Problemas #1 - *Intro a Python*

Para los problemas a continuación debe cumplir las especificaciones dadas y hacer entraga de cada problema en el formato que se define al final de cada uno con el título de "**Entrega**". Debe hacer entrega de la solución de estos problemas mediante la plataforma del curso, en la tarea correspondiente, con todos los archivos guardados dentro de una carpeta, y dicha carpeta comprimida con el formato siguiente: `nombre_apellido.[rar|tar.gz|zip]`. Se le pide encarecidamente que cumpla este formato y que los archivos estén dentro de una carpeta con el fin de facilitar la revisión.

Problema 1. *Github*

Para este primer problema se pide que ingrese a `http://github.com/` y cree un repositorio llamado `lytp`. Una vez creado el repositorio, debe crear un repositorio local, i.e. en su computador, y agregar el repositorio de Github como remoto. Luego debe agregar el script hecho en el Laboratorio #2 donde se ingresaba el nombre por teclado y como salida mostraba "Hola, *nombre_ingresado*", poniéndole como nombre `hola.py`. Finalmente debe hacer `push` a Github para subir el script.

Entrega: archivo `repo.txt` que contenga **sólo una línea** con el enlace a su repositorio de Github, donde se muestre el script `hola.py`.

Nota: verifique que el archivo se ha subido correctamente y que puede verlo desde Github, en el explorador de Internet.

.....

Problema 2. *Ambos extremos*

En Python las funciones se declaran de la siguiente forma:

```
def es_par(n):
    if n %2 == 0:
        return True
    else:
        return False
```

Al final de la línea de encabezado se pone ':', igual que con las estructuras de control, y los argumentos que acepta la función no llevan tipo.

Este problema consiste en que construya una función `ambos_extremos` donde, dado un string s, retorne un string hecho a partir de los primeros 2 y los últimos 2 caracteres del string original; si el largo del string es menor a 2, retorne el string vacío (‘‘’’).

Entrega: script `ambos_extremos.py`.

Nota: para probar el código, abra la terminal y haga `cd` *a la carpeta donde se encuentra el script y ejecute la instrucción* `from ambos_extremos import *` *para cargar el código en la shell. Una vez hecho esto, puede ejecutar la función de la siguiente forma:*

```
>>> ambos_extremos("dota")
'dota'
>>> ambos_extremos("manzana")
'mana'
>>> ambos_extremos("re")
'rere'
>>> ambos_extremos("o")
''
```

.....

Problema 3. *Fibonacci... sí, Fibonacci*

Construya una función recursiva `fibonacci(n)` que imprima el n-ésimo término de la serie de Fibonacci.

Entrega: script `fibonacci.py`.

.....

Problema 4. *Permutaciones*

En matemática, una permutación hace referencia a la reorganización de objetos o elementos. Informalmente hablando, una permutación de un conjunto de datos es una distribución determinada de esos datos en un orden en particular. Así, teniendo el conjunto {1, 2, 3}, se tienen seis permutaciones: (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), y (3,2,1). El número de permutaciones de n objetos distintos se calcula con la denominada función factorial, escrita $n!$. Programe una función `factorial(n)` que determine el factorial de un número n.

Entrega: script `factorial.py`.

Nota: la función debe ser iterativa, i.e. debe ocupar un ciclo `while`.

.....

Problema 5. *Coeficiente del demonio*

Los coeficientes binomiales son números que aparecen como coeficientes en el teorema del binomio. Esta familia de números también aparece en las combinatorias, y nos permite determinar cuántas formas hay de organizar k elementos de un conjunto de n elementos.

El coeficiente binomial se determina así:

$$\binom{n}{k} = \frac{n!}{(n-k)!\,k!} \tag{1}$$

Programe una función `combinatoria(n, k)` que retorne el resultado de la combinatoria. Asuma $0 < k \leq n$.

Entrega: script `combinatoria.py`.

.....

Problema 6. *Raíces*

Para este problema se pide que realice una función `raices(a, b, c)` que tome los tres coeficientes de una ecuación cuadrática e imprima en pantalla las raíces e indique si éstas son reales o complejas.

Entrega: script `raices.py`.

Nota: la función `type()` *retorna el tipo de un objeto.*

.....