# Tarea Automatizada #1

# Manual de Usuario


## Jhon Sebastian Rojas Rodriguez


Para hacer uso del Script de Python se necesita tener instalado el lenguaje Pyhton en el equipo, basta con abrir una terminal de Python en la ubicación del código fuente e importar el script con el comando ***import primos as pr***

De esta manera se tiene acceso tanto a la función del test de Miller-Rabin con el comando ***pr.esPrimo(n, s)*** , la cual recibe como parámetros el número ***n*** a ser testeado y opcional el número ***s*** de veces que se escoge un número a aleatorio para realizar el test(por defecto 50):

```
>>> import primos as pr
>>> pr.esPrimo(21)
(False, 1)
>>> pr.esPrimo(1973)
(True, 0.9999999999999933)
>>> pr.esPrimo(7681, 10)
(True, 0.9913389745642435)
>>>
```

Como se observa en la imagen la función retorna una dupla con un booleano indicando si el número es primo y la probabilidad de que esto sea cierto.

Adicionalmente se tiene la funcionalidad de generar números primos de ***d*** dígitos mediante la función ***pr.generarPrimo_digitos(d)***:

```
>>> import primos as pr
>>> pr.generarPrimo_digitos(100)
(829862211551208719849347229474368218380174111767070215177147959955269213686853632032533029424131462 3L, 0.9999999999997957)
>>>
```

La función retorna una dupla que se compone del número primo pedido y la probabilidad de que este numero sea un número primo.

# Tarea Automatizada #1

# Manual Técnico

## Jhon Sebastian Rojas Rodriguez

## Librerias y Herramientas de desarrollo utilizadas:

Para el desarrollo de la aplicación se utilizo un cuaderno de Jupyter en el ambiente colaborativo de Google Colab. Se hace uso de las librerías *math* y *random* de la instalación estándar de Python.

## Estructura de la aplicación:

La aplicación consiste en 4 funciones:

1. esTestigo(a, n): Función que implementa el test de miller-rabin para un testigo a dado y un número n en cuestión. Es una implementación directa del pseudocódigo presentado en el PDF de diseño de la aplicación. Cabe recalcar el uso de la función pow de la librería math para realizar el cálculo de las potencias, esto debido a que con números muy grandes el operador % por defecto es muy lento. Retorna verdadero si a es un testigo de la primalidad de n y falso en caso contrario.

2. test_miller_rabin(n, s): Función que realiza el test al número n en cuestión s veces, seleccionando un testigo a de manera aleatoria en cada iteración. Retorna un booleano indicando si el número es primo o no.

3. esPrimo(n, s = 50): Función que determina si el numero n es primo o no y la probabilidad de esto. Hace el cálculo de la probabilidad usando la expresión:

   $$\frac{1}{1+2^{-s}\ln(n-1)}$$

   Retorna una dupla con un booleano y la probabilidad, si el numero es compuesto la probabilidad que retorna es 1.

4. generarPrimo_digitos(d): Función que genera números primos con un número de digitos d determinado. Genera números aleatorios de esa longitud con los que llama la función anterior hasta que encuentra un número primo y lo retorna en una dupla junto con la probabilidad de su primalidad.