Lista de Conceitos: POO

Agosto, 2019

1 Pensar numa outra horinha

- Há muitos paradigmas de programação atualmente. Por que usar programação orientada a objetos?
- Pensando em Python, temos uma linguagem estaticamente tipada, ou seria dinâmica? Por quê?
- Python tem tipagem forte ou tipagem fraca?
- Um programa é alocado estaticamente na memória de seu computador. Há momentos em que há alocação dinâmica de memória? Quais são eles?
- O que é "herança" e como podemos usá-la? Para que serve?
- Python possui suporte para herdar atributos de mais de uma classe? Qual o nome disso?
- Você já ouviu falar de Polimorfismo? Defina-o usando suas palavras.
- Sabendo que nem todas as pessoas que tem contato com programação saberão realmente boas práticas e sabendo que códigos maiores e com excesso de redundâncias podem vir a ser confusos, usar herança se concretiza como uma boa prática?
- O que é uma Tupla?
- O que é um Dicionário?
- O que são Tipos Genéricos e como podemos usá-los em Python?

2 Uns exercícios massas

1.1. Implemente um algoritmo de busca binária usando orientação a objetos em Python. (Nota: você terá que ordenar sua lista. Por quê?)

1.2. Opcional: usando notação Big O, quão eficaz é o algoritmo acima se comparado a uma busca simples?

2.1. Crie uma pilha padrão usando orientação a objetos usando um tipo qualquer de sua preferência.

2.2. Refaça o exercício acima usando tipos genéricos.

2.3. Crie uma classe que seja uma classe-filha da classe criada no exercício anterior.

3.1. Opcional: Faça um sistema bancário em que tenha uma classe geral Conta. Devem haver outras duas classes, ContaCorrente e ContaPoupança, que irão herdar da classe principal. As especificações estão abaixo:

- class Conta: * Variáveis globais/atributos: número da conta, CPF do cliente, agência.
- class ContaCorrente: * Variáveis globais: número da conta, CPF do cliente, agência. * Métodos: saque(), depósito(), transferência().
- class ContaPoupança: * Variáveis globais: número da conta, CPF do cliente, agência. * Métodos: rendimento().

(Nota: a descrição ficou vaga propositalmente. Proponha uma forma de realizar o exercício acima da maneira que achar melhor, apenas priorize pela organização e limpeza de seu código. Não se esqueça também de comentar seu código)

3 Bibliografia e ajuda

De repente tu curte ler pra reter mais o conteúdo. Recomendo *Grokking Algorithms*, de Aditya Y. Bhargava, e *Learning Python*, de Mark Lutz!