

## Computación, Práctica 9.

### Programación en Python: estructuras de control y funciones.

Los ejercicios en Python se pueden realizar en las computadoras del aula informatizada de la misma forma que venimos haciéndolo con los ejercicios en FORTRAN, es decir, escribiendo el código fuente en un archivo, en estos casos con extensión **.py**, y luego ejecutándolo por línea de comando en la terminal escribiendo:

```
> python3 archivo.py
```

También se puede utilizar el entorno *Spyder* que se encuentra instalado en las computadoras del aula informatizada. Para acceder al mismos, en la terminal escribimos:

```
> spyder &
```

Este entorno posee su propio editor y su consola de ejecución. Una de las ventajas de *Spyder* es que nos permite ejecutar porciones del programa independientemente del resto.

#### Funciones

**Ejercicio 1.** Construir una función que reciba 3 cantidades y calcule su promedio. Evaluar esta función con tres valores a su elección.

**Ejercicio 2.** Utilizando la función definida en el ejercicio anterior, calcular el promedio pero con los valores ingresados por el usuario.

**Ejercicio 3.** Escribir una función que devuelva el resultado de calcular  $f(x) = x^2 + 10x + 1$  donde  $x$  se lee del teclado. Probar su funcionamiento llamando a la función.

**Ejercicio 4.** Escribir una función que devuelva el resultado de  $f(x) = \sin(x) + \cos(x)$  donde  $x$  está en grados. Probar su funcionamiento con varios valores, realizando un programa que utilice esta función.

**Nota:** Recordar que las funciones trigonométricas no se encuentran establecidas por "default" en Python. Se debe indicar que se requiere de su uso, importando la librería adecuada.

**Ejercicio 5.** Escribir una función que resuelva una ecuación de segundo grado de la forma  $ax^2 + bx + c$  sin utilizar la orden *IF()*. Probar la función con las con las siguientes ternas:

$$a = 1, b = -3, c = 2$$

$$a = 1, b = -2, c = 5$$

$$a = 1, b = -2, c = 1$$

**Ayuda:** no use el módulo *MATH* sino el *CMATH* que utiliza álgebra de números complejos.

#### Estructuras de Control

**Ejercicio 6.** Dada una lista de números, hacer un programa que separe e imprima aparte los divisibles por 3.

Usar como ejemplo la lista = [1, 23, 4, 5, 629, 45, 23, 875, 324, 534, 544, 745, 3, 6, 27, 54, 266].

**Ejercicio 7.** Escribir un programa que calcule la diferencia entre 22 y un número cualquiera. Si el número es más grande que 22 devuelve el doble de la diferencia.

Debe utilizar una función para resolverlo.

**Ejercicio 8.** Escribir una función que tenga como entrada un string, y si el string empieza con "La" lo devuelve sin modificar, si no le agrega "La" al comienzo.

**Ejercicio 9.** Escribir una función que descubra si el número ingresado es cercano con una diferencia menor a 100 al número 1000 o al 2000. Debe retornar como resultado verdadero o falso.

**Ejercicio 10.** Escribir un programa en Python que a partir del diccionario creado para el ejercicio 10 de la práctica 8, (el que contiene el nombre y algunas características físicas de varias personas), escriba por un lado el nombre de las que tienen ojos verdes, y por otro quienes tienen cabello negro.

**Ejercicio 11.** Basado en lo realizado en el ejercicio 5 de esta práctica, reescribir la función que resuelve la cuadrática usando la orden *IF()* (en forma similar a lo realizado en el ejercicio 2 de la práctica 4).

Para probar su funcionamiento utilice las tres ternas del ejercicio 5 pero que las mismas sean ingresadas por el usuario.

**Nota:** Usar el módulo *CMATH* sólo cuando corresponda.

**Ejercicio 12.** Construir un diccionario que contenga los nombres de varios alumnos y las tres notas de sus parciales. Escribir un programa para que un profesor ingrese el nombre del alumno, mediante una

función determine su promedio y si aprobó por promoción, aprobó pero debe rendir final o desaprobó.

**Nota:** Considerar que un alumno aprueba por promoción si su promedio es 7 o mayor, y estará desaprobado si es menor a 4. En los demás casos estará aprobada su cursada pero deberá rendir final.

**Ejercicio 13.** En el archivo *Arbolado-Publico-Urbano.dat*, se encuentran los datos del arbolado público urbano de ciertas calles de la Ciudad de La Plata. En la primera columna se listan dichas calles, en la segunda y tercera columnas, su longitud en *km* y su ancho en *m* respectivamente, y en la cuarta columna el total de árboles presentes en cada una. Escribir un programa en Python que lea los datos del archivo y almacene cada columna en una lista. A partir de dichos datos, construir una función que calcule cuántos árboles hay por cada  $10000\text{ m}^2$  en cada calle. Imprimir los resultados con formato de

tabla. ¿Qué modificaciones tendrías que hacer en el programa de lectura para almacenar los datos en un diccionario, cuyas claves sean el encabezado que distingue cada columna y como valor asociado una lista con el contenido de cada una?

**Nota:** Prestar atención al tipo de dato que se desea almacenar en cada lista. Usar el método `next` en la lectura del archivo para saltar el encabezado.

**Ejercicio 14.** Reescribir en Python el ejercicio 10 de la práctica 6, implementando el Método de Integración de Simpson como una función.

**Ejercicio 15.** Reescribir en Python el ejercicio 4 de la práctica 7, implementando el método de la Regula Falsi como una función, donde además de devolver la raíz, devuelva el número de iteraciones realizadas para encontrarla.