# Práctica 1 – Matemática Aplicada. Grupo 1$^o$B

## 1. Introducción

1. Sobre Python. ¿Por qué Python?
2. Instalación (Anaconda, Miniconda, WinPython,...)
3. Entornos de desarrollo (Thonny, Spyder, Jupyter, ...)

## 2. Operaciones Básicas

Escriba los siguientes ejemplos en el Shell (terminal, ventana de comandos...) y observe los resultados (los símbolos `>>>` no debe copiarlos, indican que las lineas se introducen en el Shell).

**Ejemplo 1.**

```
>>> 2+2
>>> 7-6
>>> 3*4
>>> 2**10
>>> abs(-10)
```

Observe que `2**10` calcula la potencia $2^{10}$ y que `abs` es la función valor absoluto.

**Ejercicio 2.** *Calcule* $0{,}93 \cdot (1{,}0225)^{1000}$.
*(Sol.* $4{,}28350844971 \cdot 10^9 = 4283508449{,}71$*)*

Observe el resultado del siguiente ejemplo. Aunque el resultado debería ser 0, al hacer los cálculos internos en punto flotante se obtiene un resultado aproximado.

**Ejemplo 3.**

```
>>> (0.1 + 0.2 + 0.7) - (0.2 + 0.7 + 0.1)
```

Observe también que se usa en el resultado notación científica, donde se usa la letra `e` para indicar una potencia de 10.

## 3. El módulo `math`

Para usar muchas funciones matemáticas y constantes como $\pi$ o $e$, importamos el módulo `math` mediante la instrucción `import math`.

En los siguientes ejemplos calculamos $\sin(\pi)$, $\ln(e)$ y $\sqrt{49}$.

**Ejemplo 4.**

```
>>> math.sin(math.pi)
>>> math.log(math.e)
>>> math.sqrt(49)
```

Observe que la función `math.log` es el logaritmo neperiano.

Otra forma de importar el módulo consiste en usar la instrucción `form math import *`, con la cual se pueden usar las funciones y comandos del módulo sin necesidad del `math.` previo, aunque puede provocar confusión con otras instrucciones o funciones del mismo nombre.

**Ejercicio 5.** *Halle el valor de la expresión* $\cos(\pi/2) \cdot e^2$.

*(Sol. un valor próximo a cero, en teoría exactamente cero).*

## 4. Un primer programa. Variables. La función `print`

Vamos a escribir ahora un programa como el siguiente. Primero creamos un programa nuevo, y luego tecleamos lo siguiente.

**Ejemplo 6.**

```
"""
Un programa muy sencillo.
"""

a=2
b=3
print("El resultado es ")
a*b
print(a+b)
```

Después lo grabamos con un nombre (por ejemplo `prog0.py` ).

La línea que se encuentra entre las tres comillas es un comentario. También se puede usar el carácter `#` para incluir un comentario, ya que el intérprete ignora ese carácter y todo lo que siga en esa línea. Para hacer los programas más legibles se recomienda incluir comentarios y usar nombres de variables significativos.

El programa define dos variables `a` y `b` y luego muestra en pantalla un mensaje y la suma de las dos variables. Se observa que la instrucción `a*b` no se muestra en pantalla, porque no lleva un `print`.

## 5. Bucles

El siguiente ejemplo muestra todos los números naturales del 0 al 9.

**Ejemplo 7.**

```
for i in range(10):
    print(i)
print("Hecho.")
```

Al ejecutar, observamos que aunque se ponga `range(10)` la variable `i` (llamada contador) toma los valores 0,...,9 dentro del bucle. El *cuerpo* del bucle, la parte que se repite, es en este caso una única instrucción, el `print(i)`. El cuerpo del bucle se distingue porque está desplazado a la derecha cuatro espacios, se dice que está indentado o sangrado. Al final de la linea del `for` hay dos puntos, antes de que comience el bucle.

Si queremos mostrar en pantalla los números naturales del 3 al 14, modificamos el programa de la siguiente manera.

**Ejemplo 8.**

```
for i in range(3,15):
    print(i)
```

El siguiente programa muestra los números impares entre el 7 y el 21, ambos incluidos.

**Ejemplo 9.**

```
for i in range(7,22,2):
    print(i)
```

Observamos que en este caso la función `range` tiene 3 parámetros, el `inicio` (en este caso 7), el `fin` (en este caso 22), y el `paso` que indica la diferencia entre un valor del contador y el siguiente, en este caso los valores se cuentan *de dos en dos*.

**Ejercicio 10.** *Escribir un programa que muestre los números del 10 al 15. Modifique el programa para que muestre los cuadrados de esos números.*

## 6. Acumuladores

Un acumulador es una variable que incrementa su valor de manera variable dentro de un bucle. Aquí los usaremos para poner más ejemplos de bucles.

En el siguiente ejemplo, el contador `i` va desde 0 hasta $N-1$, y la variable `acumulador` se ve incrementada en el valor `i` en cada iteración del bucle.

**Ejemplo 11.**

```
"""
Suma de los N primeros numeros naturales.
"""

N= 10
acumulador = 0
for i in range(N):
    acumulador = acumulador + i

print(acumulador)
```

Como resultado se obtiene $\sum_{i=0}^{N-1} i$.

Si queremos calcular el producto de varios números, debemos inicializar el acumulador con el valor 1.

**Ejemplo 12.**

```
"""
Multiplicacion de los N primeros numeros naturales.
"""

N= 10
acumulador = 1
for i in range(1,N):
    acumulador = acumulador * i

print(acumulador)
```

Observe también que no podemos comenzar el contador por 0.

**Ejercicio 13.** *Calcule el producto de los números pares entre el 4 y el 16, ambos incluidos.*

*(Sol. 5160960).*

## 7. Matrices con el módulo `numpy`

Para manejar matrices en Python, usaremos el objeto `array` del módulo `numpy`. Para estos ejemplos sencillos cargaremos el módulo usando la instrucción `from numpy import *` . (En otros casos será preferible usar `import numpy` o `from numpy import array,dot`,...).

El siguiente programa define la matriz $A$ como la identidad de orden 3, y las matrices $B$ y $C$. Luego calcula la matriz producto $C \cdot B$ y el determinante de la matriz $A$.

**Ejemplo 14.**

```
from numpy import *

A=identity(3)
B=array([[1,2],[3,4],[5,6]])
C=array([[7,8,9],[10,11,12]])

print(dot(C,B))
print(linalg.det(A) )
```

El siguiente programa calcula la inversa de una matriz.

**Ejemplo 15.**

```
B=array([[1,2,3],[2,3,4],[1,2,1]])
print( linalg.inv(B) )
```

**Ejercicio 16.** *Considere la matriz* $A = \begin{pmatrix} 1 & 3 & 2 \\ 2 & 3 & 4 \\ 0 & 1 & 1 \end{pmatrix}$. *Halle* $A^{-1} \cdot A$.

*(Sol. Identidad de orden 3)*

## 8. Álgebra lineal con el módulo `sympy`

Podemos hallar la forma escalonada reducida de una matriz mediante el módulo `sympy`.

**Ejemplo 17.**

```
from sympy import *

M = Matrix( [[1,2,3,4],[1,1,2,1],[2,3,5,5]] )
M_escalonada = M.rref()
print(M_escalonada)
```

También usamos este módulo para resolver sistemas de ecuaciones de manera simbólica.

Por ejemplo, para resolver el sistema

$$\left\{ \begin{array}{rcrcrcr} x & + & y & - & z & = & 2 \\ x & + & 2y & + & z & = & 3 \end{array} \right.$$

planteamos la matriz ampliada

$$\left( \begin{array}{rrr|r} 1 & 1 & -1 & 2 \\ 1 & 2 & 1 & 3 \end{array} \right).$$

Este sistema lineal de ecuaciones se puede resolver mediante el siguiente programa

**Ejemplo 18.**

```
from sympy import symbols, Matrix, solve_linear_system

x, y, z = symbols('x, y, z')
A = Matrix( [[1, 1, -1, 2], [1, 2, 1, 3]] )
print( solve_linear_system(A, x, y, z) )
```



## 9. Ejercicios

1. Halle una aproximación numérica de la expresión

$$\frac{e^{\pi}-2}{\operatorname{sen}(\pi/2)+1}.$$

(Sol. 10.57034631638963450286454318397427369 0...)

2. Escriba un programa que, dados dos catetos de un triángulo, calcule la hipotenusa y el perímetro del triángulo.

3. Considere las matrices

$$A=\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 4 \end{pmatrix} \text{ y } B=\begin{pmatrix} 2 & 3 & 5 \\ -1 & 2 & 1 \end{pmatrix}.$$

Halle $A+B$ y $2B-A$.

(Sol. $\begin{pmatrix} 3 & 3 & 4 \\ -1 & 4 & 5 \end{pmatrix}$, $\begin{pmatrix} 3 & 6 & 11 \\ -2 & 2 & -2 \end{pmatrix}$.)

4. Halle el determinante de la matriz

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 4 \\ 2 & 7 & 3 \end{pmatrix}.$$

(Sol. $-18$ )

5. Halle la inversa de la matriz

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & 4 \\ -2 & 5 & 0 \end{pmatrix}.$$

(Sol. $\dfrac{1}{42}\begin{pmatrix} 20 & -15 & -11 \\ 8 & -6 & 4 \\ 2 & 9 & 1 \end{pmatrix} = \begin{pmatrix} 0{,}476190 & -0{,}357143 & -0{,}261905 \\ 0{,}190476 & -0{,}142857 & 0{,}09524 \\ 0{,}047619 & 0{,}214286 & 0{,}023810 \end{pmatrix}$.)

6. Obtenga la forma escalonada reducida y, contando pivotes, halle el rango de la matriz

$$\begin{pmatrix} 1 & -2 & 3 & 0 \\ 2 & 3 & 0 & -1 \\ 4 & -1 & 6 & -1 \end{pmatrix}.$$

(Sol. Rango 2)

7. Discuta y resuelva el sistema

$$\begin{cases} 3x & -y & +4z & = 5, \\ 2x & -2y & +z & = 1, \\ -4x & +8y & +3z & = 5. \end{cases}$$

(Sol. $x = x$ (parámetro), $y = \frac{1}{7} + \frac{5}{7}x$, $z = \frac{9}{7} - \frac{4}{7}x$. Compatible indeterminado. )

## 10. La función input

Hemos visto cómo definir variables dentro de un programa para ser usadas posteriormente. Con la función `input` podemos introducir datos desde el teclado. Como argumento de la función podemos incluir un texto que aparece al pedir el dato. La función `input` devuelve al programa una cadena de caracteres; si queremos convertir esa cadena a un valor numérico, podemos usar las funciones `int` para convertir la cadena a un número entero o `float` para obtener un número en punto flotante (con decimales).

**Ejemplo 19.**

```
a=input("Introduzca el primer valor: ")
b=input("Introduzca el segundo valor: ")
```

```
print("La suma de los dos valores es...")
print(a+b)
```

Observe que hemos puesto un espacio al final de los argumentos de los `input`, así separamos estas cadenas de los `input` de los valores que introduce el usuario.

**Ejercicio 20.** *Escriba un programa que pida al usuario el radio de un círculo y calcule y muestre en pantalla el área de ese círculo y la longitud de la circunferencia. El programa debe funcionar cuando el valor del radio es un número decimal.*

*(Sol. Por ejemplo, si el radio es 2.5, el programa debe devolver un área de 19.634954084936208 y una longitud de 15.707963267948966 ).*

## 11. Condicional. El comando if

El comando `if` permite ejecutar un bloque de instrucciones *si* cierta condición es cierta.

En el siguiente ejemplo se pide un número, que se guarda en la variable `a`, y se imprime el mensaje `El valor es menor que 7` si se cumple la condición `a`$< 7$.

**Ejemplo 21.**

```
a=int(input("Introduzca un valor: "))
if a<7:
    print("El valor es menor que 7")
print("Fin del programa")
```

Debe observarse que, tras la condición, se escriben dos puntos para indicar que comienza el bloque de instrucciones que se ejecutan si la condición es cierta. Este bloque de instrucciones debe estar indentado con cuatro espacios.

**Ejercicio 22.** *Escriba un programa que pida al usuarios tres números* $a, b$ *y* $c$ *y que muestre el mensaje 'Discriminante negativo' si* $b^2 - 4ac < 0$.

En el siguiente ejemplo se incluyen dos bloques de instrucciones, uno si la condición se cumple, y otro si la condición no se cumple.

**Ejemplo 23.**

```
a=int(input("Introduzca un valor: "))
if a<7:
    print("El valor es menor que 7")
else:
    print("El valor es mayor o igual que 7")
print("Fin del programa")
```

El bloque que está indentado tras el `else:` se ejecuta cuando la condición no se cumple. La estructura `if ...: ... else ...` se puede leer como *si... entonces... en caso contrario....*

**Ejercicio 24.** *Escriba un programa que pida dos números enteros, un numerador y un denominador. Si el denominador es igual a cero, el programa debe indicarlo al usuario con un mensaje. Si no es cero, debe imprimir el cociente entre el numerador y el denominador.*

## 12. Ejercicios

1. Escriba un programa que solicite los valores de $a$, $b$ y $c$, las longitudes de los lados de un triángulo. Dados estos valores, calcule el valor del semiperímetro $s = \dfrac{a+b+c}{2}$ y del área

$$\text{area} = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$$

   i) Calcule el área de un triángulo cuyos lados miden 120, 150 y 200 metros.
   (Sol. Área$= 25 \cdot \sqrt{128639} = 8966.569856974293$ )

   ii) ¿Puede calcular el área de un triángulo de lados $a = 80$, $b = 70$ y $c = 160$?

   iii) Modifique el programa de forma que se calcule el área sólamente si la condición

```
b+c>a and a+b>c and a+c>b
```

   se cumple.

2. Escriba un programa que, tras pedir el valor $n$, calcule una aproximación del número $e$ mediante la siguiente suma

$$\sum_{i=0}^{n} \frac{1}{i!}$$

donde $i! = i \cdot (i-1) \cdot (i-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$.

(Indicación: recuerde los acumuladores de suma y de producto).