Universidad Nacional de Jujuy
Facultad de Ingeniería

Introducción a la Programación
Programación I

Estructura repetitiva

Contenido

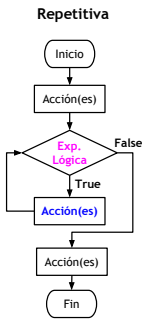
- Introducción - Sentencia repetitiva while
- Divisibilidad y factores primos
- Cálculo del mcd por el método de Euclides
- Prueba de escritorio - Traza
- Concepto de Acumulador - Contador
- Formas de finalización del ciclo while
- Sentencia repetitiva for - in serie de valores
- Sentencias de control de bucle
- Ejemplos

Introducción - Sentencia repetitiva while

Es la estructura de control que permite repetir un conjunto de **Acción(es)**. La cantidad de veces que se repite el conjunto de **Acción(es)** depende de la **Expresión lógica**.

while **Exp.Lógica**:
Acción(es)

bucle ≡ ciclo ≡ iteración ≡ repetición



Cálculo del máximo común divisor

Factores primos
2366 = 2.7.13.13
273 = 3.7.13

mcd(**2366**, **273**) = **91**

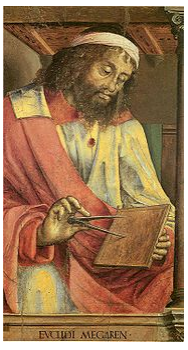
ddo	dsor
- r -	c

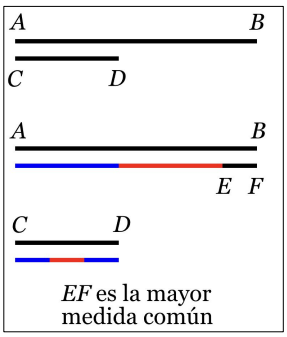
r = ddo–ddo//dsor*dsor

Divisores
2366: 1, 2, 7, 13, 14, 26, **91**, 169, 182, 338, 1183, 2366

273: 1, 3, 7, 13, 21, 39, **91**, 273

Algoritmo mcd de Euclides (330 - 275 a.C.)





Algoritmo de Euclides - Elementos

1. Dados dos segmentos *AB* y *CD* (con *AB*>*CD*), restamos *CD* de *AB* tantas veces como sea posible. Si no hay residuo, entonces *CD* es la máxima medida común.
2. Si se obtiene un residuo *EF*, este es menor que *CD* y podemos repetir el proceso: restamos *EF* tantas veces como sea posible de *CD*. Si al final no queda un residuo, *EF* es la medida común. En caso contrario obtenemos un nuevo residuo *FC* menor a *EF*.
3. El proceso se repite hasta que en algún momento no se obtiene residuo. Entonces el último residuo obtenido es la mayor medida común.

Paso	Operación
1	2366 dividido entre 273 es 8 y sobran 182
2	273 dividido entre 182 es 1 y sobran 91
3	182 dividido entre 91 es 2 y sobra 0

Prueba de escritorio o Traza

```
a = int(input('a:')) #2366
b = int(input('b:')) #273

while b != 0:
    r = a % b
    a = b
    b = r
print('mcd:', a)
```

Asignación múltiple

a, b = b, a % b

Traza

a	b	r
-	-	-
2366	-	-
2366	273	-
2366	273	182
273	273	182
273	182	182
273	182	91
182	182	91
182	91	91
182	91	0
91	91	0
91	0	0

Acumulador - Asignación

Es una variable (**v**) que aumenta o disminuye en una cantidad (**x**) constante o variable dentro del ciclo. Antes del while se asigna a la variable **v** un valor inicial (**vi**) y en cada iteración se re-evalúa la exp.lógica.

```
v = vi
while exp.lógica:
    Acción(es)
    v = v + x
    Acción(es)
```

Asignación	Equivalencia
v = vi	v = vi
v = v + x	v += x
v = v - x	v -= x
v = v*x	v *= x
v = v/x	v /= x
v = v%x	v %= x
v = v//x	v //= x
v = v**x	v **= x

Formas clásicas de finalización del ciclo

Finalización del ciclo:

- Por pedido del operador
- Por valor de salida
- Por variable lógica
- Por cantidad de repeticiones

Ejemplos:

- recaudación al final del día
- cantidad de goles a favor y en contra
- saldo de dinero luego de las compras
- cantidad de cerveza ingeridas

Finalizar por pedido del operador

```
#Acumular valores y contarlos.
acumulador = 0.0
contador = 0
continuar = 's'
while (continuar == 's'):
  valor = float(input('Ingrese valor: '))
  acumulador = acumulador + valor
  contador = contador + 1
  continuar = input('Desea continuar (s/n): ')
print('Total: ', acumulador)
print('Cantidad: ', contador)
```


Finalizar por valor de salida

```
#Acumular valores y contarlos.
acumulador = 0.0
contador = 0
valor = float(input('Ingrese valor (0) terminar: '))
while (valor != 0.0):  # 0.0 valor de salida
  acumulador = acumulador + valor
  contador = contador + 1
  valor = float(input('Ingrese valor (0) terminar: '))
print('Total: ', acumulador)
print('Cantidad: ', contador)
```


Finalizar por variable lógica

```
#Acumular valores y contarlos.
acumulador = 0.0
contador = 0
valor = float(input('Ingrese valor (0) terminar: '))
diferenteCero = (valor != 0.0)
while diferenteCero:
  acumulador = acumulador + valor
  contador = contador + 1
  valor = float(input('Ingrese valor (0) terminar: '))
  diferenteCero = (valor != 0.0)
print('Total: ', acumulador)
print('Cantidad: ', contador)
```

Finalizar por cantidad de repeticiones

```
#Acumular n cantidad de valores
acumulador = 0.0
contador = 0
n = int(input('Ingrese cantidad de valores: '))
while (contador < n):
   valor = float(input('Ingrese valor: '))
   acumulador = acumulador + valor
   contador = contador + 1
print('Total: ', acumulador)
print('Cantidad: ', contador)
```


Sentencia for en una secuencia de valores

Itera sobre los ítems de cualquier secuencia de valores: rango, lista, cadena de texto, tupla, en el orden que aparecen en la secuencia.

```
for vc in secuencia:
   Acción(es)
```

Si la secuencia es range

```
for vc in range(vi, vf, p):
   Acción(es)
```

- vi < vf, p > 0, vc < vf
- vi > vf, p < 0, vc > vf

```
vc = vi
while vc < vf:
   Acción(es)
   vc = vc + p
```


Ejemplos para la sentencia for

for i in range(3):
  print(i)
	# 0
	# 1
	# 2
		for i in range(5, 8):
		  print(i, i ** 2)
			# 5 25
			# 6 36
			# 7 49

for i in range(6, 0, -2):
  print(i)
	# 6
	# 4
	# 2
		for i in 'Yes':
		  print(i)
			# Y
			# e
			# s

```
#Acumular n cantidad de valores
acumulador = 0.0
n = int(input('Ingrese cantidad de valores: '))
for i in range(n): # i inicia en 0 y finaliza en n-1
   valor = float(input('Ingrese valor: '))
   acumulador = acumulador + valor
print('Total: ', acumulador)
print('Cantidad: ', i) # se cuenta desde 0
```


Sentencias de control de bucle

En los bucles while y for, a veces, puede surgir una condición en la que desee salir del bucle por completo, omitir una iteración o ignorar esa condición.
Las declaraciones de control de bucle cambian la ejecución de su secuencia normal. Cuando la ejecución sale de un ámbito, todos los objetos automáticos que se crearon en ese ámbito se destruyen. Python admite las siguientes declaraciones de control: break, continue, pass y la cláusula else.

Emular la sentencia do … while

```
do {
     Acción(es);
  }
while Exp.Lógica;
```

```
clave = "123"
c = 0

while True:
   c1 = input("Dame la palabra clave: ")
   if c1 == clave:
      print("Le atinaste")
      break
   else:
      print('Incorrecto')
   c += 1
   if c >= 3:
      break
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
   char c1[90];
   char clave[]="123";
   int c = 0;

   do{
      printf("Dame la palabra clave: ");
      scanf("%s", c1);
      if (strcmp (c1,clave)==0) {
         printf("Le atinaste\n");
         break; }
      else{
         printf("Incorrecto\n"); }
      c++;
    }
   while (c < 3);
   return 0;
}
```


Bibliografía y enlaces

- Libro: Introducción a la Programación con Python. Unidades 4.1, 4.2
- Algoritmo de Euclides https://es.wikipedia.org/wiki/Algoritmo_de_Euclides
- Calculadora Factores primos https://web.archive.org/web/20061110032923/http://www.btinternet.com/~se16/js/factor.htm
- Calculadora de divisores de un número http://nosolomates.es/ayuda/ayuda/divisores.htm