Ing. Ventre, Luis O.

CAPITULO 4:

ESTRUCTURAS DE CONTROL

- Los programas vistos hasta hoy, presentan las siguientes secuencias:
  - Se piden **datos** al usuario.
  - Se efectúan **cálculos** con estos datos. Se guardan resultados en variables.
  - Se **muestran** por pantalla los resultados.
- Grupos de instrucciones ejecutadas una tras otra de manera secuencial. El **flujo de ejecución** es secuencial.

- No obstante es de interés que el **flujo de control** pueda ser alterado:

  - **SENTENCIAS CONDICIONALES O DE SELECCIÓN**

    Tomando decisiones a partir de los datos/resultados lo que produzca la ejecución de uno u otro grupo de instrucciones.

  - **SENTENCIAS ITERATIVAS O DE REPETICION**

    Tomando decisiones a partir de los datos y/o resultados lo que produzca la ejecución de ciertas sentencias MAS de una vez.

- Ambos tipos de sentencias son denominadas **estructuras de control de FLUJO**, o simplemente **ESTRUCTURAS DE CONTROL.**

- Adicionalmente se trataran las sentencias de emisión y captura de excepciones.

- **Un ejemplo: Resolución de ecuaciones de primer grado.**

$$a.x + b = 0$$

- Datos del problema: generalmente (raw_input()) → a, b
- Que se desea calcular? → x

- Con estos datos, **como se calcula la salida a partir de la entrada**?:

$$x = -b/a$$

  - a PROGRAMAR!!!...

  - Primero solicitar los datos...a…b.
  - Luego…hacer el calculo….
  - Finalmente mostrar el resultado!

Sentencias Condicionales

- Nuestra primera aproximación seria:

primer_grado_7.py — primer_grado.py

```
1 a = float(raw_input('Valor de a: '))
2 b = float(raw_input('Valor de b: '))
3
4 x = -b / a
5
6 print 'Solución: ', x
```

- A copiar y ejecutar, funciona?....

Sentencias Condicionales

- Que sucede si el usuario escoge

a = 0

```
Valor de a: 0
Valor de b: 3
Traceback (innermost last):
  File 'primer_grado.py', line 3, in ?
    x = -b / a
ZeroDivisionError: float division
```

- Esto generara un error en tiempo de ejecución que el usuario del programa no comprenderá!...

- **Debemos solucionarlo.**

La sentencia IF

- Ideal seria, poder DETECTAR si

$$a = 0$$

- En cuyo caso **no ejecutar la instrucción** que provoca el error.

- **NECESITAMOS QUE UNA PARTE DEL PROGRAMA SE DEJE DE EJECUTAR EN FUNCION A UN VALOR.**

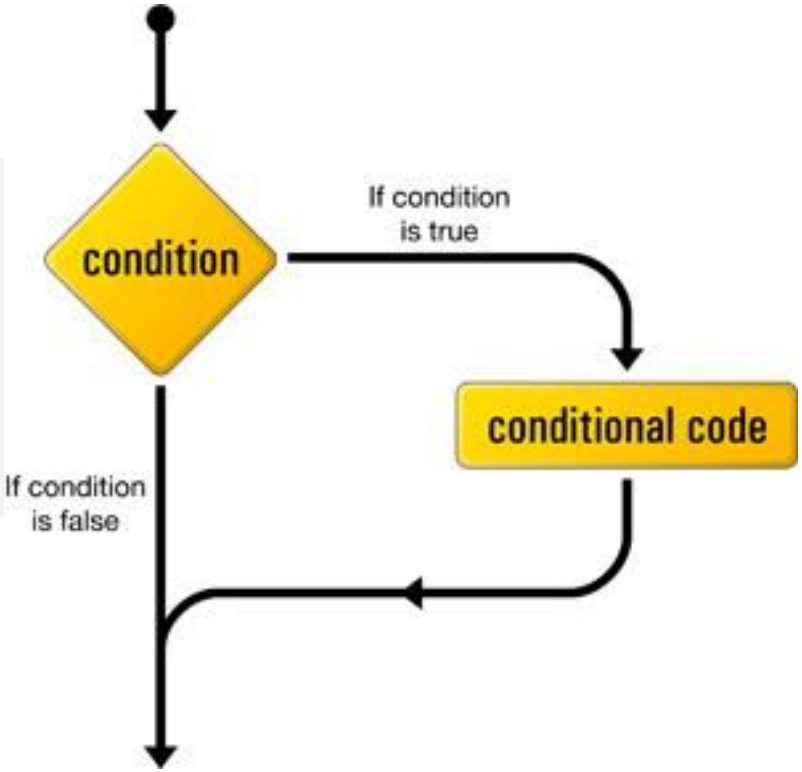
- La sentencia que resuelve esto implica:

  Al llegar a este punto, ejecuta esta(s) acción(es) solo si esta condición es cierta.

La sentencia IF

- Esta sentencia IF es denominada **CONDICIONAL**. y en python tiene la siguiente forma:

```
if condición:
    acción
    acción
    ...
    acción
```



- En nuestro caso, necesitamos detectar que a sea distinto de 0, y entonces ejecutar las instrucciones.

La sentencia IF

- Deberíamos entonces agregarle a nuestro programa:

```
if a!=0:
        x=-b/a
        print 'Solucion: ',x
```

- Leamos esto….la primera línea indica si a != (distinto) de 0.

- **OBSERVAR que la línea de la condición del IF debe finalizar con «:»**

La sentencia IF

- El programa quedara:

primer_grado_8.py

primer_grado.py

```
1 a = float(raw_input('Valor de a: '))
2 b = float(raw_input('Valor de b: '))
3
4 if a != 0:
5     x = -b/a
6     print 'Solución: ', x
```

- OBSERVAR que las líneas 5 y 6, tienen una **SEPARACION**, llamada **INDENTACION** con respecto a donde comienza el if.

- Esto implica que las líneas 5 y 6 **solo se ejecutan** si se cumple la condición del IF.

La sentencia IF

- Al ejecutarlo e ingresar a=0, que pasa?....
- Ocurre algún error?
- Se avisa al usuario de lo sucedido?, entiende el porque?
- Como lo solucionaría?

primer_grado_8.py

primer_grado.py

```
1 a = float(raw_input('Valor de a: '))
2 b = float(raw_input('Valor de b: '))
3
4 if a != 0:
5     x = -b/a
6     print 'Solución: ', x
```

La sentencia IF

- Veamos el siguiente programa....

primer_grado_9.py

primer_grado.py

```
1 a = float(raw_input('Valor de a: '))
2 b = float(raw_input('Valor de b: '))
3
4 if a != 0:
5     x = -b/a
6     print 'Solución: ', x
7 if a == 0:
8     print 'La ecuación no tiene solución.'
```

- Analicemos dos posibles ejecuciones…

primer_grado_9.py

primer_grado.py

```
1 a = float(raw_input('Valor de a: '))
2 b = float(raw_input('Valor de b: '))
3
4 if a != 0:
5 |   x = -b/a
6 |   print 'Solución: ', x
7 if a == 0:
8 |   print 'La ecuación no tiene solución.'
```

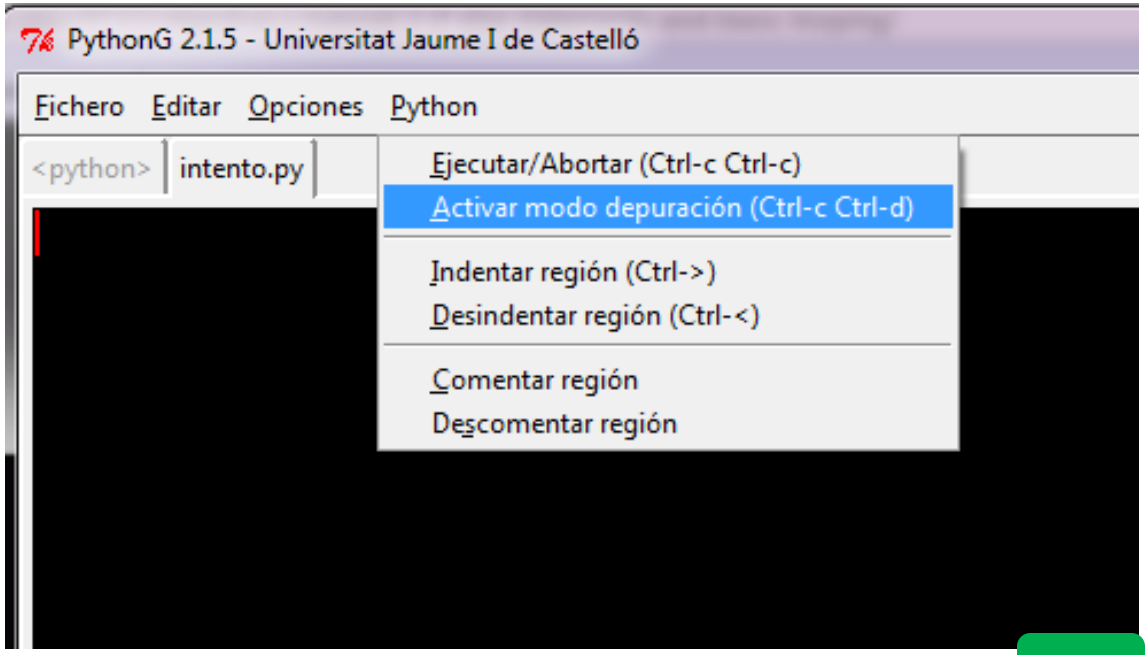
$a = 0$ y $b = 3$	$a = 1$ y $b = -1$
Las líneas 1 y 2 se ejecutan, con lo que se leen los valores de a y b.	Las líneas 1 y 2 se ejecutan, con lo que se leen los valores de a y b.
La línea 4 se ejecuta y el resultado de la comparación es *falso*.	La línea 4 se ejecuta y el resultado de la comparación es *cierto*.
Las líneas 5 y 6 se ignoran.	Se ejecutan las líneas 5 y 6, con lo que se muestra por pantalla el valor de la solución de la ecuación: **`Solución: 1`**.
La línea 7 se ejecuta y el resultado de la comparación es *cierto*.	La línea 7 se ejecuta y el resultado de la comparación es *falso*.
La línea 8 se ejecuta y se muestra por pantalla el mensaje «**`La ecuación no tiene solución.`**»	La línea 8 se ignora.

Trazas con PythonG

- Este tipo de análisis se denomina analizar la **traza de ejecución.**

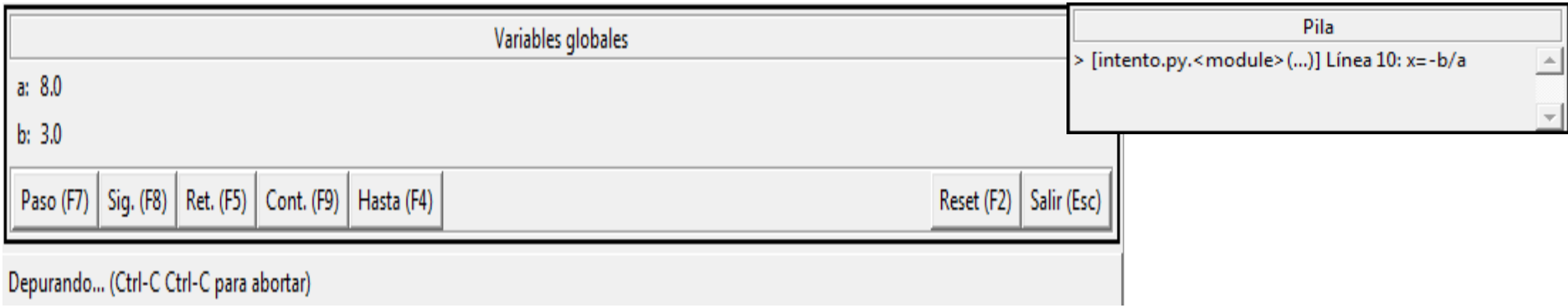
- El entorno de programación, tiene una herramienta que te permitirá seguir la traza de ejecución de un programa.

- Para activarla, selecciona **ACTIVAR MODO DEPURACION** del menú python.



Trazas con PythonG

- Con el modo de depuración activado veremos dos nuevas ventanas en la parte inferior.



- Cuando dudes de la traza de un programa **utilízalo**.

- Con la tecla F8, avanzas el programa paso a paso indicándote con una línea pintada que instrucción se va a ejecutar…

- Vamos a mejorar nuestro programa. Cuando a=0, nuestro programa imprime que la ecuación NO TIENE SOLUCION.

- Pero si además b=0, correcto seria INDICAR que la ECUACION TIENE INFINITAS soluciones.

- Veamos un ejemplo mas completo de como solucionar esto.

RECUERDA

EN PYTHON LA INDENTACION DETERMINA DE QUE SENTENCIA DEPENDE CADA BLOQUE DE SENTENCIAS

Sentencias IF Anidadas

- Veamos el siguiente código:

primer_grado_12.py

primer_grado.py

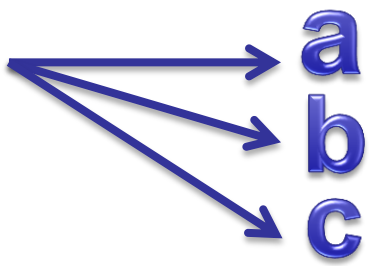
```
1  a = float(raw_input('Valor de a: '))
2  b = float(raw_input('Valor de b: '))
3
4  if a != 0:
5      x = -b/a
6      print 'Solución: ', x
7  if a == 0:
8      if b != 0:
9          print 'La ecuación no tiene solución.'
10     if b == 0:
11         print 'La ecuación tiene infinitas soluciones.'
```

RECUERDA: INDENTACION.

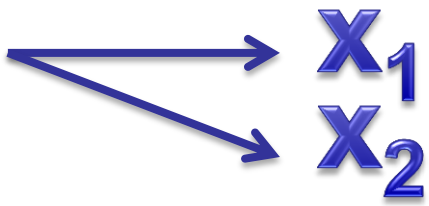
- Otro ej: resolución de ecuaciones de segundo grado.

$$a.x^2 + b.x + c = 0$$

- Datos del problema: generalmente (raw_input())



- Que se desea calcular?



- Con $x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ y $x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$.

Sentencias IF Anidadas

- El código podría ser:

segundo_grado_11.py

segundo_grado.py

```
1  from math import sqrt  # sqrt calcula la raíz cuadrada.
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
8  x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
9
10 print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
```

- …pero nuevamente si a=0 que sucede?

Sentencias IF Anidadas

- Una posible solución:

segundo_grado_12.py

segundo_grado.py

```
1  from math import sqrt
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  if a != 0:
8      x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
9      x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
10     print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
11 if a == 0:
12     print 'No es una ecuación de segundo grado.'
```

$$a.x^2 + b.x + c = 0$$

El caso contrario ELSE

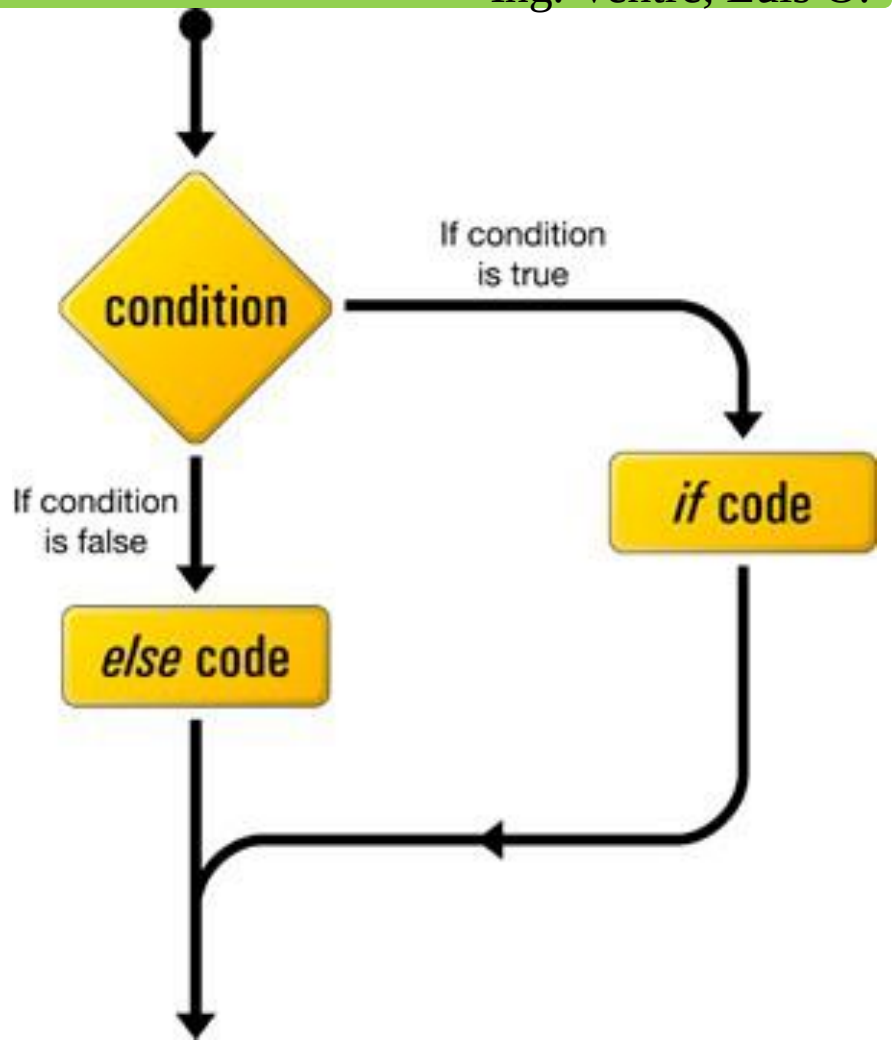
- Tanto en este ejemplo como en el anterior, hemos utilizado sentencias condicionales para ejecutar instrucciones si **SE CUMPLE** una determinada condición, y otra para el caso en que **NO SE CUMPLA** esta condición.

```
if condición:
    acciones
if condición contraria:
    otras acciones
```

- Esto es tan frecuente que el lenguaje incorpora una solución.

El caso contrario ELSE

- La forma abreviada utiliza la palabra **ELSE** que significa en Ingles SI NO o
EN CASO CONTRARIO.



- **El IF y el ELSE deben ir alineados en la misma columna**

Sentencias IF Anidadas

- Como quedaría el código:

segundo_grado_13.py

segundo_grado.py

```
1  from math import sqrt
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  if a != 0:
8      x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
9      x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
10     print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
11 else:
12     print 'No es una ecuación de segundo grado.'
```

- Seria bueno si a == 0, indicar la solución igual!

Sentencias IF Anidadas

- Como quedaría el código:

segundo_grado_14.py

segundo_grado.py

```
1  from math import sqrt
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  if a != 0:
8      x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
9      x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
10     print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
11 else:
12     x = -c / b
13     print 'Solución de la ecuación: x=%4.3f' % x
```

- Pero en este caso, si a==0 y b==0?

Sentencias IF Anidadas

segundo_grado_15.py

segundo_grado.py

```
1  from math import sqrt
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  if a != 0:
8      x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
9      x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
10     print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
11 else:
12     if b != 0:
13         x = -c / b
14         print 'Solución de la ecuación: x=%4.3f' % x
15     else:
16         if c != 0:
17             print 'La ecuación no tiene solución.'
18         else:
19             print 'La ecuación tiene infinitas soluciones.'
```

Analice el siguiente programa, funciona?

segundo_grado_16.py

segundo_grado.py

```
1  from math import sqrt
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  if a == 0:
8      if b == 0:
9          if c == 0:
10             print 'La ecuación tiene infinitas soluciones.'
11         else:
12             print 'La ecuación no tiene solución.'
13     else:
14         x = -c / b
15         print 'Solución de la ecuación: x=%4.3f' % x
16 else:
17     x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
18     x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
19     print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
```

Analice el siguiente programa, funciona?

segundo_grado_17.py

segundo_grado.py

```
1  from math import sqrt
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  if a == 0 and b == 0 and c == 0:
8      print 'La ecuación tiene infinitas soluciones.'
9  else:
10     if a == 0 and b == 0:
11         print 'La ecuación no tiene solución.'
12     else:
13         if a == 0:
14             x = -c / b
15             print 'Solución de la ecuación: x=%4.3f' % x
16         else:
17             x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
18             x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
19             print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
```

Y si a=1, b=2, y c=3????.....

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

```
if a != 0:
    if b**2 - 4*a*c >= 0:
        x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
        x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
        print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
    else:
        print 'No hay soluciones reales.'
else:
    if b != 0:
        x = -c / b
        print 'Solución de la ecuación: x=%4.3f' % x
    else:
        if c !=0:
            print 'La ecuación no tiene solución.'
        else:
            print 'La ecuación tiene infinitas soluciones.'
```

El diseño por refinamientos sucesivos

- Al comenzar a programar es erróneo pensar el programa COMPLETO en una primera instancia.

- ESTRATEGIA a seguir:

1. **Versión en papel** del algoritmo **SIMPLE**. Ingreso de datos, cálculos e impresión de resultados.

2. Analizar funcionamiento para **valores particulares**. Divisiones por cero por ejemplo.

El diseño por refinamientos sucesivos

3. Cada vez que plantees una de estas preguntas y encuentres la respuesta **MODIFICA** el programa en consecuencia.

4. **No** hagas **mas de un cambio** a la vez.

5. Si el programa funciona OK para todos los posibles valores, ya casi has terminado.

6. Ahora que entiendes todo el funcionamiento del programa, **transcríbelo** a la PC.

El diseño por refinamientos sucesivos

- Nadie es capaz de escribir un programa de principio a fin sin errores en una sola sentada.
- Un error frecuente es **comenzar a escribir el código en la PC** sin comprender completamente el programa.
- Esto implicará que los **detalles del lenguaje interfieran** con el diseño de la solución. Llevaba dos puntos al final de la línea, al imprimir llevaba comillas?...etc
- Si ya has realizado un programa y no funciona, mas complicado es **parchar** el mismo que escribir de 0 uno que funcione.

MAXIMO de una serie de números

- Veamos otro ejemplo. Para utilizar sentencias condicionales.

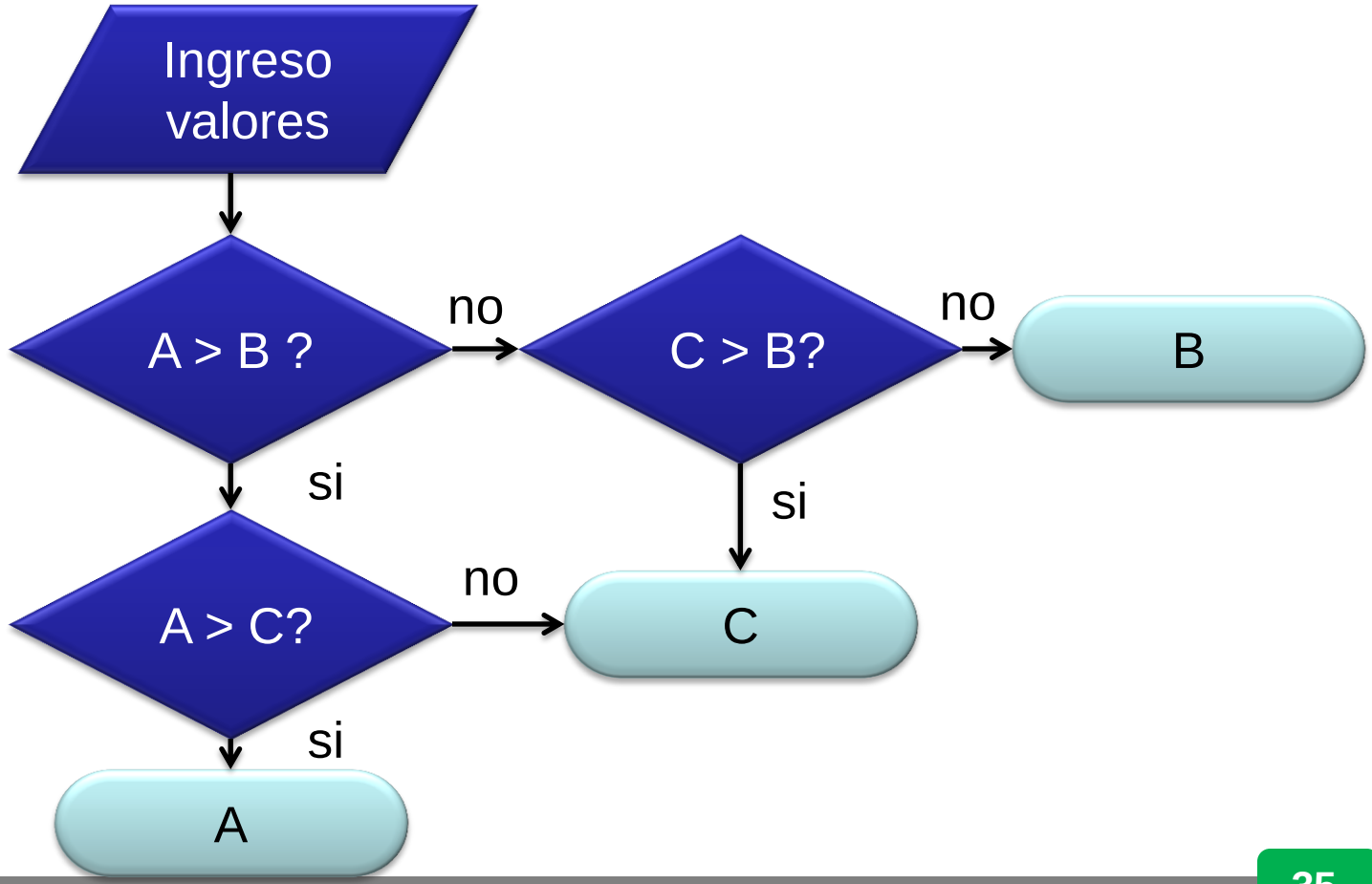
- El objetivo del programa es **SOLICITAR** al usuario el ingreso de dos números, y luego imprimir por pantalla el MAYOR.

- Haz tu primer intento!...

- La sección de cálculos de tu programa puede parecerse a:

```
4 if a > b:
5     maximo = a
6 else:
7     maximo = b
```

- Y si ahora el objetivo seria DETERMINAR EL MAYOR de 3 números?...



MAXIMO de una serie de números

Ing. Ventre, Luis O.

- Una vez comprendido el algoritmo el código es simple!...

```
4
5  if a > b:
6      if a > c:
7          maximo = a
8      else:
9          maximo = c
10 else:
11     if b > c:
12         maximo = b
13     else:
14         maximo = c
15
16 print 'El máximo es', maximo
```

Y si ahora habría que determinar el **mayor de 5 valores**?....

MAXIMO de una serie de números

- Observa esta solución…y analiza como resolverlo para el máximo de 5 valores!

maximo_de_tres_4.py

maximo_de_tres.py

```
1  a = int(raw_input('Dame el primer número: '))
2  b = int(raw_input('Dame el segundo número: '))
3  c = int(raw_input('Dame el tercer número: '))
4
5  candidato = a
6  if b > candidato:
7      candidato = b
8  if c > candidato:
9      candidato = c
10 maximo = candidato
11
12 print 'El máximo es', maximo
```

MAXIMO de una serie de números

- También pueden utilizarse operadores lógicos y comparaciones…

maximo_de_tres.py

```
1  a = int(raw_input('Dame el primer número: '))
2  b = int(raw_input('Dame el segundo número: '))
3  c = int(raw_input('Dame el tercer número: '))
4
5  if a >= b and a >= c:
6      maximo = a
7  if b >= a and b >= c:
8      maximo = b
9  if c >= a and c >= b:
10     maximo = c
11 print 'El máximo es', maximo
```

MAXIMO de una serie de números

- Lea este programa e indique que valores provocaran cada impresión…

```
1  mes = int(raw_input('Dame un mes: '))
2
3  if 1 <= mes <= 3:
4     print 'Invierno.'
5  else:
6     if mes == 4 or mes == 5 or mes == 6:
7        print 'Primavera.'
8     else:
9        if not (mes < 7 or 9 < mes):
10          print 'Verano.'
11       else:
12          if not (mes != 10 and mes != 11 and mes != 12):
13             print 'Otoño.'
14          else:
15             print 'Ningún año tiene %d meses.' % mes
```

Evaluación con CORTOCIRCUITOS

- En python la evaluación de expresiones lógicas es algo especial.
- Hay **división por 0** en los siguientes ej?

a=0

```
if a != 0 and 1/a > 1:
    ...
```

a=0

```
if a == 0 or 1/a > 1:
    ...
```

- Cuando hay una expresión **AND**, y el primer termino es **falso NO CONTINUA** el análisis y devuelve falso.

- Cuando hay una expresión **OR**, y el primer termino es **verdadero** NO CONTINUA el análisis y devuelve verdadero

Ultimo problema: Menús de usuario

- Supongamos que deseamos desarrollar un programa que:
  - Luego del ingreso de 1 valor por teclado.
  - Brinde un **menú**, en cual el usuario pueda elegir 1 de 3 **opciones.**
  - A través del ingreso de otra valor, en este caso una **letra.**
  - En función a la elección se deberá hacer **el calculo** seleccionado e **imprimir** por pantalla.
  - Se anima?

Ultimo problema: Menús de usuario

- Supongamos un programa que SOLICITE EL RADIO DE UN CIRCULO.

- Luego SOLICITE una opción:

  - ‘a’ para CALCULO DEL DIAMETRO
  - ‘b’ para CALCULO DEL PERIMETRO
  - ‘c’ para CALCULO DE LA SUPERFICIE

- Y si después quisiera mejorarlo y si teclea una opción invalida avisarle?

Ultimo problema: Menús de usuario

```
3  radio = float(raw_input('Dame el radio de un círculo: '))
4
5  # Menú
6  print 'Escoge una opción: '
7  print 'a) Calcular el diámetro.'
8  print 'b) Calcular el perímetro.'
9  print 'c) Calcular el área.'
10 opcion = raw_input('Teclea a, b o c y pulsa el retorno de carro: ')
11
12 if opcion == 'a':  # Cálculo del diámetro.
13     diametro = 2 * radio
14     print 'El diámetro es', diametro
15 else:
16     if opcion == 'b': # Cálculo del perímetro.
17         perimetro = 2 * pi * radio
18         print 'El perímetro es', perimetro
19     else:
20         if opcion == 'c': # Cálculo del área.
21         area = pi * radio ** 2
22         print 'El área es', area
```

Forma compacta ELIF

- En los programas como el anterior, las opciones nos llevan a **INDENTAR** el programa cada vez mas a la derecha…

- Esto hace parecer MAS COMPLICADO de lo que es un programa, por la cantidad de niveles de indentación.

- Python ofrece una forma **compacta** de expresar el código:

- Un ELSE que esta seguido de un IF, se puede reemplazar por un ELIF (condicion):

Forma compacta ELIF

- El código:

```
if condición:
    ...
else:
    if otra condición:
        ...
```

se reemplaza por el siguiente y se ahorra una indentación.

```
if condición:
    ...
elif otra condición:
    ...
```

Veamos como quedaría el programa anterior!!

Ultimo problema: Menús de usuario

```
3  radio = float(raw_input('Dame el radio de un círculo: '))
4
5  print 'Escoge una opción: '
6  print 'a) Calcular el diámetro.'
7  print 'b) Calcular el perímetro.'
8  print 'c) Calcular el área.'
9  opcion = raw_input('Teclea a, b o c y pulsa el retorno de carro: ')
10
11 if opcion == 'a':
12     diametro = 2 * radio
13     print 'El diámetro es', diametro
14 elif opcion == 'b':
15     perimetro = 2 * pi * radio
16     print 'El perímetro es', perimetro
17 elif opcion == 'c':
18     area = pi * radio ** 2
19     print 'El área es', area
20 else:
21     print 'Sólo hay tres opciones: a, b o c. Tú has tecleado', opcion
```

Repasando!...

- **Programas secuenciales, como alterar el flujo de ejecución.**
- **Estructuras de CONTROL.**
- **Sentencias condicionales.**
- **La sentencia IF….su uso en resolución ecuaciones 1er grado.**

- Trazas con PythonG…..el modo DEPURADOR.
- Sentencias condicionales ANIDADAS. If adentro If adentro…
- Resolución ecuaciones 2do grado.
- Ejemplo para determinar el MAXIMO de una serie de números….

Lo visto!

- Evaluación de expresiones lógicas en Python…
- Como implementar MENUS de USUARIO, con diferentes opciones.
- Forma compacta para estructuras condicionales MULTIPLES: ELIF