrimo: print(n, "es primo")
  else: print(n, "es compuesto")
```

¿Necesito probar todos los números si ya no es primo?

¿Cuánto demora en verificar si 179426549 es primo? (lo es)