Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2024

PROYECTO No.1

202110511 – Fredy Alexander Esteban Pineda

Resumen

En este proyecto simulábamos una empresa llamada Pisos de Guatemala SA, donde tuvimos que crear un programa para la empresa que era como un robot que permitía poder cambiar el patrón de los pisos. Los pisos tenías forma de matriz y podía variar la cantidad de filas y columnas de cada uno, cada piso contiene azulejos blancos o negros y pueden intercambiarse o dar vuelta.

Pero solo se puede realizar una operación a la vez y cada una de estas operaciones también tienen un valor por lo que habrá que realizar un algoritmo que optimice los movimientos y logre obtener el menor costo posible ya que se ha pactado una cuota fija con los clientes y debe ser rentable el costo del robot.

Por lo que habrá que hacer uso de la programación orientada a objetos y condicionales para poder encontrar la manera de hacer un cambio de patrón optimo y el usuario puedo verificarlo.

Palabras clave

Objeto, Programación orientada a objetos, condición, Lista y ciclo.

Abstract

In this project, we simulated a company called "Pisos de Guatemala SA", where we had to create a program for the company that was like a robot allowing to change the pattern of the floors. The floors had a matrix shape, and it was possible to vary the number of rows and columns of each one. Each floor contained black or white tiles that could be exchanged or flipped.

However, only one operation can be performed at a time, and each of these operations also has a cost, so we need to develop an algorithm that optimizes the movements and achieves the lowest possible cost since a fixed fee has been agreed with the customers, and the cost of the robot must be profitable.

Therefore, we need to make use of object-oriented programming and conditionals to find a way to make an optimal pattern change, and the user should be able to verify it.

Keywords

Object, Object-Oriented Programming, Condition, List and cycle.

Introducción

Como sabemos la programación ha avanzado a lo largo de los años y con ella hemos podido avanzar en la tecnología y lo utilizamos como la solución en muchos aspectos de la vida. La programación orientada a objetos nos ayudara a poder darle solución ya que la empresa Pisos de Guatemala S.A quiere dar una solución a n cantidad de pisos y a n cantidad de clientes.

Por lo que para este problema será necesario hacer uso de la abstracción de las características indispensables para resolver el problema. Utilizaremos objetos por medio de clases en el lenguaje Python con sus diferentes atributos, así como será necesario el hacer uso de estructuras de datos abstractas para el manejo de los objetos que son conceptos que indagaremos más profundamente en este ensayo.

Clases y objetos

En Python, las clases son como moldes para crear objetos. Imagina que una clase es como una receta de cocina para hacer galletas: defines los ingredientes y las instrucciones necesarias para hacerlas. Los objetos, por otro lado, son las galletas reales que haces siguiendo esa receta. Cada objeto creado a partir de una clase tiene sus propias características y comportamientos, pero sigue la estructura definida por la clase.

Por ejemplo, si tienes una clase llamada "Perro", puedes crear objetos individuales que sean perros específicos. Cada perro puede tener su propio nombre, edad, raza, y puede hacer cosas como ladrar, correr y comer. La clase "Perro" define cómo se ven y se comportan los perros en general, pero los objetos individuales creados a partir de esta clase pueden tener sus propias características únicas.

En resumen, las clases en Python te permiten crear plantillas para organizar y estructurar tu código de manera más clara y eficiente. Los objetos son instancias específicas de esas clases que pueden tener sus propios datos y comportamientos, lo que hace que tu código sea más modular y fácil de entender.

```python
class Patron:
    def __init__(self, codigo, azulejos):
        self.codigo = codigo
        self.azulejos = azulejos

    def __str__(self):
        return f"Código: {self.codigo}, Patrón: {self.azulejos}"
```

Figura 1. Creación de una clase y un objeto

Podemos observar la clase Patrón con el objeto patrón y sus atributos.

Listas y nodos

En Python, una lista enlazada es una estructura de datos donde cada elemento, llamado "nodo", contiene un valor y una referencia al siguiente nodo en la lista. Imagina una cadena de cuentas bancarias, donde cada cuenta está conectada a la siguiente mediante una cadena invisible. Los nodos son como los eslabones de esa cadena.

Por ejemplo, puedes tener un nodo que representa el número 5 y que está conectado al siguiente nodo que representa el número 10, que a su vez está conectado al siguiente nodo que representa el número 15, y así

sucesivamente. Cada nodo contiene un valor y una referencia al siguiente nodo en la lista.

Las listas enlazadas son útiles cuando necesitas insertar o eliminar elementos con frecuencia, ya que puedes hacerlo simplemente cambiando las referencias entre los nodos, sin necesidad de desplazar elementos como en una lista convencional. Sin embargo, acceder a elementos específicos en una lista enlazada puede ser menos eficiente que en una lista convencional, ya que necesitas recorrer la lista nodo por nodo hasta llegar al elemento deseado.

En resumen, una lista enlazada en Python es una colección de nodos conectados entre sí, donde cada nodo contiene un valor y una referencia al siguiente nodo en la lista. Esta estructura de datos es útil cuando necesitas realizar inserciones o eliminaciones frecuentes, pero puede ser menos eficiente para acceder a elementos específicos en comparación con otras estructuras de datos como las listas convencionales.

```python
from .Nodo import Nodo

class Lista:
    def __init__(self):
        self.primero = None
        self.size = 0

    def insertar(self, dato):
        nuevo = Nodo(dato) # Creamos un nuevo nodo
```

Figura 2. Creación de una clase Lista

Podemos observar la clase Lista que se relaciona con el objeto nodo .

```python
class Nodo:
    def __init__(self, data):
        self.dato = data
        self.siguiente = None
```

Figura 3. Creación de una clase y objeto nodo

El nodo guarda la información y almacena en la lista y apunta a un nodo siguiente y así es como se enlazan.

Ciclos

En Python, los ciclos son estructuras de control que te permiten repetir un bloque de código una cierta cantidad de veces o hasta que se cumpla una condición específica. Un tipo común de ciclo es el bucle "for", que recorre elementos de una secuencia, como una lista o una cadena de texto, ejecutando el mismo bloque de código para cada elemento. Por ejemplo, puedes usar un bucle "for" para recorrer una lista de números y realizar alguna operación en cada uno de ellos.

Además del bucle "for", Python también ofrece el bucle "while", que se ejecuta mientras una condición sea verdadera. Esto te permite repetir un bloque de código hasta que se cumpla cierta condición de salida. Los ciclos son una herramienta poderosa en Python para automatizar tareas repetitivas y procesar grandes cantidades de datos de manera eficiente.

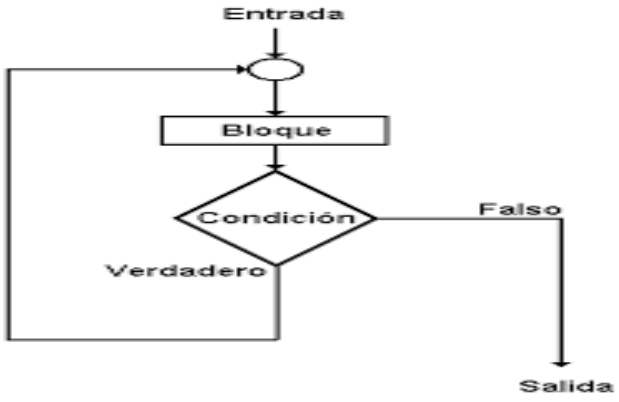

Figura 4. Ciclos en Python

Podemos observar cómo es un ciclo ya que son estructuras que nos ayudan en la búsqueda

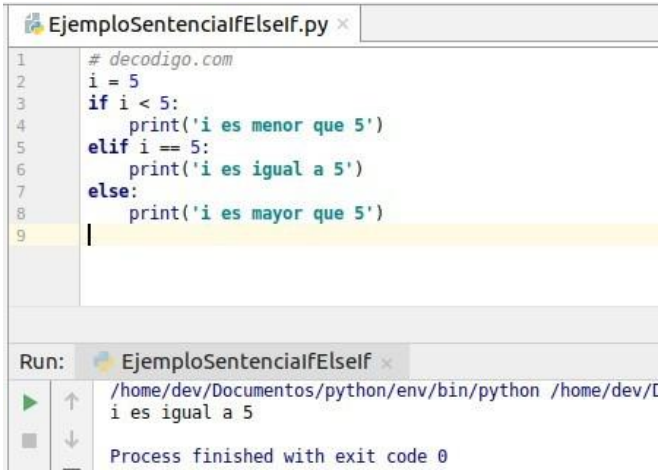

Figura 5. Sentencias en Python

Podemos observar cómo es una sentencia y nos muestra los caminos que toma el programa según la condición.

Sentencias

Las sentencias en Python son instrucciones que realizan acciones específicas cuando se ejecutan. Estas pueden ser simples, como asignar un valor a una variable, o complejas, como estructuras de control que controlan el flujo del programa. Por ejemplo, la sentencia "if" permite ejecutar un bloque de código si se cumple una condición determinada.

Además, Python ofrece sentencias como "while" para repetir un bloque de código mientras una condición sea verdadera, y "for" para iterar sobre elementos en una secuencia. Las sentencias también incluyen operaciones como "def" para definir funciones y "import" para importar módulos externos, lo que amplía la funcionalidad del programa. En resumen, las sentencias en Python son la base de la programación y proporcionan las herramientas necesarias para crear programas funcionales y eficientes.

Conclusiones

Pudimos observar que el lenguaje Python cuenta con muchas herramientas para la resolución de problemas.

Enfatizando, la abstracción es gran parte de la resolución de un problema de programación para poder llegar a un resultado sólido.

La programación orientada a objetos nos enseña que es más importante identificar los elementos que van a constituir el programa y que actúan como una red entre ellos.

Referencias bibliográficas

Máximo 5 referencias en orden alfabético.

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.

Smith, J. (2020). Python Programming for Beginners (3rd ed.). Wiley.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

Flanagan, D. (2011). JavaScript: The Definitive Guide. O'Reilly Media.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). Database System Concepts (6th ed.). McGraw-Hill.

Extensión:

Diagrama de clases de la solución del programa