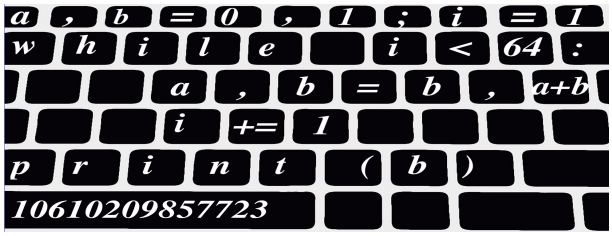

FUNDAMENTOS DE COMPUTACIÓN
Grado en ingeniería en tecnologías Industriales
Universidad de Cantabria

Programación Funcional
Noviembre del 2021

---

1. Escribir qué se muestra por pantalla cuando se ejecuta el siguiente programa:

```
                                    |
def f1(x,y):                        |
    print('f1:',x,y)                |
    return x+y                      |
                                    | SOLUCION--->
def f2(x,y):                        |
    print('f2:',x,y)                |
    return x*y                      |
                                    |
print(f1(f2(6,5),f1(2,4)))          |
```

2. Se considera la función f en Python y se ejecuta 4 veces con los argumentos indicados. Escribir en la tabla el valor que devuelve y su tipo.

funcion.py

```
1  def f(x):
2      if x % 2 != 0:
3          if x**2 <= 36:
4              return "Los Peños"
5          else:
6              return x // 3
7      else:
8           if x < 0 and abs(x) > 5:
9               return False
10          elif not x + 2 > 8:
11              return x / 2
12     return "La Mina Fontoria"
```

Llamada Funcion	Valor Devuelto	Tipo de Valor
f(2)		
f(13)		
f(-8)		
f(10)		

3. Escribe una función denominada `insertar` que tenga como argumentos dos cadenas `c1` y `c2` y que devuelva la lista de todas las cadenas, donde se ha insertado la cadena `c1` entre todos los caracteres de la cadena `c2`:

```
>>> insertar("abc","123")
['abc123', '1abc23', '12abc3', '123abc']
>>> insertar("a","")
['a']
>> insertar("","abc")
['abc', 'abc', 'abc', 'abc']
>>> insertar(" ","abc")
[' abc', 'a bc', 'ab c', 'abc ']
```

4. Determina, razonadamente, la salida de esta función mediante una expresión matemática dependiente del argumento de entrada x, que supondremos un número natural.

```
1  def g(x):
2      j = 0
3      while j // 2 < x:
4          j += 1
5          return j + 1
```

5. A continuación aparecen varias funciones Python que toman un argumento como entrada. Suponiendo que esa entrada es un número natural que denotamos mediante la variable n; indica, para cada función, si alguna de las expresiones matemáticas de la lista (y en caso afirmativo, cuál) representa su salida.

$$\text{mín}(7,n), \quad n, \quad \sum_{i=1}^{n} i, \quad n^2+n+1, \quad 2^n, \quad \prod_{i=1}^{n} i, \quad 7, \quad 2^{n+1}, \quad 0, \quad \sum_{i=1}^{n} i^2$$

```
def alpha(beta):
    suma = 0
    for x in range(1,beta+1):
        suma += 1
    return suma

def beta(suma):
    alpha = 1
    for x in range(0, suma+1):
        alpha *= 2
    return alpha

def gamma(x):
    suma = 0
    for beta in range(7, x):
        suma *= beta
        return beta
    return x

def delta(n):
    suma = n
    for x in range(1, n+1):
        suma *= 1
    return suma

def epsilon(alpha):
    prod = 0
    suma = 1
    while suma <= alpha + 1:
        prod += suma
        if prod > 6:
            return prod
```

```
def zeta(n):
    suma = 0
    i = 1
    while i <= n:
        suma += i
        n -= 1
    return suma

def eta(i):
    return n

def theta(suma):
    zeta = 1
    return zeta
    while zeta > 0:
        zeta -= 1
    return zeta

def iota(x):
    kappa = 1
    for i in range(0, x):
        suma = i
        suma = suma ** 2
        kappa += suma
    return kappa - 1

def kappa(n):
   alpha = n
   beta = 1
   gamma = beta
   while beta <= n:
       gamma += alpha + beta
       beta += 1
       alpha -= 1
   return gamma
```

6. Escribir qué se muestra por pantalla cuando se ejecuta el siguiente programa:

```
def campoo(a):                    | >>> a = [-1,2,-3,-4]       |              SOLUCION
    b = False                     |                            |
    for k in range(len(a)):       | >>> print(liguerucu(a))    | ----->
          b = b or cred(a ,k)     |                            |
    return b                      | >>> print(a)               | ----->
                                  |                            |
def cred(a,k):                    | >>> print(liguerucu(a))    | ----->
    a[k] = a[k]-2                 |                            |
    return a[k] < 0               | >>> print(a)               | ----->
                                  |                            |
```

7. Implementar una función de acuerdo con la documentación descrita abajo.

```
def zigzag(s1,s2):
""" Entrada: dos cadenas s1 y s2 ---> SALIDA:   una cadena
  Requisito previo: las dos cadenas tienen la misma longitud len(s1)==len(2).
   Devuelve una cadena conteniendo los caracteres alternativos de s1 y s2
   comenzando con s1[0], entonces s2[1], s1[2], s2[3],...
"""
```

```
>>> zigzag('abc', '123')
'a2c'
>>> zigzag('abcd', '1234')
'a2c4'
"""
```

8. ¿Cuál es la salida de $F([30, 40, 10, 20])$?

```
def F(x):
    n = len(x)
    for k in range(n-1):
        if x[k]>x[k+1]:
            t = x[k]            SOLUCION ----->
            x[k] = x[k+1]
            x[k+1] = t
        print(x)
```

9. Douglas R. Hofstadter, ganador del premio Pulitzer de 1980 con la obra *Gödel, Escher, Bach: an Eternal Golden Braid*, definió las siguientes funciones mutuamente recursivas:

hembra.py

```
1 def hembra(n):
2     if n == 0:
3         return n
4     f = hembra(n-1)
5     return n - macho(f)
```

macho.py

```
1 def macho(x):
2     if x == 0:
3         return 0
4     m = macho(x-1)
5     return x - hembra(m)
```

¿Cuál es la salida de `hembra(2)` y `macho(2)` ?

`hembra(2)`	`macho(2)`

10. Implementa una función `comun_caracter` que tenga como argumentos tres cadenas $c1, c2, c3$ no vacias y que devuelva el booleano `True` si las tres cadenas tienen un carácter en común, o `False` en caso contrario.

11. ¿Qué valor devuelve la siguiente función cuando se ejecuta con argumento $n = 4$? ¿Y cuando el argumento que toma es $n = 33$?

```
1 def f(n):
2     i=0
3     while 2**i<=n:
4         i+=1
5     return i
```

12. Escribe una función para dibujar triángulos como los de el ejemplo. Esta función ha de tomar dos argumentos: un número entero que actuará como parámetro del dibujo (el número de símbolos que forman los catetos del triángulo) y un valor *booleano* del que dependerá la colocación de la figura:

```
>>> triangulo(1,False)
$
>>> triangulo(2,True)
$ $
  $

>>> triangulo(3,False)
$ $ $
$ $
$
```

```
>>> triangulo(4,True)
$ $ $ $
  $ $ $
    $ $
      $
 >>> triangulo(5,False)
$ $ $ $ $
$ $ $ $
$ $ $
$ $
$
```