Programação Orientada A Objetos Com Python
Linguagens e Paradigmas de Programação
Sistemas de Informação
Alexandre de Sá Carneiro Wanderley[1]
Nascimento, F. L [2]

Programação Orientada a Objetos

[1]alexandre@ifce.edu.br
[2]lucasnasm@gmail.com

Sumário

Resumo: Nesta pesquisa apresenta-se um breve histórico, conceitos e exemplos de aplicações que seguem o paradigma de programação orientado a objetos. Para tanto, objetiva-se investigar como o paradigma OO aplica-se a linguagem de programação *Python*, relacionando os conceitos desse paradigma com aplicações prática construídas utilizando-se *Python*. Trata-se de um estudo exploratório descritivo com um levantamento sobre o assunto na literatura, buscando identificar os principais autores que exploraram o tema e que utilizaram desses em suas pesquisas. Portanto, conclui-se (Em construção...)

Palavras-chave: Paradigma. Programação. Python. Computação.

1 Introdução

o crescimento do desenvolvimento de software e a exigência do mercado por aplicações cada vez mais complexas levou a uma expressiva necessidade por metodologias que possibilitassem abstrair e modularizar as estruturas dos programas existentes. Assim, pode-se destacar linguagens de programação que suportam orientação a objetos: Haskell, Java, C++, Python, PHP, Ruby, Pascal, entre outras.

2 Histórico

Em 1967, Kristen Nygaard e Ole-Johan Dahl, do Centro Norueguês de Computação em Oslo, desenvolveram a linguagem Simula. Derivada do Algol, o Simula I e Simula 76 podem ser consideradas as primeiras linguagens a introduzir os primeiros conceitos de orientação a objetos. Em princípio eram usadas para realizar simulações do comportamento de partículas de gases.

Os conceitos de objetos, classes e herança nesse estágio de desenvolvimento eram tratados não necessariamente da forma que que se conhece hoje hoje, por exemplo, o conceito de herança surgiu no Simula 67, pois até então falava-se apenas em *subclassing*.

Em 1970, Alan Kay, Dan Ingalls e Adele Goldberg, do Centro de Pesquisa da Xerox, desenvolveram a linguagem totalmente orientada a objetos.

Em 1979–1983, Bjarne Stroustrup, no laboratório da AT & T, desenvolveu a linguagem de programação C++, uma evolução da linguagem C

Smaltalk - muitos programadores da épocо não se sentiam confortáveis em ultilizar um novo paradigma de programaç19ão.

1983 C++, ObjetC, 86 objectpascal java c e objectivec.. gratúita e multiplataforma, ”escrava uma vez execute em qualquer lugar.”

3 Conceitos

Neste capítulo tem-se uma abordagem sobre os principais conceitos que cerca a programação orientada a objetos , destacando princípios e características; um breve contexto histórico acerca da evolução das linguagens OO, com ênfase nas aplicações definidas em uma escala evolutiva das linguagens. Em seguida tem-se uma explanação sobre os pilares da programação orientada a objetos uso de grafo aplicado à análise de relações sociais, no qual são abordados os conceitos que envolvem a teoria de grafo e sua aplicação na análise de relações sociais, mostrando as principais características e tecnologias empregadas e um estudo sobre os modelos de visualização de grafos. Ao final, são apresentadas pesquisas

relacionadas ao presente estudo que abordam o uso de grafos com propósito de expor a interação entre os participantes de fórum, a partir de estratégias que se relacionam com a teoria dos grafos.

3.1 Abstração

A criação de uma classe abstrata em Python é bastante útil e serve para definir o esqueleto para uma subclasse. Contudo esse conceito não é nativo do python e a sua implementação é melhor definida por meio da biblioteca ABC. Abaixo é apresentado uma simples e eficiente forma de uso do biblioteca padrão abc do Python 3.6. Cabe ser observado que esse módulo foi adicionado ao Python 2.6 definido na proposta: PEP 3119.

```
1    from abc import ABC, abstractmethod
2
3        class AbstractOperation(ABC):
4
5            def __init__(self, operand_a, operand_b):
6                self.operand_a = operand_a
7                self.operand_b = operand_b
8                super(AbstractOperation, self).__init__()
9
10           @abstractmethod
11           def execute(self):
12               pass
```

Neste código pode ser observado

3.2 Encapsulamento

3.3 Herança

3.4 Polimorfismo

4 Propriedades Principais

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra

ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

- Símbolos literais, como x, y $\&c.$, representando as coisas como sujeitos de nossas concepções.
- Sinais de operação, como $+, -, *$, representando as operações pelas quais as concepções das coisas são combinadas ou solucionadas de modo a formar novas conceitos envolvendo os elementos.
- O sinal de identidade, $=$.
- E estes símbolos da lógica fazem parte dos objetos e seguem a leis definidas, em parte concordando com diferentes leis dos símbolos ligados a ciência da Álgebra.

4.1 Código em Python

Exemplo de aplicação em *Python*.

```
1     class Poupanca(Conta):
2         def __init__ (self ,numero):
3             super().__init__(numero)
4             self.__rendimento = 0.0
5
6         def consultar_rendimento(self):
7             return self.__rendimento
8
9         def gerar_rendimento(self,taxa):
10            self.__rendimento += super().consultar_saldo()*taxa/100
11    conta = Poupanca(1)
12    conta.creditar(200.0)
13    conta.gerar_rendimento(10)
14    print(conta.consultar_saldo())
15    print(conta.consultar_rendmento())
```

5 Conclusão

Com suporte nos recursos didáticos e a partir do que foi explanado, espera-se que com esse relatórios sucinto ter dado uma visão geral da história da álgebra booleana. Como mencionado anteriormente, foram levantados alguns estudo, consequente não são suficiente para cobrir os diferentes assuntos que surgiram neste processo e no desenvolvimento do trabalho. Assim para um estudo mais detalhado sobre o assunto, faz-se necessário uma pesquisa mais profunda sobre o trabalho de Boole e complementando com o estudo de De Morgan. Contudo, acredita-se que como definida, e no que se prepôs essa pesquisa o objetivo tenha sido alcançado.

6 Bibliografia

Referências

[1] Janet Heine Barnett. “Origins of Boolean algebra in the logic of classes: George Boole, John Venn and CS Peirce”. Em: *AMC* 10 (2011), p. 12.

[2] J. Gregg. *Ones and zeros: understanding Boolean algebra, digital circuits, and the logic of sets*. IEEE Press understanding science & technology series. IEEE Press, 1998. ISBN: 9780780334267.

[3] PF Strawson. “The Mathematical Analysis of Logic. By George Boole. Pp. 82.(Oxford: Basil Black well. 1948. Price 7s. 6d.)” Em: (1950).

[4] R.J. Tocci et al. *Sistemas digitais: princípios e aplicações*. PRENTICE HALL BRASIL, 2011. ISBN: 9788576050957. URL: https://books.google.com.br/books?id=QgYxNAAACAAJ.