Orientação a objetos em Python

Por ser uma linguagem multiparadigma, Python se encaixa no conceito "O pau para toda obra, mestre de ninguém", com isso não é a ferramenta mais avançada para casos específicos. Assim é o caso da OO em Python, apesar de possuir um suporte para o paradigma, algumas funcionalidades são rasas e outras simplesmente inexistentes.

Abstração

É um pilar que acaba ficando mais a cargo do programador do que da ferramenta, desse modo irá variar no contexto e nível de experiência que o programador está inserido.

Encapsulamento

Python não implementa de forma prática o encapsulamento. Não existem atributos privados ou protegidos, é tudo público. Existe uma convensão adotada pela comunidade que determina que se o atributo começa com “__” (dois underlines) ele é privado. Entretanto o interpretador não garante essa mesma integridade.

Herança

O Python implementa herança fazendo o uso de uma sintaxe simples, apenas basta informar a superclasse entre parênteses e após o nome no momento da declaração da subclasse. Em caso de heranças múltiplas só é preciso adicionar as superclasses separadas por vírgula.

Exemplo está na pasta herança.

Polimorfismo

O polimorfismo é a sobre escrita de métodos em uma situação de heranças, onde a subclasse define um novo comportamento para um método que já está presente na superclasse. Não sendo necessário fazer uso de annotation e decorators.

Exemplo está na pasta polimorfismo.

Composição e agregação

Python suporta ambos os conceitos de forma simples, é preciso definir um atributo de uma classe como sendo do tipo da outra classe.

Exemplo está na pasta composiçao agregaçao.

Métodos estáticos

Métodos estáticos são aqueles que pertencem a classe e não ao objeto, podem ser chamados sem a necessidade de uma instância. Isso é denotado com o uso do decorator “@statimethod”.

Exemplo está na pasta estatico.

Classes e métodos abstratos

Uma classe é dita abstrata caso ela possua um ou mais métodos abstratos, e métodos abstratos são métodos especiais que não possuem implementação e devem obrigatoriamente ser implementados por subclasses. Classes abstratas não podem ser instanciadas como objetos e servem apenas para serem herdadas. Em Python, uma classe precisa herdar o módulo “abc“ (abstract base classes). Métodos abstratos devem ser indicados pelo decorator “@abc.abstractmethod”.

Exemplo está na pasta abstrato.

Cálculo Lambda em Python

1**.** Quem foi o idealizador do cálculo lambda e quando ele foi proposto (aproximadamente)?

Alonzo Church, final da década de 30.

2. Qual a relação entre cálculo lambda e máquina de Turing?

Ambos são formas de representar formalmente o que conhecemos como computação. Além disso, as duas coisas são equivalentes, no que chamamos de hipótese Church-Turing.

3. O que é o cálculo lambda? Em que ele foi útil na computação?

O cálculo lambda é uma notação criada por Church para realizar computações de uma forma puramente funcional e matemática. Além de avançar em estudos de computabilidade, o cálculo lambda também é base para as linguagens de programação funcionais modernas.

4. O que são variáveis livres (independentes)? E variáveis vinculadas (dependentes)? Cite exemplos de variáveis livres e vinculadas em uma expressão lambda.

Variáveis livres são aquelas externas a uma função lambda específica, isso significa que elas não estão ligadas a um valor que deve ser passado para substituição na função, mas também significa que ela precisa de um valor global previamente definido.

Enquanto isso as variáveis vinculadas são aquelas que tem o valor dependente de um argumento que será aplicado durante a avaliação da expressão.

Em “(lx + x y) 2” podemos ver que “y” é livre, enquanto “x” é vinculada.

5. O que significa currying em cálculo lambda? Exemplifique.

Uma curried function é uma função definida para receber seus argumentos de forma sequencial, um de cada vez, ao invés de receber todos ao mesmo tempo.

Por exemplo, podemos ter uma função normal “λx.λy -> x+y” que ao ser passada para sua *curried form* é escrita como “λx -> (λy -> x + y)”.

6. O que significa uma expressão ser um "combinador", em cálculo lambda? Cite um exemplo de expressão que é um combinador.

Significa que uma expressão é um combinador se ela não possui nenhuma variável livre.
Exemplo: λx.λy.xyx

8. O que significa dizer que duas expressões lambda são α-equivalentes? Exemplifique.

Elas variam apenas no nome das variáveis dependentes. Exemplo: λx.x e λy.y são α-equivalentes, pois ambas x e y são variáveis dependentes, vinculadas a uma abstração.

9. O que é a operação de α-conversão (α-renomeação)? Exemplifique.

É a operação de “trocar” (renomear) apenas variáveis dependentes (vinculadas). Exemplo: (λx.x) {f/x} ===> (λf(x) {f/x}) ===> (λf(f)) ===> (λf.f)

10. O que é Redução-Beta (β-Redução)?

E é uma expressão para x, isto é, λx. E é uma função de x. Assim, (λx.E)N refere-se à aplicação da função para a entrada N. Formalmente: (λx.E)N β-reduz para E[x → N]. Exemplo: (λx.x)y ===> x[x → y] ===> y

11. O que significa dizer que uma expressão lambda está em sua "forma normal"? Exemplifique com uma expressão lambda em sua forma normal.

Quando não for mais possível efetuar nenhuma redução, então dizemos que estamos na forma normal.