



IN

DEV FULL STACK- PYTHON
AULA 3 - PYTHON OO

Orientação a Objetos

Você aprendeu sobre função...

- As funções ajudam a organizar o código;
- Devemos criar funções específicas para cada ação;
- Ex (Sistema médico):

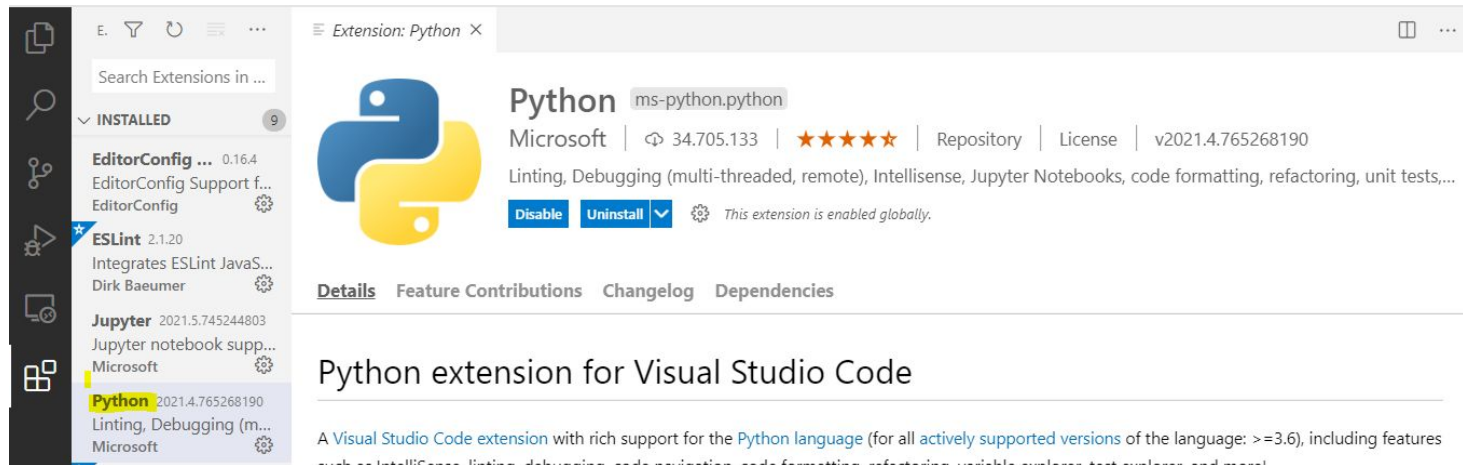
- MarcarConsulta() → Estamos falando sobre Consulta
- ConsultarDisponibilidade() → Estamos falando sobre Médico
- Receitar() → Estamos falando sobre Receita
- MarcarDisponibilidadeMedica() → Estamos falando sobre Médico

Arquivos diferentes

- Medico.py
- Consulta.py
- Receita.py
- Paciente.py

VS Code com Python

- Instalar extensão Python no vs code;



Orientação a Objetos

Orientação a Objetos

- Surgiu como uma alternativa a essas características da programação estruturada;
- Objetivo: aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real
- Baseia-se principalmente em dois conceitos chave: **classes** e **objetos**.

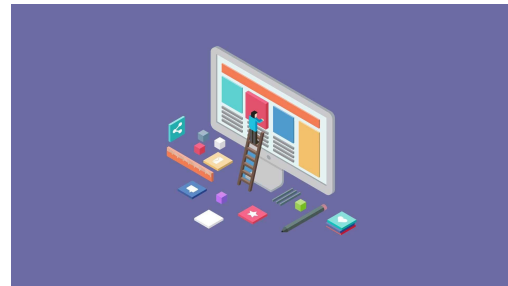
Classe



MinhaClasse.py

Classes

- Forma de definir um tipo de dado;
- Formada por dados e comportamentos;
- Os dados são chamados de Atributos e são as variáveis que representam um item do mundo real;



Classes

Exemplos:

Carro
marca modelo ano automatico
acelerar() frear() ligar()

ContaCorrente
numero agencia banco senha saldo
depositar() sacar() transferir()

Aluno
nome endereco idade cpf
matricular() estudarModulo()

Classes

- Exemplos práticos

 Carro.py

```
1 class Carro:
```

 ContaCorrente.py

```
1 class ContaCorrente:
```

 Aluno.py

```
1 class Aluno:
```

Classes - Construtor e Atributos

- **Construtor:**
 - Na criação do objeto obrigará atribuição de valores iniciais para atributos;
- **Atributos:**
 - Características importantes para a classe

ContaCorrente
numero agencia banco senha saldo
depositar() sacar() transferir()

Classes - Construtor e Atributos

- Construtor
- Atributos

```
class ContaCorrente:  
    def __init__(self, numero, senha, agencia='555', banco='01', saldo=15.0):  
        self.numero = numero  
        self.senha = senha  
        self.agencia = agencia  
        self.banco = banco  
        self.saldo = saldo
```



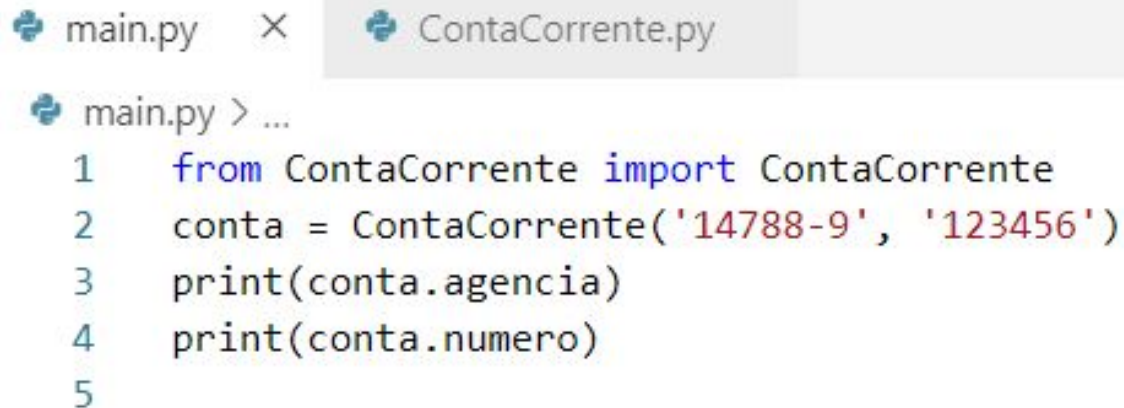
Construtor

ContaCorrente

numero
agencia
banco
senha
saldo

depositar()
sacar()
transferir()

Objetos



```
main.py  X  ContaCorrente.py  
main.py > ...  
1  from ContaCorrente import ContaCorrente  
2  conta = ContaCorrente('14788-9', '123456')  
3  print(conta.agencia)  
4  print(conta.numero)  
5
```

Atribuição de valores
Item do mundo real

Self

```
from ContaCorrente import ContaCorrente
```


```
conta = ContaCorrente('14788-9', '123456')  
print(conta.agencia)  
print(conta.numero)
```

Objeto 1

```
conta2 = ContaCorrente('14733-6', '654321')  
print(conta.agencia)  
print(conta.numero)  
print(conta.saldo)
```

Objeto 2

Self



```
from ContaCorrente import ContaCorrente

{conta} = ContaCorrente('14788-9', '123456')
print(conta.agencia)
print(conta.numero)

conta2 = ContaCorrente('14733-6', '654321',
print(conta.agencia)
print(conta.numero)
print(conta.saldo)

class ContaCorrente:
    def __init__(self, numero, senha, agencia, banco, saldo):
        self.numero = numero
        self.senha = senha
        self.agencia = agencia
        self.banco = banco
        self.saldo = saldo
```


Métodos

- Comportamentos que o objeto pode ter;
- Modifica os valores dos atributos do objeto;

Carro
marca modelo ano automatico
acelerar() frear() ligar()

ContaCorrente
numero agencia banco senha saldo
depositar() sacar() transferir()

Aluno
nome endereco idade cpf
matricular() estudarModulo()

Métodos

```
class ContaCorrente:
```

```
    def __init__(self, numero, senha, agencia='555', banco='01', saldo=15.0):  
        self.numero = numero  
        self.senha = senha  
        self.agencia = agencia  
        self.banco = banco  
        self.saldo = saldo
```

```
    def sacar(self, valor):  
        self.saldo = self.saldo - valor
```

Inserir condicional

```
from ContaCorrente import ContaCorrente  
conta = ContaCorrente('14788-9', '123456')  
print(conta.agencia)  
print(conta.numero)
```

```
conta2 = ContaCorrente('14733-6', '654321')  
print(conta.agencia)  
print(conta.numero)  
print(conta.saldo)
```

```
conta.sacar(10)
```

```
print(conta.saldo)
```

Realizar Transferência

```
def trasnfere(self, conta, valor):  
    self.saldo -= valor  
    conta.saldo += valor
```

```
from ContaCorrente import ContaCorrente  
conta = ContaCorrente('14788-9', '123456')  
print(conta.agencia)  
print(conta.numero)  
  
conta2 = ContaCorrente('14733-6', '654321')  
print(conta.agencia)  
print(conta.numero)  
print(conta.saldo)  
  
conta.sacar(10.0)  
conta.trasnfere(conta2, 5.0)  
  
print(conta.saldo)  
print(conta2.saldo)
```

Atividade

Criar uma classe de Carro, que corre até no máximo 120km/h. O carro deve ser cadastrado com marca, modelo, ano, velocidade, se está ligado ou não e se é automático ou não.

O carro deve conter funcionalidades de:

- ligar
- Acelerar: apenas se o carro estiver ligado (uma quantidade que não passe da velocidade máxima de 120.
- desligar
- verificarMacha com as regras abaixo:
- **1ª marcha:** 0 a 20km
- **2ª marcha:** ao atingir 20 km/h;
- **3ª marcha:** entre 30 e 35 km/h;
- **4ª marcha:** entre 45 e 50 km/h; e.
- **5ª marcha:** acim de 60 km/h.
-

Atividade

The logo consists of the letters 'IN' in a white, serif font, centered within a solid red square. The background of the entire image is a vibrant red with abstract, overlapping geometric shapes in various shades of red and orange, creating a sense of depth and movement.

IN

71 3901 1052 | 71 9 9204 0134

@infinity.school

www.infinityschool.com.br

Salvador Shopping Business | Torre Europa Sala 310
Caminho das Árvores, Salvador - BA CEP: 40301-155