

Introducción a la computación

Primer Cuatrimestre de 2014

Programa

Un **programa** es una secuencia de **instrucciones**.

- Asignación
VARIABLE = EXPRESIÓN
- Condicional
if CONDICION: PROG1 else: PROG2
- Ciclo
while CONDICION: PROG1

Volviendo a nuestro ejemplo...

```
fil = 1
while fil <= 5:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil + 1
```

```
fil = 1
while fil <= 5:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil + 1
```

Output:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
fil = 1
while fil <= 5:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil + 1
```

Output:

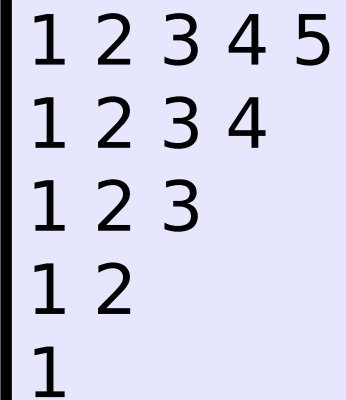
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

?

```
fil = 5
while fil >= 1:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil - 1
```

Output:



```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
fil = 5
while fil >= 1:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil - 1
```

Output:

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
1 2 3 4 5
1 2 3 4 5
1 2 3
1 2 3
1
1
```

?

```
fil = 5
while fil >= 1:
    if fil % 2 != 0:
        col = 1
        while col <= fil:
            print col,
            col = col + 1
        print
        col = 1
        while col <= fil:
            print col,
            col = col + 1
        print
    fil = fil - 1
```

Output:

```
1 2 3 4 5
1 2 3 4 5
1 2 3
1 2 3
1
1
```



```

fil = 5
while fil >= 1:
    if fil % 2 != 0:
        col = 1
        while col <= fil:
            print col,
            col = col + 1
        print
        col = 1
        while col <= fil:
            print col,
            col = col + 1
        print
    fil = fil - 1

```

Output:

```

1 2 3 4 5
1 2 3 4 5
1 2 3
1 2 3
1
1

```

```

1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3
1 2 3
1 2 3
1
1
1

```

?

```

fil = 1
while fil <= 5:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil + 1

```

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

```

fil = 5
while fil >= 1:
    if fil % 2 != 0:
        col = 1
        while col <= fil:
            print col,
            col = col + 1
        print
        col = 1
        while col <= fil:
            print col,
            col = col + 1
        print
    fil = fil - 1

```

```

1 2 3 4 5
1 2 3 4 5
1 2 3
1 2 3
1
1

```

```

fil = 5
while fil >= 1:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil - 1

```

```

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

```

¿Cómo puedo hacer para reusar este código sin tener que copiarlo una y otra vez?

```
def imprimir_fila(fil):  
    col = 1  
    while col <= fil:  
        print col,  
        col = col + 1  
    print
```

Ejemplos:

<code>imprimir_fila(2)</code>	imprime: 1 2
<code>imprimir_fila(5)</code>	imprime: 1 2 3 4 5

```

fil = 1
while fil <= 5:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil + 1

```

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

```

fil = 5
while fil >= 1:
    col = 1
    while col <= fil:
        print col,
        col = col + 1
    print
    fil = fil - 1

```

```

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

```

```

fil = 5
while fil >= 1:
    if fil % 2 != 0:
        col = 1
        while col <= fil:
            print col,
            col = col + 1
        print
        col = 1
        while col <= fil:
            print col,
            col = col + 1
        print
    fil = fil - 1

```

```

1 2 3 4 5
1 2 3 4 5
1 2 3
1 2 3
1
1

```

```

fil = 1
while fil <= 5:
    imprimir_fila(fil)
    fil = fil + 1

```

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

```

fil = 5
while fil >= 1:
    if fil % 2 != 0:
        imprimir_fila(fil)
        imprimir_fila(fil)
    fil = fil - 1

```

```

1 2 3 4 5
1 2 3 4 5
1 2 3
1 2 3
1
1

```

```

fil = 5
while fil >= 1:
    imprimir_fila(fil)
    fil = fil - 1

```

```

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

```

Resultado: Código modular.

- Más claro para los humanos.
- Más fácil de actualizar.

(Ej: ¿Qué pasa si quiero separar los números con “,” en lugar de “ ”?)

Función

Una **función** es una unidad de código que **aísla** una parte de un cómputo. Es un programa dentro de un programa.

- Permite dividir un problema en **problemas más simples**.
- Permite **ordenar** conceptualmente el código para que sea más fácil de entender.
- Permite **reutilizar** soluciones a problemas pequeños en la solución de problemas mayores.

Procedimiento

```
def imprimir_filas(fil):  
    col = 1  
    while col <= fil:  
        print col,  
        col = col + 1  
    print
```

Procedimiento == Función que no devuelve valor alguno.

Función

```
def raiz_cuadrada(n):  
    i = 1  
    while i * i <= n:  
        i = i + 1  
    return i - 1
```

Ejemplos:

<code>raiz_cuadrada(8)</code>	devuelve: 2
<code>raiz_cuadrada(9)</code>	devuelve: 3
<code>raiz_cuadrada(10)</code>	devuelve: 3


```
#!/usr/bin/python

def raiz_cuadrada(n):
    i = 1
    while i * i <= n:
        i = i + 1
    return i - 1

x = 1
while x <= 5:
    print raiz_cuadrada(x)
    x = x + 1
```

Cada ejecución de una función tiene su **propio espacio de memoria**, como si fuera otro programa.

n, **i** son **alcanzables** dentro de `raiz_cuadrada`, pero no fuera.

Un detalle técnico

En Python, las funciones deben definirse **antes** de ser usadas.

- Antes == Más arriba en el archivo de código.
- Por eso, el código principal suele definirse abajo de todo.

Repaso de la clase de hoy

- Modularidad del código: funciones.
- Alcance (scope) de variables.

Próximas clases teóricas

- Especificación de problemas.
- Corrección de programas.