

INTRODUÇÃO A ORIENTAÇÃO A OBJETOS

A linguagem Python foi criada para o desenvolvimento orientado a objetos, o que torna a programação mais simples e com maior facilidade em desenvolver grandes projetos com outros programadores.

A orientação a objetos é mais estável e dá maior praticidade para reutilização do código.

TÉCNICAS DE ORIENTAÇÃO A OBJETOS NO PYTHON

Conceitos de orientação a Objetos:

Classes

Definição do objeto

- Conjunto de atributos e funções utilizado como modelo para criação dos objetos

Objetos

Instância de uma classe

Variável que possui todas as características comuns á classe, porém, com valores diferentes em seus atributos

Abstração

Acesso às utilidades da Classe

Ação de utilizar mensagens para acessar os recursos de uma classe

Atributos

Características da classe

Os atributos são características do elemento que a classe representa

Funções

Ações do objeto com retorno

Função nativa do Python que retorna um valor declarado dentro da classe

Métodos

Ações do Objeto

Função nativa do Python que não possui retorno declarado dentro de uma classe

Exceção

Tratamento de erros

Controla o fluxo de execução durante um erro

Mensagem

Chamada a um atributo, método ou função

Herança

Superclasse ou Subclasse

Uma classe pode herdar atributos, métodos e/ou funções de outra

Encapsulamento

Níveis de acesso

O encapsulamento permite que os atributos sejam vistos somente nas classes onde foram declarados, definindo o nível de acesso de atributos, métodos ou funções.

Polimorfismo

Sobrescrita

Escolher entre os atributos, métodos e/ou funções que sobrescrever ou que foram sobrescritos.

CRIAÇÃO DE OBJETOS

É sempre importante ter os conceitos da orientação a objetos fixados em mente

Classe: Podemos dizer que a classe é o projeto do objeto, contendo o código de programação

Objeto: É a execução do código de uma classe.

Quando executamos o código de uma classe é criado um novo objeto na memória.

Uma nova instância no objeto é um tipo abstrato de informação de um novo tipo de dado.

Instância: É o objeto sendo executado.

Quando criamos um objeto, afirmamos que estamos criando uma instância dele.

CRIANDO CLASSES EM PYTHON

Dentro do seu IDE favorito, crie um novo arquivo python e atribua o nome 'Cliente'

1. Abra a IDE
2. Crie um novo arquivo com a extensão .py
3. Atribua o nome Clientes
4. Salve o arquivo

Estrutura da classe:



Definimos a classe iniciando com `class`

Adicionamos o nome da classe, neste caso se chama `Cliente`

Ao final da declaração adicionamos o símbolo `:` para finalizar a declaração

O que estiver indentado e dentro do escopo da classe, será atribuído a essa classe, neste caso foi utilizado o código `pass` para apenas para preencher o escopo.

Classes são um tipo abstrato de dados, sendo assim, haverá valores e esses necessitam de funções específicas para serem manipulados.

Temos dois tipos básicos de membros que compõe uma classe:

Atributos (propriedades)

Os atributos armazenam as características de uma classe.

Os atributos são as declarações de variáveis da classe

Métodos

São ações da classe, suas funções.


Representam os estados e ações dos objetos quando instanciados.

MÉTODO CONSTRUTOR

O método construtor é definido de forma implícita ou explícita por todas as classes e, como o próprio nome já cita, é utilizado para construir um objeto.

Todas as vezes que um objeto estiver sendo criado (instanciado), é por meio do construtor que ele será inicializado.

Inserindo o método construtor na classe Cliente



```
1 class Cliente:  
2     def __init__(self):  
3         pass
```

A palavra `def` é utilizada para iniciar a declaração do método

Para definir o construtor, é utilizado a sintaxe: `__init__`


O `__init__` é um método especial que sempre será chamado quando criarmos um objeto da classe.

Incluimos o parâmetro obrigatório `self` que, resumidamente, exporta as características do objeto.

A palavra reservada representa o objeto em si, portanto, sempre que quisermos especificar atributos de objetos, devemos associá-los à palavra reservada `self`.

COMO ADICIONAR ATRIBUTOS A CLASSE

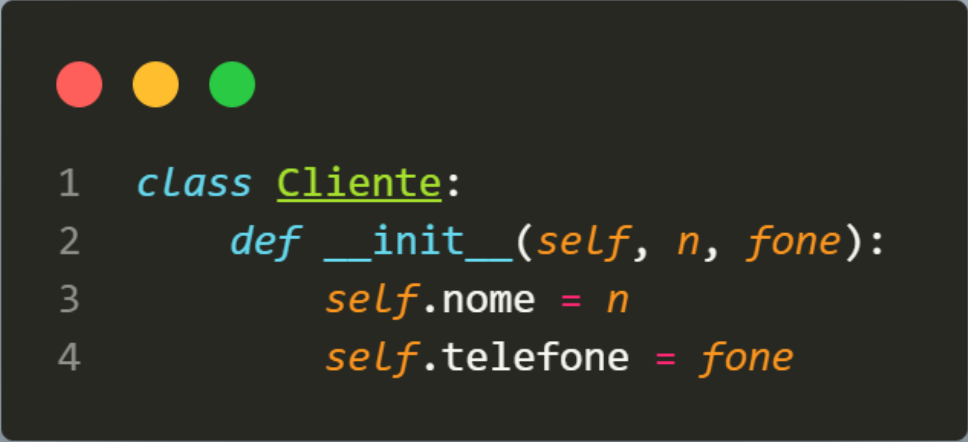
Para adicionar atributos a uma classe, basta definir o nome do atributo acompanhado da palavra reservada `Self`, no método especial denominado `__init__` do método construtor.



```
1 class Cliente:
2     def __init__(self):
3         self.nome
4         self.telefone
```

COMO PERSONALIZAR O MÉTODO CONSTRUTOR

O método construtor da classe pode conter um conjunto de parâmetros. Com isso podemos determinar os valores para inicialização dos atributos.



```
1 class Cliente:
2     def __init__(self, n, fone):
3         self.nome = n
4         self.telefone = fone
```

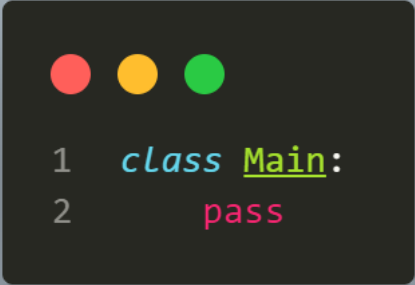
Com o parâmetro Self são passados os parâmetros que serão utilizados para inicialização dos atributos.

Os valores definidos como parâmetro serão passados para os atributos.

CRIANDO REFERÊNCIA DE CLASSES

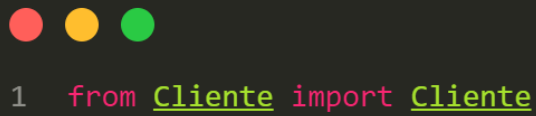
Para instanciar o objeto de uma classe para outra, devemos criar a referência da classe que será instanciada, neste caso, será necessário criar um novo arquivo python.

- Crie um novo arquivo python e atribua o nome como main.py
- Dentro deste novo arquivo, defina uma classe com um escopo vazio, igual está na sintaxe abaixo:



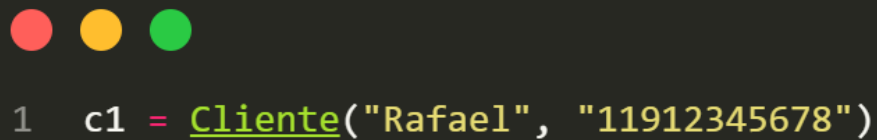
```
1 class Main:
2     pass
```

- Dentro do arquivo Main, importe a classe do outro arquivo utilizando a sintaxe **from Cliente import Cliente**



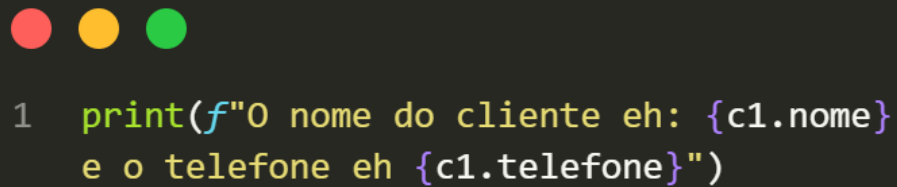
```
1 from Cliente import Cliente
```

- Após importar a classe do arquivo Cliente, crie um novo objeto, passe a classe que será instanciada e os parâmetros da classe



```
1 c1 = Cliente("Rafael", "11912345678")
```

- Para verificar o resultado, crie uma função Print e defina o objeto com os parâmetros para exibir na tela o que foi definido



```
1 print(f"O nome do cliente eh: {c1.nome}  
e o telefone eh {c1.telefone}")
```

Resultado:



```
1  O nome do cliente eh: Rafael e o telefone eh 11912345678
2
3  [Done] exited with code=0 in 3.119 seconds
```

PRATICANDO: Projeto Controle Bancário

Vamos criar um novo arquivo e definir uma nova classe contendo os parâmetros: titular, número, saldo.

- Crie um novo arquivo nomeado Conta.py
- Crie a classe Conta
- Crie o método construtor
- Passe os parâmetros: self, titular, número, saldo
- Defina o saldo com valor 0
- Defina o número com valor número
- Defina titular com valor titular
- Vá até o arquivo Main.py e importe a biblioteca Conta.py
- Crie um novo objeto e instancie com a classe Conta
- Passe os parâmetros desejados
- Crie uma função print e exiba o resultado na tela.

Resultado:



```
1  class Main:
2      pass
3
4  from Conta import Conta
5  c1 = Conta("Rafael", "12345", "R$100,00")
6  print(f"Titular: {c1.titular}\nNumero da conta: {c1.numero}\nSaldo disponivel: {c1.saldo}")
```

