S2-Pilas-y-Colas

Recibir n números, meterlos a una pila, sacarlos e imprimirlos - Básico

Recibir n números, meterlos a una pila, sacarlos e imprimirlos - Básico

Ficheros requeridos: pila.py (Descargar)
Tipo de trabajo: Individual

Vamos a comprobar que si entiendas el funcionamiento básico de una pila (en inglés, *stack*). Te daremos un número N seguido de N números. Tu objetivo es leer los números, y guardarlos en la pila para, posteriormente, sacarlos e imprimirlos.

Ejemplo:

Entrada

5

1 2 3 4 5

Salida

5 4 3 2 1

Nota: Recuerda que Python ya trae sus propia estructura para pilas: *deque (*cuyo nombres es la abreviación de Double-Ended QUEue, en español, bicola).

S2 - Invertir una lista simplemente enlazada utilizando una pila como apoyo - Sencillo

S2 - Invertir una lista simplemente enlazada utilizando una pila como apoyo - Sencillo

Ficheros requeridos: invertir.py (Descargar)
Tipo de trabajo: Individual

Nombre archivo: invertir.py

Dada una lista simplemente enlazada, invertirla (en inglés, *reverse it*) haciendo uso de una pila (en inglés, *stack*). En Python, las pilas se hacen con la librería *deque*.

El error "None type has no atribute next" o "None type has no atribute val" quiere de decir que la nada no tiene nada. Como la nada no tiene nada, entonces la nada no tiene ni siguiente (en inglés, *next*) ni valor (en inglés, *VALue*.

El error "int has no atribute next" o "int has no atribute val" sucede cuando digo que el siguiente de un nodo es un entero. Recordemos que el siguiente de un nodo NO es un entero, es un nodo. El siguiente de un nodo, es un nodo.

Entrada:

La cabeza de una lista simplemente enlazada como parámetro de una función.

Salida:

La cabeza de la nueva lista simplemente enlazada con retorno de una función.

No hay que imprimir nada.
No hay que retornar una pila.

Ejemplo 1:

Input:
1 -> 2 -> 3 -> 4 -> 5
Output:
5 -> 4 -> 3 -> 2 -> 1

S2 - Mensajes Encriptados - Sencillo

S2 - Mensajes Encriptados - Sencillo

Ficheros requeridos: mensaje.py (Descargar)
Tipo de trabajo: Individual

Una manera de encriptar mensajes, es colocar paréntesis de manera arbitraria y, todo lo que está dentro de un paréntesis, ponerlo al revés; por ejemplo, “Olimpiada de Informática” se puede encriptar como “Olimpia(ad) de I(rofn)mática”. Los paréntesis también se pueden anidar; es decir, otra forma de encriptar el mismo mensaje podría ser “Olimpia(ám(nfor)I ed (da))tica”.

Para validar que un mensaje esté bien encriptado, se debe verificar que los paréntesis están de forma correcta; es decir, que todo paréntesis que abre vuelva a cerrar. Para ayudar a verificar que el mensaje está bien encriptado, debes crear una función que reciba como parámetro el mensaje s (la longitud de este está entre 0 <= s.length <= 3*10^4), y debes retornar (NO imprimir) el número de paréntesis que encriptan correctamente el mensaje.

Para realizar este ejercicio con pilas, se debe ingresar a la pila cada paréntesis que abre. Adicionalmente, cuando se encuentra un paréntesis que cierra, se debe sacar de pila un paréntesis de abre. Para darse una idea, si la pila queda vacía es porque todo paréntesis que abrió, se cerró.

Ejemplo:

Entrada

)()())

Salida

4

Ya que 4 es la longitud de los paréntesis que encriptan correctamente: “()()”

Nota: Recuerda que,en Python, la lista enlazada, pila y cola se implementan con *deque*.

S2 - Notación Polaca Inversa - Intermedio

S2 - Notación Polaca Inversa - Intermedio

Ficheros requeridos: polaca.py (Descargar)
Tipo de trabajo: Individual

La notación polaca inversa (RPN) es una alternativa a la construcción de expresiones por paréntesis, en la que se utiliza un orden diferente (operando1 operando2 operador) para escribir los operandos y las operaciones. A diferencia de la notación tradicional (operando1 operador operando2), donde la suma de A más B se escribe:

A + B

En la notación polaca inversa, la misma operación se escribe:

A B +

La ventaja de la notación RPN es que no requiere paréntesis para garantizar el orden adecuado de las operaciones, pues estas se van realizando a medida que aparecen. Por ejemplo, la expresión clásica (2+3)*4, que es imposible escribir sin paréntesis por la jerarquía de la multiplicación, en notación RPN se escribe:

2 3 + 4 *

Mientras que 2 + 10 / 5 se escribiría:

2 10 5 / +

Entrada

Una línea de texto que contiene una expresión en notación polaca inversa, formada por dígitos del 1 al 9, los operadores aritméticos +, - , * y / y espacios que separan cada elemento.

Salida
El resultado de calcular adecuadamente la expresión polaca inversa.

Ejemplo1:

Entrada

2 3 +

Salida

5

Ejemplo2:

Entrada

12 6 - 3 1 + 2 * /

Salida

```
0.75
```

S2 - Editor de texto - Intermedio

S2 - Editor de texto - Intermedio

Ficheros requeridos: editor.py (Descargar)
Tipo de trabajo: Individual

Implementar un editor de texto. El editor contiene inicialmente una cadena vacía, S. Realiza Q operaciones de los siguientes 4 tipos:
1. Añadir(W) - Añade una cadena al final de S.
2. borrar(k) - Borra los últimos k caracteres de S.
3. imprimir(k) - Imprime el carácter k de S.
4. deshacer - Deshace la última operación (no deshecha previamente) del tipo 1 o 2, volviendo al estado en que se encontraba antes de esa operación.

Ejemplo:

```
indice   S       ops[indice]  Explicacion
-----   ------   ----------  -----------
0       abcde    1 fg        Añade fg
1       abcdefg  3 6         Imprime la sexta letra - f
2       abcdefg  2 5         Elimina las ultimas 5 letras
3       ab       4           Deshace la ultima operacion, index 2
4       abcdefg  3 7         Imprime el septimo caracter - g
5       abcdefg  4           Deshace la ultima operacion, index 0
6       abcde    3 4         Imprime el cuato caracter - d

El resultado deberia imprimirse como:
f
g
d
```

Formato de entrada
La primera línea contiene un número entero,que denota el número de operaciones.
Cada línea i de las siguientes Q (donde 0 <= i < Q) define una operación a realizar. Cada operación comienza con un único número
Si la operación requiere un argumento, t va seguido de su argumento separado por espacios. Por ejemplo, si t = 1 y W = "abcd", l

Restricciones
La suma de las longitudes de todos W en la entrada <= 1e6.
La suma de k sobre todas las operaciones de borrado <= 2*1e6.
Todos los caracteres de entrada son letras inglesas minúsculas.
Se garantiza que la secuencia de operaciones dada como entrada es posible de realizar.

Formato de salida
Cada operación de tipo 3 debe imprimir el carácter en una nueva línea.

Ejemplo entrada

```
8
1 abc
3 3
2 3
1 xy
3 2
4
4
3 1
```

Ejemplo salida

```
c
y
a
```

Recibir n números, meterlos a una cola, sacarlos e imprimirlos - Básico

Recibir n números, meterlos a una cola, sacarlos e imprimirlos - Básico

Ficheros requeridos: cola.py (Descargar)
Tipo de trabajo: Individual

Vamos a comprobar que si entiendas el funcionamiento básico de una cola. Te daremos un número N seguido de N números. Tu objetivo es leerlos, y guardarlos en la cola para, posteriormente, sacarlos e imprimirlos.

Ejemplo:

Entrada

5

1 2 3 4 5

Salida

1 2 3 4 5

Nota: Recuerde que Python ya trae su propia cola llamada deque (abreviación en inglés de Double-Ended QUEue, en español, bicola).

S2 - Generar Binarios - Sencillo

S2 - Generar Binarios - Sencillo

Ficheros requeridos: genBin.py (Descargar)
Tipo de trabajo: Individual

Nombre archivo: genBin.py

En el curso de Organización de Computadores (obligatoria para Sistemas, electiva para matemáticas), profundizaremos sobre el uso de números binarios en la definición de las operaciones que realiza la unidad central de procesamiento (CPU). Mientras tanto, resolvamos el siguiente problema: Dado un entero N, genera los binarios entre 1 y N.

Entrada:

Se recibe, por pantalla (en inglés, por *input*), un entero N

Salida:

Se imprime, en la pantalla (en inglés, *print*), la lista con los binarios, entre 1 y N, en forma de cadena de caracteres.

Ejemplo 1:

Input:
7
Output:
["1", "10", "11", "100", "101", "110", "111"]

Ficheros requeridos

genBin.py

```
from collections import deque
def binarios(n):
    pass

n = input()
print(binarios(n))
```

S2 - Los soldados reclutados por Gilgamesh - Intermedio

S2 - Los soldados reclutados por Gilgamesh - Intermedio

Ficheros requeridos: soldados.py (Descargar)
Tipo de trabajo: Individual

En un intento por derrotar al toro celeste de la diosa Ishtar, Gilgamesh comenzó a reclutar soldados para enviarlos a combatir contra el toro. Mientras los hombres de Uruk son reclutados uno por uno, Gilgamesh decide en qué momento mandar a un soldado pelear contra el toro. Cada soldado será enviado en el orden en que fue reclutado. Ayude a Gilgamesh a determinar quién es el siguiente soldado que irá peleando contra el poderoso toro.

Entrada

La entrada estará conformada por un número arbitrario de líneas. Cada línea corresponderá a una de dos acciones: reclutar un hombre, que será denotado con la palabra "LLAMA" y el nombre del soldado, o mandar pelear a un hombre, que será denotado con la palabra "MANDA". Puede suponer que cada nombre estará conformado por a lo mucho 10 letras mayúsculas.

Salida

Por cada acción imprima el nombre del soldado involucrado. En caso de que no haya soldados para enviar deberá escribir la palabra "IMPOSIBLE".

Ejemplo1

Entrada
LLAMA NEBO
LLAMA NABU
MANDA
LLAMA MARDUK
MANDA
MANDA
MANDA
LLAMA MARDUC

Salida
NEBO
NABU
NEBO
MARDUK
NABU
MARDUK
IMPOSIBLE
MARDUC