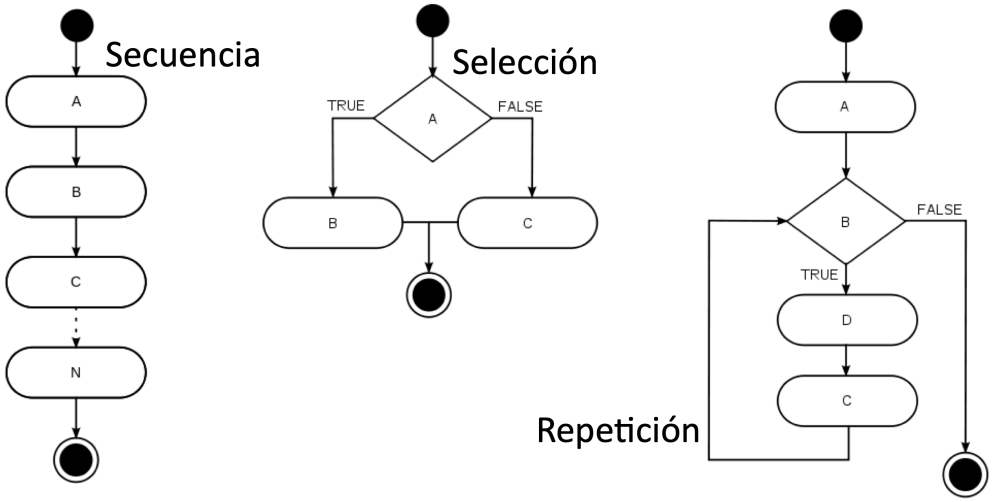
2. Entradas y Salidas. Estructuras de Control de Flujo

1 Introducción

Un programa consta de un conjunto de instrucciones organizadas mediante las siguientes estructuras de control:



2 Salida por pantalla (*print*)

```
print(string_0[,...,string_n,sep=' ', end='\n', flush=False])
```

Argumentos:

- `sep`: Especifica el separador entre cadenas (por defecto: un espacio blanco)
- `end`: Especifica la cadena finalizadora (por defecto: nueva línea)
- `flush`: Valor *True* indica que se refresca el buffer de escritura inmediatamente (permite mostrar texto en el momento preciso en que se desee)

Caracteres especiales:

Carácter especial	Significado
\n	Nueva línea
\t	Tabulador
\'	Comilla simple '
\"	Comillas dobles "
\r	Ir al principio de la línea de texto en pantalla
\%	Carácter %

[1]:
```
print('Hola,')
print('¿qué tal estás,','amigo mío?')
```

```
Hola,
¿qué tal estás, amigo mío?
```

[2]:
```
print('Hola,',end=' ') # Cambia el finalizador
print('¿qué tal estás,','amigo mío?')
```

```
Hola, ¿qué tal estás, amigo mío?
```

[3]:
```
print ('Juana','Valentín','María del Mar','Ana',sep=', ') # El separador es una␣
↪cadena
```

```
Juana, Valentín, María del Mar, Ana
```

[4]:
```
# Ejemplo para mostrar una barra de progreso:
from time import sleep

sleep(0.4)
print('\r['+'#'*1+' '*9+']  10%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*2+' '*8+']  20%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*3+' '*7+']  30%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*4+' '*6+']  40%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*5+' '*5+']  50%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*6+' '*4+']  60%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*7+' '*3+']  70%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*8+' '*2+']  80%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*9+' '*1+']  90%',sep='',end='',flush=True); sleep(0.4)
print('\r['+'#'*10+'] 100%',sep='',flush=True)
```

```
[##########] 100%
```

2.1 Aplicación de formato a cadenas de caracteres:

- Mecanismo basado en el operador %
- Utilización del método *format*
- Cadenas formateadas (*f-strings*)

2.1.1 Mecanismo basado en el operador %

```
print('Cadena con especificadores de conversión %_0, %_1,..., %_n'  %(valor0,
                                                                valor1,...,valor_n))
```

Un especificador de conversión contiene dos o más caracteres con los siguientes componentes en orden:

1. Carácter %: inicio del especificador
2. Clave (opcional): variable que se desea imprimir, colocada entre paréntesis (útil con diccionarios)
3. Etiqueta (opcional): afecta a la forma en que se muestran algunos tipos de datos
4. Anchura mínima del campo de texto (opcional)
5. Precisión (opcional): `.X`, donde `X` es el número de cifras decimales
6. Modificador de longitud (opcional)
7. Tipo de conversión (tipo de dato que se desea mostrar)

Etiqueta	Significado
`#`	Utilizar un tipo de conversión alternativo
`0`	Si el valor es numérico, se rellena con ceros por la izquierda
`-`	El valor se ajusta hacia la izquierda (predomina sobre la etiqueta `0`)
` `	Insertar un espacio en blanco antes de una cadena vacía o de un valor numérico de signo positivo
`+`	Añade el signo ('+' o '-' al valor numérico) (sobreescribe a la etiqueta)

Tipo de conversión	Significado
`d/i`	Valor entero en formato decimal
`o`	Valor entero en formato octal
`x/X`	Valor entero en formato hexadecimal (minúsculas/mayúsculas)
`e/E`	Valor real en formato exponencial (minúsculas/mayúsculas)
`f/F`	Valor real en formato decimal (minúsculas/mayúsculas)
`c`	Carácter (acepta un valor entero o una cadena con un único carácter)
`r`	Cadena de caracteres (convierte cualquier objeto usando *repr()*)
`s`	Cadena de caracteres (convierte cualquier objeto usando *str()*)
`a`	Cadena de caracteres (convierte cualquier objeto usando *ascii()*)
`%`	Imprime el carácter %

[5]:
```
a=0x1A    # Valor hexadecimal 1Ah=00011010b=26
r=51.123601
str="mi cadena"
```

Valores numéricos y cadena de caracteres

[6]:
```
print('Número entero: %d \nNúmero real: %f \nCadena de caracteres: %s'␣
 ↪%(a,r,str))
```

```
Número entero: 26
Número real: 51.123601
Cadena de caracteres: mi cadena
```

Valores numéricos en campos de 7 caracteres, y precisión de 3 decimales, rellenando con 0s por la izquierda. Cadena en campo de 20 caracteres

[7]:
```
print('Número Entero: %07d \nNúmero real: %07.3f \nCadena de caracteres: %20s'␣
 ↪%(a,r,str))
```

```
Número Entero: 0000026
Número real: 051.124
Cadena de caracteres:            mi cadena
```

Valor entero en formatos decimal con signo, octal y hexadecimal

[8]:
```
print ("Formatos de un valor entero:")
print ("Decimal: a=%+d; -a=%+d" %(a,-a))
print ("Octal: a=%o=%05o \t\t Octal alternativo: a=%#o=%#05o"%(a,a,a,a))
print ("Hexadecimal: a=%x=%08X \t Hexadecimal alternativo: a=%#x=%#08X␣
 ↪"%(a,a,a,a))
```

```
Formatos de un valor entero:
Decimal: a=+26; -a=-26
Octal: a=32=00032                Octal alternativo: a=0o32=0o032
Hexadecimal: a=1a=0000001A       Hexadecimal alternativo: a=0x1a=0X00001A
```

Valor real en formato decimal y exponencial

[9]:
```
print("Formatos de un valor real: ")
print("Decimal: r=%.2f"%(r))
print("Exponencial: r=%.2e=%.2E"%(r,r))
```

```
Formatos de un valor real:
Decimal: r=51.12
Exponencial: r=5.11e+01=5.11E+01
```

2.1.2 Utilización del método *format*

```
cadena.format(cadena_0[,...,cadena_n])
```

[10]:
```
print(f'Me llamo {} y tengo {} años'.format('Julia',14))
```

```
  File "<ipython-input-10-e65b6f9b67ba>", line 1
    print(f'Me llamo {} y tengo {} años'.format('Julia',14))
                 ^
SyntaxError: f-string: empty expression not allowed
```

[11]:
```
valor0='10'
valor1=11
print('Muestro tres valores: {0}, {1} y {2}'.format(valor0,valor1,'hola'))
```

```
Muestro tres valores: 10, 11 y hola
```

[12]:
```
print('Los mismos valores en otro orden: {2}, {0} y {1}'.
 ↪format(valor0,valor1,'hola'))
```

```
Los mismos valores en otro orden: hola, 10 y 11
```

[13]:
```
print('Otra manera de formatear cadenas: {nombre} {apellido}'.
 ↪format(nombre='Andrea',apellido='del Canto'))
```

```
Otra manera de formatear cadenas: Andrea del Canto
```

[14]:
```
print('Se puede mezclar formatos: {nombre} {apellido} tiene {0} años'.
 ↪format(21,nombre='Rogelio',apellido='Rodríguez'))
```

```
Se puede mezclar formatos: Rogelio Rodríguez tiene 21 años
```

2.1.3 Cadenas formateadas (*f-strings*)

- Una cadena formateada comienza con `f` o `F` antes de las comillas.
- Admite el uso de expresiones indicadas entre llaves `{ }`
  `f'Ejemplo: {expresion1}, {expresion2}`
- Se puede indicar un **especificador de formato** a cada expresión después del carácter `:`
  - valor entero (`:xd`): representación con un mínimo de *x* columnas
  - valor decimal (`:x.yf`): *x* columnas, *y* posiciones decimales
  - Rellenar con 0s por la izquierda: `:0xd`, `:0x.yf`
- **Otros modificadores**:
  - `!a` aplica la función *ascii()* (convertir a formato representable en código ASCII)
  - `!s` aplica la función *str()* (convertir a cadena de caracteres)
  - `!r` aplica la función *repr()* (imprime la información que representa a un objeto)

[15]:
```
f'Número entero con 5 columnas y ceros por la izquierda: {-1:05d}'
```

[15]:
```
'Número entero con 5 columnas y ceros por la izquierda: -0001'
```

[16]:
```
f'Valor literal:{5.2622:05.2f}' # Se representa el resultado con al menos 5␣
↪caracteres
```

[16]:
```
'Valor literal:05.26'
```

[17]:
```
valor=5
f'El cuadrado de {valor} es {valor**2}'
```

[17]:
```
'El cuadrado de 5 es 25'
```

[18]:
```
día=15
mes=1
año=2020
print(f'Hoy es: {día:02}/{mes:02}/{año}')
```

```
Hoy es: 15/01/2020
```

[19]:
```
numero_0=23.4
print(f'Número: {numero_0!s}')
```

```
Número: 23.4
```

3 Entrada de datos por teclado (*input*)

Lectura de una cadena de texto introducida desde el teclado

`input([<prompt>])`

Argumento:

> `<prompt>` (optativo): Cadena de caracteres que se muestra para indicar que el sistema permanece a la espera de entrada de datos por teclado

`input` devuelve una cadena de caracteres. En caso necesario, ha de convertirse al formato apropiado (*int()*, *float()*, *complex()*)

[20]:
```
nombre=input('¿Cómo te llamas?')
edad=int(input('¿Cuántos años tienes?'))
altura=float(input('¿Cuánto mides (metros)?'))
cplx=complex(input('Escribe un número complejo (a+bj):'))
```

```
¿Cómo te llamas? Sofía
¿Cuántos años tienes? 8
```

```
¿Cuánto mides (metros)? 1.28
Escribe un número complejo (a+bj): 2+3j
```

[21]:
```
print(f'Hola, {nombre}: tienes {edad} años y mides {altura:.2f} metros')
print(f'Has escrito el número complejo {cplx}')
```

```
Hola, Sofía: tienes 8 años y mides 1.28 metros
Has escrito el número complejo (2+3j)
```

4 Instrucciones de selección

`if`
`if...else`
escalera `elif`

- Permiten ejecutar bloques de código diferentes dependiendo del cumplimiento de una condición
- Una condición es una expresión cuyo resultado puede ser `True` o `False`
- Un bloque se identifica porque todas sus instrucciones están indentadas hacia la derecha con respecto a la propia instrucción de selección

4.1 Instrucciones de selección: *if*

```
if condición:
    # Si condición==True, se ejecutan las instrucciones_T
    instrucción_T0
    instrucción_T1
    ...
    instrucción_Tn
```

[22]:
```
num=float(input('Numerador?'))
den=float(input('Denominador?'))

if den!=0:
    resultado=num/den
    print (f'{num}/{den}={resultado:.2f}')

if den==0:
    print ('Error de división: El denominador es 0')
```

```
Numerador? 5
Denominador? 2
```

```
5.0/2.0=2.50
```

[23]:
```
'''Comprobar el signo del resultado de una operación entre enteros'''
a=int(input('valor entero a?'))
b=int(input('valor entero b?'))

if a-b>0:
    print(f'{a}-{b} es un valor positivo')
if a-b<0:
    print(f'{a}-{b} es un valor negativo')
if a-b==0:
    print(f'{a}-{b} es cero')
```

```
valor entero a? 5
valor entero b? 6
```

```
5-6 es un valor negativo
```

[24]:
```
'''Comprobar si un entero es par o impar'''
valor=int(input('Escribe un número entero:'))

if valor%2==0: # comprueba si el resto de la división entre 2 es 0
    print (f'{valor} es un número PAR')
if valor%2!=0: # comprueba si el resto de la división entre 2 NO es 0
    print (f'{valor} es un número IMPAR')
```

```
Escribe un número entero: 53
```

```
53 es un número IMPAR
```

4.2 Instrucciones de selección: *if…else*

```
if condición:
    # Si condición==True, se ejecutan las instrucciones_T
    instrucción_T0
    instrucción_T1
    ...
    instrucción_Tn
else:
    # Si condición==False, se ejecutan las instrucciones_F
    instrucción_F0
    instrucción_F1
    ...
    instrucción_Fn
```

[25]:
```
'''División de números reales'''

num=float(input('Numerador?'))
den=float(input('Denominador?'))
```

```
if den!=0:
    resultado=num/den
    print (f'{num}/{den}={resultado:.2f}')
else:
    print ('Error de división: El denominador es 0')
```

```
Numerador? 4
Denominador? 2.1
```

```
4.0/2.1=1.90
```

[26]:
```
'''División de números complejos'''
num=complex(input('Numerador?'))
den=complex(input('Denominador?'))

if den!=0:
    resultado=num/den
    print (f'{num}/{den}={resultado}')
else:
    print ('Error de división: El denominador es 0')
```

```
Numerador? 2+1.5j
Denominador? 1-j
```

```
(2+1.5j)/(1-1j)=(0.25+1.75j)
```

[27]:
```
'''Comprobación del derecho a voto en España'''

edad=int(input('Cuántos años tienes?'))

if edad>=18:
    derecho_Voto=True
else:
    derecho_Voto=False

print(f'Tu derecho a votar es {derecho_Voto}')
```

```
Cuántos años tienes? 17
```

```
Tu derecho a votar es False
```

[28]:
```
'''Resolución ecuación de primer grado ax+b=0'''

print('Escribe los valores de los coeficientes de la ecuación ax+b=0')
a=float(input('a?'))
b=float(input('b?'))

if a!=0:
    sol=-b/a
```

```
    print (f'La solución de {a}*x{b:+}=0 es x={sol:.3}')
else:
    print (f'{a}*x{b:+}=0 no es una ecuación')
```

```
Escribe los valores de los coeficientes de la ecuación ax+b=0

a? -3
b? 2

La solución de -3.0*x+2.0=0 es x=0.667
```

[29]:
```
'''Resolución ecuación de segundo grado ax^2+bx+c=0'''
''' x0=(-b+sqrt(b^2-4ac))/(2a)'''
''' x1=(-b-sqrt(b^2-4ac))/(2a)'''

from math import sqrt   #importar la función raíz cuadrada

print('Escribe los valores de los coeficientes de la ecuación ax^2+bx+c=0')
a=float(input('a?'))
b=float(input('b?'))
c=float(input('c?'))

if a!=0:
    x0=(-b+(sqrt(b**2-4*a*c)))/(2*a)
    x1=(-b-(sqrt(b**2-4*a*c)))/(2*a)
    print (f'Soluciones ecuación 2o grado: x0={x0:.2}; x1={x1:.2}')
else:
    if b!=0:
        sol=-c/b
        print (f'Solución ecuación 1er grado: x={sol:.2}')
    else:
        print ('Los coeficientes no corresponden a una ecuación')
```

```
Escribe los valores de los coeficientes de la ecuación ax^2+bx+c=0

a? 1
b? 4
c? 0.1

Soluciones ecuación 2o grado: x0=-0.025; x1=-4.0
```

Nota: Comprueba que al resolver $x^2 + 2x + 3 = 0$ se genera un error ¿Por qué? (*soluciones complejas*)

[31]:
```
# '''Resolución ecuación de segundo grado ax^2+bx+c=0'''
'''(versión mejorada)'''
''' x0=(-b+sqrt(b^2-4ac))/(2a)'''
''' x1=(-b-sqrt(b^2-4ac))/(2a)'''

from math import sqrt   #importar la función raíz cuadrada
```

```
print('Escribe los valores de los coeficientes de la ecuación ax^2+bx+c=0')
a=float(input('a?'))
b=float(input('b?'))
c=float(input('c?'))

if a!=0:
    if b**2-4*a*c>=0:
        x0=(-b+(sqrt(b**2-4*a*c)))/(2*a)
        x1=(-b-(sqrt(b**2-4*a*c)))/(2*a)
        if (x0!=x1):
            print (f'Soluciones ecuación 2o grado: x0={x0:.2}; x1={x1:.2}')
        else:
            print (f'Soluciones ecuación 2o grado: x0=x1={x0:.2}')
    else:
        print ('Las soluciones de la ecuación son valores complejos')
else:
    if b!=0:
        sol=-c/b
        print (f'Solución ecuación 1er grado: x={sol:.2}')
    else:
        print ('Los coeficientes no corresponden a una ecuación')
```

```
Escribe los valores de los coeficientes de la ecuación ax^2+bx+c=0

a? 1
b? -2
c? 1

Soluciones ecuación 2o grado: x0=x1=1.0
```

Ejemplo: Calcular el mayor de 3 valores

[32]:
```
'''Obtener el mayor de tres valores'''
a=float(input('a?'))
b=float(input('b?'))
c=float(input('c?'))

if a>b:
    if a>c:
        mayor=a
    else:
        mayor=c
else:
    if b>c:
        mayor=b
    else:
```

```
        mayor=c
print (f'El valor más alto es {mayor}')
```

```
a? 2
b? 3
c? -1
```

```
El valor más alto es 3.0
```

[33]:
```
'''Obtener el mayor de tres valores (alternativa)'''
a=float(input('a?'))
b=float(input('b?'))
c=float(input('c?'))

if a>b and a>c:
    mayor=a
else:
    if b>c:
        mayor=b
    else:
        mayor=c
print (f'El valor más alto es {mayor}')
```

```
a? 4
b? 3
c? 5
```

```
El valor más alto es 5.0
```

[34]:
```
'''Obtener el mayor de tres valores (alternativa 2)'''
a=float(input('a?'))
b=float(input('b?'))
c=float(input('c?'))

mayor=a
if b>mayor:
    mayor=b
if c>mayor:
    mayor=c
print (f'El valor más alto es {mayor}')
```

```
a? 3
b? 4
c? 2
```

```
El valor más alto es 4.0
```

Ejemplo: Programa con menú de opciones

[35]:
```
cadena=input('Escribe una frase:')
```

```
Escribe una frase: Esta es mi frase
```

[36]:
```
titulo='MENU DE OPERACIONES POSIBLES'
num_letras_titulo=len(titulo) # Calcula la longitud de la cadena titulo
```

[37]:
```
print (titulo)
print ('='*num_letras_titulo)
print ('a. Convertir a Mayúsculas')
print ('b. Convertir a minúsculas')
print ('c. Convertir a tipo Título')
print ('Resto de opciones: contar número de letras de la frase')
```

```
MENU DE OPERACIONES POSIBLES
============================
a. Convertir a Mayúsculas
b. Convertir a minúsculas
c. Convertir a tipo Título
Resto de opciones: contar número de letras de la frase
```

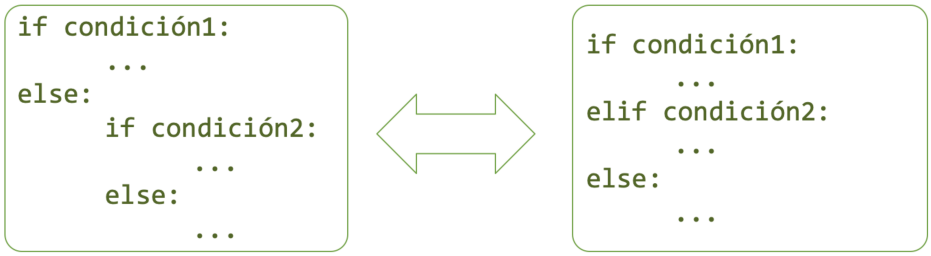
[38]:
```
operacion=input('Elija su opción:')

if operacion=='a':
    print (cadena.upper())
else:
    if operacion=='b':
        print(cadena.lower())
    else:
        if operacion=='c':
              print(cadena.title())
        else:
              print(f'Tu frase tiene {len(cadena)} caracteres')
```

```
Elija su opción: c

Esta Es Mi Frase
```

4.3 Instrucciones de selección: escalera *elif*



Ejemplo: gestión de opciones de menú utilizando la escalera *elif*

[39]:
```
operacion=input('Elija su opción:')

if operacion=='a':
    print (cadena.upper())
elif operacion=='b':
    print(cadena.lower())
elif operacion=='c':
    print(cadena.title())
else:
    print(f'Tu frase tiene {len(cadena)} caracteres')
```

```
Elija su opción: b
```

```
esta es mi frase
```

5 Estructuras de repetición: *while*

```
while condición:
    # Si condición es True, se ejecutan las instrucciones T
    instruccion_T0
    instrucción_T1
    ...
    instrucción_Tn
```

- Si condición es *False*, entonces no se ejecutan las instrucciones T (0 iteraciones)
- Para que el proceso iterativo finalice, el valor de condición ha de cambiar a *False* mientras se ejecutan las instrucciones T

[40]:
```
i=0
while i<5:
    print (i,end='::')
    i+=1
```

```
0::1::2::3::4::
```

5.1 Control por contador vs control por centinela

Control por contador:

- Una variable (contador) se encarga de contar el número de iteraciones
- Se utiliza cuando el número de repeticiones es conocido

Control por centinela:

- El proceso iterativo finaliza cuando una variable (centinela) toma el valor *False*
- Se utiliza cuando el número de repeticiones es desconocido

Ejemplo de control por contador: tabla de multiplicar (5x)

[41]:
```
j=1 # variable contador
while j<=10:
    print (f'5 * {j} = {5*j}')
    j+=1
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Ejemplo de control por centinela: bucle con número de repeticiones decidido por el usuario

[42]:
```
n=int(input('Escribe 0 para finalizar: '))

while n!=0:  # n es la variable centinela
    n=int(input('¿No quieres que finalice aún? (0:salir)'))

print ('¡Por fin he salido del bucle!')
```

```
Escribe 0 para finalizar:  3
¿No quieres que finalice aún? (0:salir) 2
¿No quieres que finalice aún? (0:salir) 0

¡Por fin he salido del bucle!
```

¿Qué problema tiene el siguiente programa?

```
letra=input('Pulsa [C] para continuar:')

while letra!='C' or letra!='c': #letra es la variable centinela
    print ('¡INCORRECTO!')
    letra=input('Pulsa [C] para continuar:')

print ('¡Muchas gracias!')
```

Solución: Hay un **bucle infinito**. La comparación debe hacerse con and

[43]:
```
letra=input('Pulsa [C] para continuar:')

while letra!='C' and letra!='c': #letra es la variable centinela
    print ('¡INCORRECTO!')
    letra=input('Pulsa [C] para continuar:')

print ('¡Muchas gracias!')
```

```
Pulsa [C] para continuar: s
¡INCORRECTO!
Pulsa [C] para continuar: add
¡INCORRECTO!
Pulsa [C] para continuar: c
¡Muchas gracias!
```

Ejemplo: cálculo de un sumatorio

$$\sum_{i=0}^{i=100} (i+1)^2$$

El número de repeticiones es conocido → control por contador

[44]:
```
i=0 # valor inicial del contador
suma=0 # Valor inicial de la suma: elemento neutro de la suma

while i<=100:
    elemento_i=(i+1)**2
    suma+=elemento_i # actualización de la suma parcial
    print (f'elemento({i})={elemento_i}\t suma({i})={suma}')
    i+=1 # actualización del contador
print (f'Suma total={suma}')
```

```
elemento(0)=1    suma(0)=1
elemento(1)=4    suma(1)=5
elemento(2)=9    suma(2)=14
elemento(3)=16   suma(3)=30
elemento(4)=25   suma(4)=55
...
elemento(99)=10000       suma(99)=338350
elemento(100)=10201      suma(100)=348551
Suma total=348551
```

Ejemplo: cálculo de $n!$ (solución iterativa)

$$n! = \prod_{j=1}^{j=n} j$$

El número de repeticiones es conocido → control por contador

[45]:
```
num=int(input('Introduzca un número entero mayor que 0:')) # Valor inicial del␣
 ↪contador
prod=1 # Valor inicial del resultado: elemento neutro del producto

j=num
while j>=1:
    prod*=j
    print (f'{j}', end=' ')
    if j!=1:
        print ('*', end=' ')
    j-=1

print (f'= {prod}')
```

```
Introduzca un número entero mayor que 0: 5
```

```
5 * 4 * 3 * 2 * 1 = 120
```

¿Qué ocurre cuando se calcula el valor $0!$?

Cálculo de $n!$ con control del parámetro solicitado por teclado

[46]:
```
num=int(input('Introduzca un número entero mayor que 0:')) # Valor inicial del␣
 ↪contador

# Mientras que num sea menor o igual que 0, se vuelve a pedir su valor:
while num<=0:
    num=int(input('Introduzca un número entero mayor que 0:'))
```

```
prod=1 # Valor inicial del resultado: elemento neutro del producto

j=num
while j>=1:
    prod*=j
    print (f'{j}', end=' ')
    if j!=1:
        print ('*', end=' ')
    j-=1

print (f'= {prod}')
```

```
Introduzca un número entero mayor que 0: 0
Introduzca un número entero mayor que 0: -2
Introduzca un número entero mayor que 0: 3
```

```
3 * 2 * 1 = 6
```

Ejemplo: Elección de una opción de menú

Condición de funcionamiento: El menú se vuelve a mostrar si no se elije una opción válida

[47]:
```
titulo='MENÚS DISPONIBLES'
longitud_titulo=len(titulo)
plato_0='a. Burger Especial'
plato_1='b. Burguer Extra'
plato_2='c. Burguer de Pollo'
plato_3='d. Ensalada'
plato_4='e. Croquetas de Jamón'
plato_5='f. Especial de Brócoli'
```

[48]:
```
opcion='' # Inicialmente no hay ninguna opción seleccionada

while opcion!='a' and opcion!='b' and opcion!='c' and \
                      opcion!='d' and opcion!='e' and opcion!='f':
    print (titulo)
    print ('='*longitud_titulo)
    print (plato_0)
    print (plato_1)
    print (plato_2)
    print (plato_3)
    print (plato_4)
    print (plato_5)
    opcion=input('Seleccione su menú:')

print (f'Ha elegido el menú {opcion}. ¡Muchas gracias!')
```

```
MENÚS DISPONIBLES
=================
a. Burger Especial
b. Burguer Extra
c. Burguer de Pollo
d. Ensalada
e. Croquetas de Jamón
f. Especial de Brócoli

Seleccione su menú: 0

MENÚS DISPONIBLES
=================
a. Burger Especial
b. Burguer Extra
c. Burguer de Pollo
d. Ensalada
e. Croquetas de Jamón
f. Especial de Brócoli

Seleccione su menú: f

Ha elegido el menú f. ¡Muchas gracias!
```

Nota: Las estructuras de datos de tipo *lista* permiten simplificar la comprobación de las opciones válidas:

```
lista_opciones=['a','b','c','d','e','f']
```

[49]:
```
opcion='' # Inicialmente, no se ha elegido ninguna opción

# Esta lista de opciones incluye también las letras mayúsculas:
lista_opciones=['a','b','c','d','e','f', 'A','B','C','D','E','F']

while opcion not in lista_opciones:
    print (titulo)
    print ('='*longitud_titulo)
    print (plato_0)
    print (plato_1)
    print (plato_2)
    print (plato_3)
    print (plato_4)
    print (plato_5)
    opcion=input('Seleccione su menú:')

print (f'Ha elegido el menú {opcion.lower()}. ¡Muchas gracias!')
```

```
MENÚS DISPONIBLES
=================
a. Burger Especial
b. Burguer Extra
```

```
c. Burguer de Pollo
d. Ensalada
e. Croquetas de Jamón
f. Especial de Brócoli

Seleccione su menú: a

Ha elegido el menú a. ¡Muchas gracias!
```

Ejemplo: Menú de opciones que se repite hasta que se seleccione la opción *Salir*

[50]:
```
cadena=input('Escribe una frase:')
titulo='Qué deseas hacer con tu frase?'
longitud_titulo=len(titulo)
```

```
Escribe una frase: ¡pRueba Con eSTa caDENa!
```

[51]:
```
opcion=''

while opcion!='0': # Nota: '0' porque input devuelve una cadena de texto
    print (titulo)
    print ('\t 1. Escribir una frase nueva')
    print ('\t 2. Convertir a mayúsculas')
    print ('\t 3. Convertir a minúsculas')
    print ('\t 4. Convertir a tipo título')
    print ('\t 0. Terminar')
    opcion=input('Elige tu opción:')

    if opcion=='1':
        cadena=input('Escribe una frase:')
    elif opcion=='2':
        print (cadena.upper())
    elif opcion=='3':
        print (cadena.lower())
    elif opcion=='4':
        print (cadena.title())
```

```
Qué deseas hacer con tu frase?
         1. Escribir una frase nueva
         2. Convertir a mayúsculas
         3. Convertir a minúsculas
         4. Convertir a tipo título
         0. Terminar

Elige tu opción: 4

¡Prueba Con Esta Cadena!
Qué deseas hacer con tu frase?
         1. Escribir una frase nueva
```

```
        2. Convertir a mayúsculas
        3. Convertir a minúsculas
        4. Convertir a tipo título
        0. Terminar

Elige tu opción: 0
```

6 Estructuras de repetición: *for*

```
for variable in conjunto_de_valores:
    instrucción_0
    instrucción_2
    ...
    instrucción_N
```

Si conjunto_de_valores=[elemento_0, elemento_1,..., elemento_n]

- **Número de iteraciones: n+1** (número de elementos de conjunto_de_valores)
- **Valor de *variable* en cada iteración**:
  - Iteración 0: elemento_0 (primer elemento de conjunto_de_valores)
  - Iteración 1: elemento_1 (segundo elemento de conjunto_de_valores)
    ...
  - Iteración n+1: elemento_n (último elemento del conjunto de valores)

[52]:
```
lista_profes=['Eugene','Diana','Juan Carlos','Conchi','María Rosa',\
              'Camino','Petri','Jesús']

for nombre in lista_profes:
    print ('\t %s' %(nombre))
```

```
        Eugene
        Diana
        Juan Carlos
        Conchi
        María Rosa
        Camino
        Petri
        Jesús
```

Se puede conocer el número de la iteración en curso convirtiendo una lista a un tipo de datos enumerado:

`enumerate(lista)`

[53]:
```
for indice,nombre in enumerate(lista_profes):
    print ('%d -> %s'%(indice,nombre))
```

```
0 -> Eugene
1 -> Diana
```

```
2 -> Juan Carlos
3 -> Conchi
4 -> María Rosa
5 -> Camino
6 -> Petri
7 -> Jesús
```

6.1 Secuencias de números enteros

Números enteros consecutivos:

`range(v0,v1)` Incluye a v0, pero no a v1

Números enteros con paso k:

`range(v0,v1,k)` Incluye a v0, pero no a v1

- Una secuencia es válida como conjunto de valores para un bucle de tipo for
- Una secuencia se puede convertir a lista: `list(secuencia)`

[54]:
```
secuencia1=range(10,16)
print(list(secuencia1))
```

```
[10, 11, 12, 13, 14, 15]
```

[55]:
```
secuencia2=range(1,21,3)
print(list(secuencia2))
```

```
[1, 4, 7, 10, 13, 16, 19]
```

[56]:
```
secuencia3=range(20,-20,-5)
print(list(secuencia3))
```

```
[20, 15, 10, 5, 0, -5, -10, -15]
```

Ejemplo: tabla de multiplicar con bucles for

[57]:
```
n=int(input('De qué número quieres la tabla de multiplicar'))

for i in range(1,11): # i varía entre 1 y 10, ambos incluidos
    s=n*i # Calcula n*i
    print (f'{n}*{i}={s}')
```

```
De qué número quieres la tabla de multiplicar 5
```

```
5*1=5
5*2=10
5*3=15
5*4=20
```

```
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
5*10=50
```

7 Estructuras de repetición: rotura de bucles (*break*)

La instrucción `break` sale de un bucle y continúa con la ejecución del programa

[58]:
```
for i in range(0,101):
    print (i, end='')
    if i>=20:
        break
    print (', ',end='')
print ('\nFIN')
```

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
FIN
```

8 Anidamiento de estructuras de repetición

```
for v1 in conjunto_1:
    # v1 va tomando los valores de conjunto_1
    for v2 in conjunto_2:
        # v2 va tomando los valores de conjunto_2
```

[59]:
```
for i in ['a','b','c']:
    for j in [0,1,2,3,4,5]:
        print ('%s%d'%(i,j), end=' ')
    print ('\n')
```

```
a0 a1 a2 a3 a4 a5

b0 b1 b2 b3 b4 b5

c0 c1 c2 c3 c4 c5
```

Ejemplo: Imprimir varias tablas de multiplicar

[60]:
```
for i in [2,4,6]:
    print (f'Tabla del {i}')
    for j in range(1,11):
        print (f'{i}*{j}={i*j}')
```

```
Tabla del 2
2*1=2
2*2=4
...
2*10=20
Tabla del 4
4*1=4
4*2=8
...
4*10=40
Tabla del 6
6*1=6
6*2=12
...
6*10=60
```

9 Tratamiento de excepciones: *try...except*

La estructura de control `try ... except` simplifica la captura y el tratamiento de excepciones:

```
try: # Intenta ejecutar este bloque, susceptible de generar excepciones
    instrucción_P1
    ...
    instrucción_Pn
except: # En caso de error de ejecución, ejecuta el bloque siguiente
    instrucción_E1
    ...
    instrucción_E2
```

Se pueden capturar los diferentes tipos de excepciones que captura el sistema operativo:

- `ValueError`
- `ZeroDivisionError`

  ...

[61]:
```
num=float(input('¿Numerador?'))
den=float(input('¿Denominador?'))
```

```
try:
    sol=num/den
    print (f'{num}/{den}={sol:.2}')
except:
    if den==0:
           print ("Lo siento, se trata de una división entre 0")
```

```
¿Numerador? 1.111
¿Denominador? 3.1416
```

```
1.111/3.1416=0.35
```

Ejemplo: Ecuación de segundo grado

[62]:
```
'''Resolución ecuación de segundo grado ax^2+bx+c=0'''
'''(versión con try...except)'''
''' x0=(-b+sqrt(b^2-4ac))/(2a)'''
''' x1=(-b-sqrt(b^2-4ac))/(2a)'''

from math import sqrt   #importar la función raíz cuadrada

print('Escribe los valores de los coeficientes de la ecuación ax^2+bx+c=0')
a=float(input('a?'))
b=float(input('b?'))
c=float(input('c?'))

try:
    x0=(-b+(sqrt(b**2-4*a*c)))/(2*a)
    x1=(-b-(sqrt(b**2-4*a*c)))/(2*a)
    if (x0!=x1):
        print (f'Soluciones ecuación 2o grado: x0={x0:.2}; x1={x1:.2}')
    else:
        print (f'Soluciones ecuación 2o grado: x0=x1={x0:.2}')
except:
    if a==0 and b!=0:  # Es una ecuación de primer grado
        sol=-c/b
        print (f'Solución ecuación 1er grado: x={sol:.2}')
    elif b**2-4*a*c<0:
        print ('Las soluciones de la ecuación son valores complejos')
    else:
        print ('Los coeficientes no corresponden a una ecuación')
```

```
Escribe los valores de los coeficientes de la ecuación ax^2+bx+c=0
```

```
a? 1
b? 2
c? -2
```

```
Soluciones ecuación 2o grado: x0=0.73; x1=-2.7
```

El programa anterior genera excepciones diferentes cuando intenta resolver las siguientes dos ecuaciones:

$4x^2 + x + 4 = 0 \rightarrow$ `ValueError`

$0x^2 + 2x + 1 = 0 \rightarrow$ `ZeroDivisionError`

[63]:
```
a=4; b=1; c=4
x0=(-b+(sqrt(b**2-4*a*c)))/(2*a)
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-63-884ff32ac803> in <module>
  1 a=4; b=1; c=4
----> 2 x0=(-b+(sqrt(b**2-4*a*c)))/(2*a)


ValueError: math domain error
```

[64]:
```
a=0; b=2; c=1
x0=(-b+(sqrt(b**2-4*a*c)))/(2*a)
```

```
---------------------------------------------------------------------------

ZeroDivisionError                         Traceback (most recent call last)

<ipython-input-64-76074048f98c> in <module>
  1 a=0; b=2; c=1
----> 2 x0=(-b+(sqrt(b**2-4*a*c)))/(2*a)


ZeroDivisionError: float division by zero
```

Ejemplo: Ecuación de segundo grado capturando excepciones *ZeroDivisionError* y *ValueError*

[65]:
```
'''Resolución ecuación de segundo grado ax^2+bx+c=0'''
'''(versión con try...except capturando ZeroDivisionError y ValueError)'''
''' x0=(-b+sqrt(b^2-4ac))/(2a)'''
''' x1=(-b-sqrt(b^2-4ac))/(2a)'''

from math import sqrt   #importar la función raíz cuadrada

print('Escribe los valores de los coeficientes de la ecuación ax^2+bx+c=0')
a=float(input('a?')); b=float(input('b?')); c=float(input('c?'))

try:
    x0=(-b+(sqrt(b**2-4*a*c)))/(2*a)
    x1=(-b-(sqrt(b**2-4*a*c)))/(2*a)
    if (x0!=x1):
        print (f'Soluciones ecuación 2o grado: x0={x0:.2}; x1={x1:.2}')
    else:
        print (f'Soluciones ecuación 2o grado: x0=x1={x0:.2}')
except ZeroDivisionError:
    if a==0 and b!=0:  # Es una ecuación de primer grado
        sol=-c/b
        print (f'Solución ecuación 1er grado: x={sol:.2}')
    else:
        print ('Los coeficientes no corresponden a una ecuación')
except ValueError:
    if b**2-4*a*c<0:
        print ('Las soluciones de la ecuación son valores complejos')
except: # Resto de posibles excepciones no especificadas
        print ('Se ha detectado un error en tiempo de ejecución')
```

```
Escribe los valores de los coeficientes de la ecuación ax^2+bx+c=0

a? 4
b? 1
c? 4

Las soluciones de la ecuación son valores complejos
```