Argentina Programa 4.0

Universidad Nacional de San Luis

Desarrollador Python

Práctico Nro. 6.3

Lenguaje de Programación Python: Soporte Orientado a Objetos

Autor:

Dr. Mario Marcelo Berón

Argentina Programa 4.0
Universidad Nacional de San Luis

Práctico Nro. 6.3: *Tipos de Dato Integrados*

Ejercicio 1: Defina la clase Punto la cual permite realizar las siguientes operaciones:

1. Crear un punto en el origen de coordenadas o en coordenadas indicadas por el usuario.
2. Sumar puntos utilizando el operador +.
3. Restar puntos utilizando el operador -.
4. Recuperar la coordenada x de un punto.
5. Convertir un punto en una tupla.
6. Comparar dos puntos utilizando el operador ==.
7. Convertir un punto a tupla, lista y string.
8. Proveer una representación textual de un punto.

Ejercicio 2: Defina el tipo de dato Estudiante el cual permite almacenar la siguiente información: nombre, edad, carrera que cursa. El tipo debe permitir:

1. Crear estudiante con datos ingresados por el usuario. Cuando se desea crear un estudiante sin datos el tipo dispara una excepción.
2. Recuperar y modificar los datos usando propiedades.
3. Proveer una representación textual de un estudiante.
4. Convertir un estudiante a un string.
5. Permitir que un estudiante se pueda utilizar como clave en un diccionario.
6. Comparar estudiantes utilizando los operadores relacionales ==,<,<=,>,>=.

Ejercicio 3: Defina el tipo de dato ContadorLimitado el cual permite almacenar un valor inicial y un valor límite. Un contador limitado va incrementando su valor hasta que llega al límite. Si el contador está en el límite y se le da la orden de contar se mantiene en el valor límite. El tipo debe permitir:

1. Crear un contador con un valor límite. Si un contador se crea sin un valor límite se produce una excepción TypeError.
2. Crear un contador con un valor inicial y un valor límite. Si un contador se crea sin un valor límite se produce una excepción TypeError.
3. Proveer una representación textual de un contador.
4. Convertir a un string un contador.
5. Comparar estudiantes utilizando los operadores relacionales ==,<,<=,>,>=.
6. Convertir un contador a int, float, str.
7. Asignar un contador a otro contador utilizando la asignación =.

Ejercicio 4: Una Pila es una estructura de datos en donde los datos se insertan y suprimen por el tope. Una pila implementa las siguientes operaciones:

1. Crear crea una pila vacía.
2. empty retorna como resultado True si la pila está vacía y False en otro caso.
3. full retorna como resultado True si la pila está llena y False en otro caso.
4. push incorpora una un elemento al tope de la pila.
5. pop elimina un elemento del tope de la pila.
6. top retorna como resultado el elemento que se encuentra en el tope de la pila.

Teniendo en cuenta el funcionamiento de una pila se pide:

1. Defina la clase PilaDeEnteros la cual implementa una pila de enteros.
2. Implemente las funciones antes mencionadas con las siguientes características:
   a) La operación push se realiza con el operador +. Es decir si a una pila p se le quiere insertar el número 2 eso se escribe p+2.
   b) La operación de pop se implementa con el operador –. Es decir si se desea eliminar un elemento de la pila p el programador tiene que escribir p–.
3. Además de las operaciones típicas de pila la clase permite:
   a) Comparar si dos pilas son iguales utilizando el operador =.
   b) Proveer una representación de string para una pila.
   c) Calcular la longitud de una pila.

Ejercicio 5: Defina la clase ListaDeEnterosPares. La clase se comporta como una lista excepto que los elementos son números enteros pares. Un intento de incorporar un valor que no sea un entero par dispara una excepción TypeError.

Ejercicio 6 : Defina la clase MatrizCuadrada la clase permite realizar las siguientes operaciones:

1. Crear una matriz de nxn donde n es ingresado por el usuario.
2. Retornar una fila mediante una operación de indexación como por ejemplo M[1] retorna la fila 1 de la matriz.
3. Recuperar los elementos de una columna de la matriz con un método.
4. Sumar dos matrices cuadradas utilizando el operador +.
5. Restar dos matrices cuadradas utilizando el operador -.
6. Multiplicar una matriz cuadrada por un escalar utilizando el operador *.

7. Construir una representación de string que pueda ser evaluada por *eval*.
8. Imprimir una matriz.