Herramientas de programación en Python

1. Fundamentos de programación

1.3 Estructuras (Composiciones) Algorítmicas Secuenciales y Alternativas

1. Secuenciales
2. Alternativas o condicionales

Pedro Gomis

pedro.gomis@upc.edu

- La **programación estructurada** contempla el uso de ***estructuras*** o ***composiciones:***
  - **secuenciales**,
  - de **selección** o **alternativas** e
  - **iterativas**
- Con ellas se puede resolver cualquier función computable.
- Siguiendo la tesis de Alan Turing, se puede resolver cualquier función o algoritmo que pueda ser calculado por un computador.
- Al conjunto de sentencias elementales (asignación de valores o expresiones a variables, input, print, etc.) se le añaden las estructuras o composiciones algorítmicas que comprenden secuencias, alternativas e iteraciones o bucles como parte del código de los **programas** y **funciones**

Composición secuencial

- Es la más simple en programación pues se trata de una serie de acciones, instrucciones o sentencias que se procesan secuencialmente en bloque una a una.

Estructura secuencial

Pseudocódigo	Python	Pascal	C
inicio	Instrucción 1	**begin**	{
Instrucciones	Instrucción 2	Instrucciones	Instrucciones
…	…	…	…
fin	Instrucción n	**end**	}

Ejemplo: Programa que convierte grados Celsius a Fahrenheit

```
# Programa que convierte ºC a ºF
C = float(input("Entra temperatura en ºC: "))
F = 1.8*C + 32
print(C, "ºC equivale a",  F, "ºF")
```

```
Entra temperatura en ºC: 27
27.0 ºC equivale a  80.6 ºF
>>>
```

Composición secuencial

Ejemplo: calculo del área y el perímetro de un círculo a partir de su radio

```
# Programa que calcula perímetro y área de un círculo
from math import pi
r = float(input('Introduce el radio del círculo: '))
per = 2*pi*r
area = pi*r**2
print('Perímetro =', per)
print('Área =', area)
```

```
Introduce el radio del círculo: 1.2
Perímetro = 7.5398223686155035
Área = 4.523893421169302
```

Composición secuencial

Ejemplo: calculo del área y el perímetro de un círculo a partir de su radio, con opciones para el manejo del número de decimales

```
# Programa que calcula perímetro y área de un círculo
from math import pi
r = float(input('Introduce el radio del círculo: '))
per = 2*pi*r
area = pi*r**2
# funcionalidad de printf en Python*
print('Perímetro = %.2f', % per)
# función interna round()
print('Área =', round(area, 2))
```

```
Introduce el radio del círculo: 1.2
Perímetro = 7.54
Área = 4.52
```

* Ver detalles del uso de printf en Python en:
http://www.python-course.eu/python3_formatted_output.php

Composición secuencial

```
# Programa para convertir segundos a días, horas, minutos, y segundos
s = int(input("Entra segundos: "))
print(s, 'segundos son:')
m = s//60
s = s%60
h = m//60
m = m%60
d = h // 24
h = h % 24
print(d, "días,",h,"horas,",m, "minutos y", s, "segundos")
```

```
>>> Entra segundos: 3600
3600 segundos son:
0 días, 1 horas, 0 minutos y 0 segundos
```

```
>>> Entra segundos: 400000
400000 segundos son:
4 días, 15 horas, 6 minutos y 40 segundos
```

Composición secuencial

```
# Programa para convertir segundos a horas, minutos, y segundos
def pasar_s_a_hms(s):
    '''convertir segundos a horas, minutos, y segundos. Ejemplos:
    >>> pasar_s_a_hms(61)
    (0, 1, 1)
    >>> pasar_s_a_hms(60)
    (0, 1, 0)
    >>> pasar_s_a_hms(3600)
    (1, 0, 0)
    >>> pasar_s_a_hms(123456)
    (34, 17, 36)
    '''
    m = s//60
    s = s%60
    h = m//60
    m = m%60
    return h, m, s
if __name__ == "__main__":
    print(pasar_s_a_hms(10))
    print(pasar_s_a_hms(75))
    print(pasar_s_a_hms(3699))
    import doctest
    doctest.testmod()
```

```
(0, 0, 10)
(0, 1, 15)
(1, 1, 39)
>>>
```

Composición secuencial

Ejemplo:

```
from math import log10
def  ganancia_dB(x, y):
    """
     Calcula ganancia en dB: 20log(y/x)
    """
    gain = y/x
    dB = 20*log10(gain)
    return dB
Vi = 10
Vo = 1000
GdB = ganancia_dB(Vi, Vo)
print('Ganancia =', GdB,'dB')
```

- **Nombre** de la función: ganancia_dB, sus **parámetros** son x e y, la palabra reservada **return** indica que la función es **productiva *o fruitful function*** (devuelve un resultado).
- Al ***devolver*** un resultado de tipo dato simple, como float en esta función, lo pudiéramos usar en una expresión aritmética:

```
Ganancia = 40.0 dB
```

Composición secuencial

Bisiesto. Escribe un programa que incluya la función **bisiesto(n)**, que indique si un año es o no es un año bisiesto (devuelve valor **Booleano**).
Después de la *reforma gregoriana*, los años bisiestos son los múltiples de cuatro que no terminan con dos ceros, y también los años que terminan con dos ceros que, después de eliminar estos dos ceros, son divisibles por cuatro. Así, 1800 y 1900, aunque eran múltiplos de cuatro, no eran años bisiestos; Por el contrario, 2000 fue un año bisiesto. Dicho en formalismo lógico: un año bisiesto es divisible por 400 o bien divisible por 4 exceptuando los divisibles por 100.

```
def bisiesto(A):
    """Función que devuelve si un año es bisiesto o no
    >>> bisiesto(2018)
    False
    >>> bisiesto(2020)
    True
    """
    return A % 400 == 0 or (A % 4 == 0 and A % 100 != 0)

year = int(input("Entra un año: "))
if bisiesto(year):
    print(year,'es bisiseto')
else:
    print(year,'NO es bisiseto')
```

Composición secuencial

Cambio de monedas. Diseña una función **cambio_monedas**(cent) que reciba una cierta cantidad de **céntimos** (c) y retorne su equivalente en el mínimo número de monedas de curso legal (2 euros, 1 euro, 50 céntimos, 20 céntimos, 10 céntimos, 5 céntimos, 2 céntimos y 1 céntimo).
La función recibe un valor cent que representa una cantidad de céntimos y devuelve los valores de las monedas legales desde 2 euros hasta 1 céntimo

```
def cambio_monedas2(c):
    e2 = c // 200
    c = c % 200
    e = c // 100
    c = c % 100
    c50 = c // 50
    c = c % 50
    c20 = c // 20
    c = c % 20
    c10 = c // 10
    c = c % 10
    c5 = c // 5
    c = c % 5
    c2 = c // 2
    c = c % 2
    return e2, e, c50, c20, c10, c5, c2, c
```

Con asignaciones múltiples

```
def cambio_monedas(c):
    e2,c = c // 200, c % 200
    e, c = c // 100, c % 100
    c50, c = c // 50, c % 50
    c20, c = c // 20, c % 20
    c10, c = c // 10, c % 10
    c5, c = c // 5, c % 5
    c2, c = c // 2, c % 2
    return e2, e, c50, c20, c10, c5, c2, c
```

Composiciones alternativas

- La estructura o composición que selecciona ejecutar un conjunto de instrucciones u otro es la ***alternativa*** o ***condicional***.
- Si se cumple como ***cierta*** una expresión lógica entonces el programa realiza una acción o bloque de secuencia de instrucciones. Si, opcionalmente, la condición es ***falsa*** entonces el programa realiza otro bloque de instrucciones.

Estructura alternativa simple o condicional

```
if condición:
    secuencia_de_instrucciones
```

- Ejemplo de de hallar el valor absoluto de un número :

```
x = float(input("Entra un número: "))
if x < 0:
    x = -x
print('El valor absoluto es:',x)
```

Composiciones alternativas

Estructura alternativa simple o condicional

- Ejemplo de cubrir saldo de cuenta corriente descubierto, desde cta. ahorros:

```
#Balance financiero
CuentaAhorros = 10000
saldo = float(input("Cuál es el saldo de la cuenta corriente?: "))
if saldo < 0:
    transferir = -saldo
    CuentaAhorros = CuentaAhorros - transferir
    saldo = saldo + transferir
print('Fondos cuenta de ahorro:', CuentaAhorros)
```

```
Cuál es el saldo de la cuenta corriente?: -350
Fondos cuenta de ahorro: 9650.0
>>>
```

Composiciones alternativas

Estructura alternativa doble (if – else)

- La sintaxis es:

```
if condición:
    secuencia_de_instrucciones_condicion_cierta
else:
    secuencia_de_instrucciones_condicion_falsa
```

- Por ejemplo, resolver ecuación de 1º grado, evitando posible división por 0

```
# Halla solución x, de ecuación ax + b = 0
print('Programa que halla el valor de x de la ecuación: ax + b= 0')
a = float(input('a= '))
b = float(input('b= '))
if a != 0:
    x = -b/a
    print('x =', x)
else:
    print('No es posible dividir por cero')
```

Composiciones alternativas

Estructura alternativa doble (if – else)

- El programa de conversión de grados Celsius a Fahrenheit, visto previamente, se puede mejorar advirtiendo la posible inclusión de una temperatura irreal por debajo del cero absoluto:

```
# Programa que convierte ºC a ºF
C = float(input("Entra temperatura en ºC: "))
if C < -273.15:
    print('Temperatura en ºC irreal por debajo del 0 absoluto!!')
else:
    F = 1.8*C + 32
    print(C, 'ºC equivale a',  F, 'ºF')
```

Composiciones alternativas

Estructura alternativa doble (if – else)

- Función **paridad**, que devuelve un ***string*** que indica si un número es par o impar:

```
def paridad(num):
    '''Comprueba si un número es par o impar
    Ejemplos:
    >>> paridad(4)
    'par'
    >>> paridad(5)
    'impar'
    '''
    if num%2 == 0:
        res = "par"
    else:
        res = "impar"
    return res
if __name__ == "__main__":
    print(paridad(3))
    print(paridad(10))
    import doctest
    doctest.testmod()
```

Composiciones alternativas

Ejemplo: Paso de cronómetro

```
import doctest
def paso(h, m, s):
    """Retorna siguiente valor de horas,
    minutos y segundos de un cronómetro

    Ejemplos:
    >>> paso(0,0,0)
    (0, 0, 1)
    >>> paso(0,59,59)
    (1, 0, 0)
    >>> paso(23,59,59)
    (0, 0, 0)
    """
    s += 1
    if s == 60:
        s = 0
        m += 1
        if m == 60:
            m = 0
            h += 1
            if h == 24:
                h = 0
    return h, m, s
```

```
if __name__ == "__main__":
    print(paso(0, 0, 0))
    print(paso(0, 0, 59))
    print(paso(0, 59, 59))
    print(paso(23, 59, 59))
    doctest.testmod()
```

```
(0, 0, 1)
(0, 1, 0)
(1, 0, 0)
(0, 0, 0)
>>>
```

""" Añadir 1 segundo a (h, m, s) y devolver nueva hora.
Escriba una función add1sec (h, m, s) que reciba como argumento la hora del reloj, expresada en horas, minutos y segundos, y devuelva, nuevamente en horas, minutos y segundos, la hora del reloj aumentada en un segundo.
Los argumentos recibidos cumplen las condiciones previas h <24, m <60 y s <60, y el tiempo de reloj resultante debe obedecer a las mismas desigualdades.
"""

```
def add1sec(h,m,s):
    """
    >>> add1sec(0, 1, 2)
    (0, 1, 3)
    >>> add1sec(3, 5, 9)
    (3, 5, 10)
    >>> add1sec(19, 45, 59)
    (19, 46, 0)
    >>> add1sec(12, 59, 59)
    (13, 0, 0)
    """
    s = s + 1
    if s == 60:
        s = 0
        m = m + 1
        if m == 60:
            m = 0
            h = (h + 1) % 24
    return h, m, s
```

```
if __name__ == "__main__":
    import doctest
    print(doctest.testmod())
```

Composiciones alternativas

Estructura alternativa múltiple o anidada (if - elif - else)

- La sintaxis es:

```
if condición1:
    secuencia_de_instrucciones_si_condicion1_cierta
elif condición2:
    secuencia_de_instrucciones_si_condicion2_cierta
elif condición3:
    secuencia_de_instrucciones_si_condicion3_cierta
...
else:
    secuencia_de_instrucciones_si_condiciones_previas_falsas
```

Opcional

- Ejemplo, función signo:

$$signo(x) = \begin{cases} +1 \; si \; x > 0 \\ 0 \; si \; x = 0 \\ -1 \; si \; x < 0 \end{cases}$$

```
# Función signo(x), version elif
x = float(input("Entra x: "))
if x < 0:
  signo = -1
elif x == 0:
  signo = 0
else:
  signo = 1
print('El signo de', x, '=', signo)
```

Composiciones alternativas

Estructura alternativa múltiple o anidada (if - elif - else)

- El programa siguiente convierte una calificación numérica a formato cualitativo, de texto, considerando el formato español de calificación universitaria:

```
print('Programa que convierte una nota numérica a
cualitativa')
nota = float(input('Nota numérica: '))
if nota < 5:
  calif = 'Suspenso (SS)'
elif nota < 7:
  calif = 'Aprobado (AP)'
elif nota < 9:
  calif = 'Notable (NT)'
else:
  calif = 'Sobresaliente (SB)'
print('La nota', nota, 'equivale a', calif)
```

- Este programa puede ser robusto a notas erróneas, como -3 o 15 introduciendo el siguiente código, en lugar de “if nota < 5:”:

```
if (nota <0) or  (nota > 10):
  calif = 'Nota no válida'
elif nota < 5:
```

Composiciones alternativas

```
def calificacion(nota):
    '''Muestra la calificación
    a partir de la nota numérica

    Ejemplos:
    >>> calificacion(3)
    'Suspenso'
    >>> calificacion(5)
    'Aprobado'
    >>> calificacion(7.5)
    'Notable'
    >>> calificacion(9.5)
    'Sobresaliente'
    '''
    if nota < 5:
        calif = "Suspenso"
    else:
        if nota < 7:
            calif = "Aprobado"
        else:
            if nota < 9:
                calif = "Notable"
            else:
                calif = "Sobresaliente"
    return calif
if __name__ == "__main__":
    print(calificacion(4))
    print(calificacion(8))
    import doctest
    doctest.testmod()
```

```
def calificacion(nota):
    '''Muestra la calificación
    a partir de la nota numérica
    Ejemplos:
    >>> calificacion(3)
    'Suspenso'
    >>> calificacion(5)
    'Aprobado'
    >>> calificacion(7.5)
    'Notable'
    >>> calificacion(9.5)
    'Sobresaliente'
    '''
    if nota < 5:
        return "Suspenso"
    elif nota < 7:
        return "Aprobado"
    elif nota < 9:
        return "Notable"
    else:
        return "Sobresaliente"

if __name__ == "__main__":
    print(calificacion(4))
    print(calificacion(8))
    import doctest
    doctest.testmod()
```

Composiciones alternativas

Ejemplo: Solución a la ecuación de segundo grado

$a\,x^2 + b\,x + c = 0$

Se ha de calcular

$$x1, x2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Casos especiales:

a=0, b≠0: polinomio de 1er grado,

a=0, b=0, c≠0: No tiene solución real!

a=0, b=0, c=0, Tiene infinitas soluciones

Si el argumento : b^2-4ac < 0

Entonces NO tiene solución real

Las soluciones serían complejas (con parte imaginaria), que se pueden calcular en Python

Algoritmo en pseudocódigo

```
algorisme ecuacio
leer(a, b, c)
   si a = 0 entonces
        si b ≠ 0 entonces
              x1 := -c/b
              escrbir(x1)
        sino
            si c ≠ 0 entonces
                 escrbir ('No té solució!')
            sino
                  escrbir ('Té infinites soluc')
            fisi
        fisi
    sino
        arg := b**2-4*a*c
        si arg < 0 entonces
             escrbir ('No té solució real!')
        sino  x1:= (-b + arg**(1/2))/(2*a)
              x2: = (-b – arg**(1/2))/(2*a)
              escrbir (x1, x2)
        fisi
    fisi
fialgorisme
```