FICHA TRABALHO 6

MODALIDADE:	Aprendizagem +	Escolha um item.	
CURSO:	Técnico/a Especialista em Gestão de Informação e Ciência dos Dados		
UFCD:	Programação em Python	CÓDIGO UFCD:	10805
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Exercícios de classes com a linguagem Python

Classes em Python

Uma classe permite definir um objeto com base nos estados (os campos ou variáveis) e respetivas ações/métodos que definem o seu comportamento (funções).

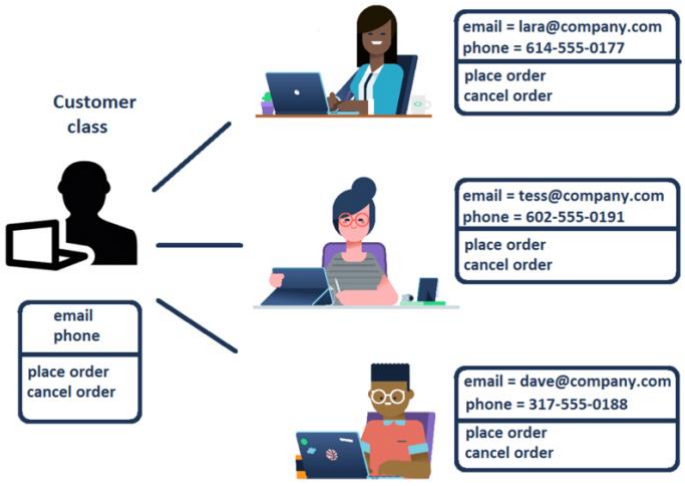
Ou seja, uma classe permite a criação de objetos com determinadas estruturas e comportamentos específicos.

Podemos pensar numa classe como um esboço (estrutura) de uma pessoa.

A classe contém todos os detalhes sobre as informações da pessoa (definidos pelo programador), como os estados e comportamentos.

Com base nessas descrições, construímos a estrutura de representa a estrutura da pessoa.

A partir da classe, podemos criar as informações de várias pessoas (com dados diferentes), com base na classe pessoa, do qual, compartilham a mesma estrutura entre ambas.



1.1 – Criação de classes

1. ***Uma classe tem sempre um nome***, onde a primeira letra deve sempre começar ***por maiúscula***;

```
# 1 - declaração do nome da classe
class Pessoa:
```

2. ***Fazer a implementação do construtor de inicialização*** (responsável por receber os dados para contruir o objeto). Este ***é o primeiro "método"*** que se ***constrói dentro da classe e*** que será executado quando ***um novo objeto é instanciado***.

   Dentro do construtor, devemos indicar ***os atributos*** da estrutura da classe.

   Os atributos são divididos em duas partes:

   1. ***Variáveis de instância***: definidos dentro de métodos e pertencem apenas ao objeto criado. No exemplo da classe Pessoa, id, nome e idade são ***atributos de instância*** porque são definidos dentro do método __init__ e pertencem a cada objeto com base na classe Pessoa.

```
def __init__(self, id, nome, idade):
```

   2. ***Variáveis da classe***: são as variáveis compartilhadas entre todas as instâncias da classe e podem ser acedidos diretamente pela classe ou por qualquer uma de suas instâncias. Colocamos a ***palavra reservada self*** atrás do nome para indicar que é uma variável de classe e não confundir com a variável que está declarada nos parenteses do método __init__):

```
self.ID = id
self.Nome = nome
self.Idade = idade
```

Metodo __init__ completo:

```
# 2 - Construtor de inicialização e variáveis de instância
def __init__(self, id, nome, idade):
    self.ID = id
    self.Nome = nome
    self.Idade = idade
```

4. ***Por último, criamos os métodos da classe*** que são responsáveis ***pelo comportamento das instâncias/objetos quando recebem mensagens provenientes de alguma parte do programa***;

   Exemplo 1: Criar o método para mostrar os dados os dados do objeto

```python
# 3 - Método para mostrar os dados das variáveis do objeto
def MostraDadosPessoa(self):
    print(f"ID: {self.ID} ,Nome: {self.Nome}, Idade: {self.Idade}")
```

   Exemplo 2: Verificar se a idade é maior de 18 anos

```python
def VerificaIdade(self):
    if(self.Idade >= 18):
        print("É maior ou igual que 18 anos.")
    else:
        print("É menor de idade.")
```

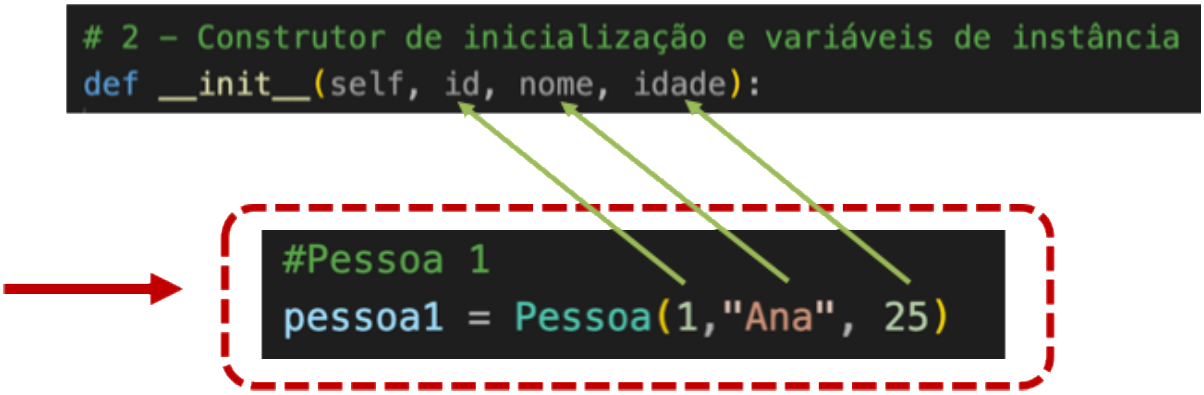
Código da classe completo (com a combinação dos 4 passos):

```python
# 1 - Declaração do nome da classe
class Pessoa:

    # 2 - Construtor de inicialização e variáveis de instância
    def __init__(self, id, nome, idade):
        self.ID = id
        self.Nome = nome
        self.Idade = idade

    # 3 - Método para mostrar os dados das variáveis do objeto
    def MostraDadosPessoa(self):
        print(f"ID: {self.ID} ,Nome: {self.Nome}, Idade: {self.Idade}")

    def VerificaIdade(self):
        if(self.Idade >= 18):
            print("É maior ou igual que 18 anos.")
        else:
            print("É menor de idade.")
```

1.2 – Utilização das classes

Depois de criada a nossa classe, vamos utilizar a mesma. Não esqueçam de que os ***objetos*** são ***instâncias de uma classe***. Para poder utilizar uma classe, temos de seguir os seguintes passos:

Passo 1 - Criar uma variável e invocar a classe pretendida para criar um novo objeto (instância) da classe ***Pessoa*** e passar os valores pedidos no construtor de inicialização __init__***:***



Passo 2 - A classe recebe os novos dados do objeto e de seguida, e as variáveis da classe passam a informação para todos os métodos. Neste exemplo, vamos invocar o método para mostrar os dados. Para tal, vamos escrever a sequência:

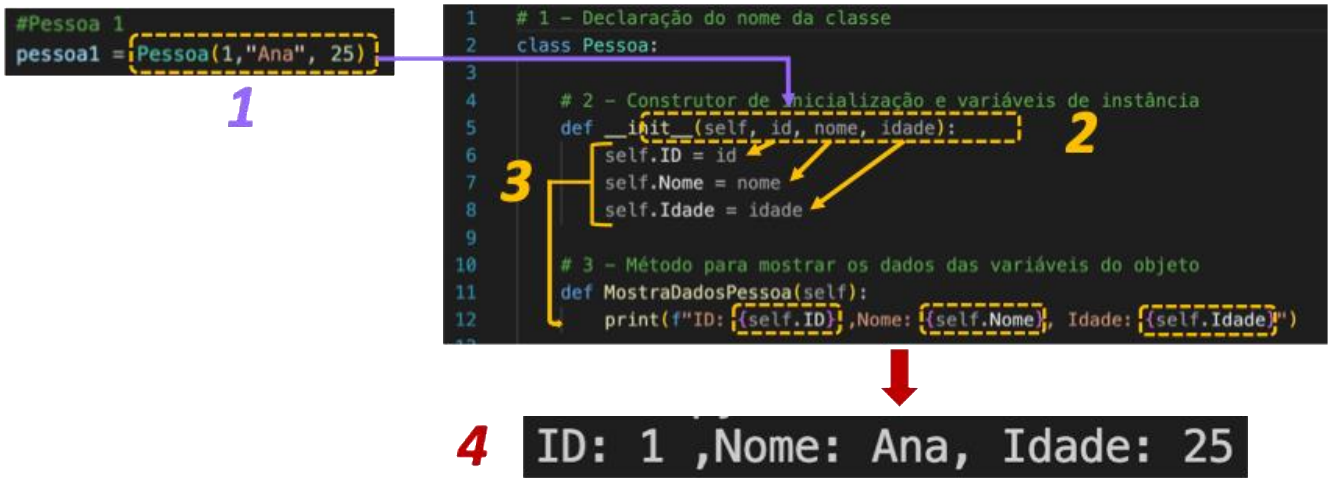
- o nome do objeto criado;
- Colocar . (ponto final) para aceder aos valores dentro do objeto;
- Chamar o método definido na classe com o nome ***MostraDadosPessoa();***

```
#Pessoa 1
pessoa1 = Pessoa(1,"Ana", 25)
pessoa1.MostraDadosPessoa()
```

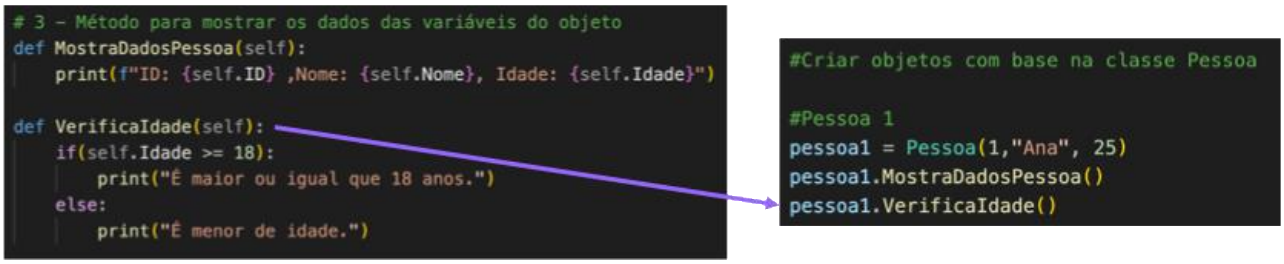
Este por sua vez, vai chamar o método em causa, que neste caso, vai mostrar a mensagem com os dados das variáveis da classe que forma passados:

```
ID: 1 ,Nome: Ana, Idade: 25
```

Como funciona por detrás do código:



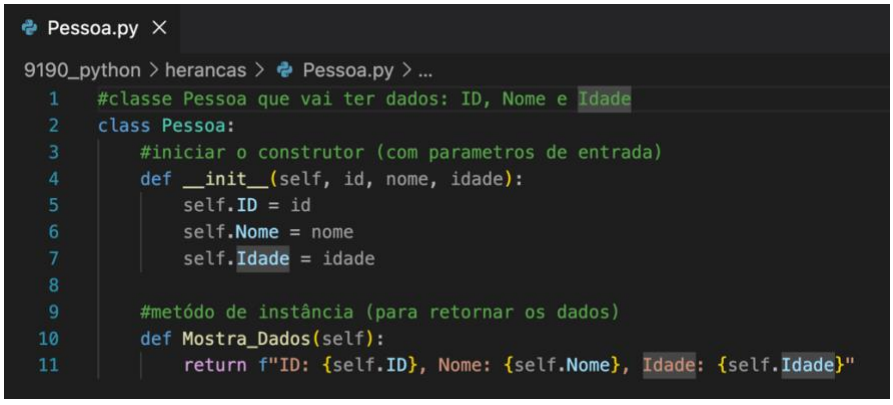
Passo 3 – Podem ***criar outros métodos na classe para outras finalidades***. Por exemplo, usar a variável de classe idade e criar o método para verificar se a idade é maior ou igual a 18 anos:



Importação de classes (em ficheiros separados)

Também podemos separar o código e colocar as classes em ficheiros separados e importar as mesmas quando necessárias.

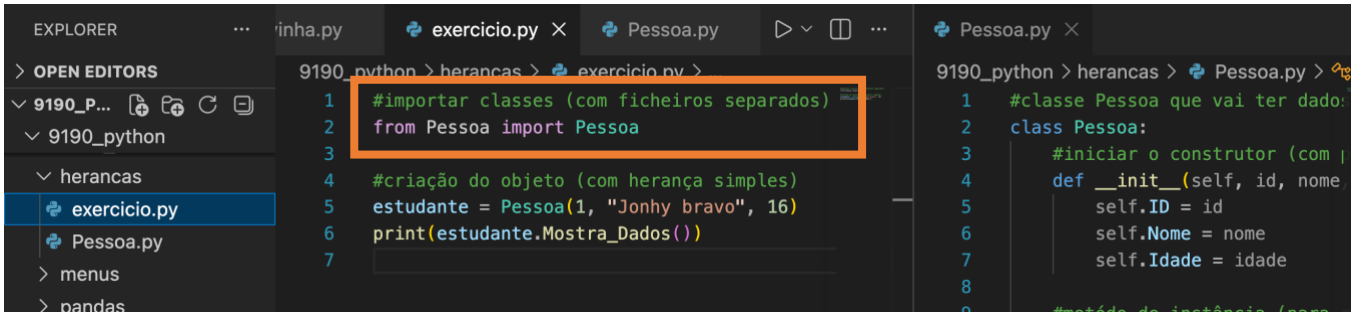
Neste exemplo, a estrutura da classe Pessoa foi separada num ficheiro à parte. Reparem que neste ficheiro apenas temos a construção da classe:



Explicação da funcionalidade da importação de classes com Python

Após isso, foi criado o outro ficheiro com o nome **exercicio.py** e logo no início ficheiro, foi importada a referência para ir buscar os dados do ficheiro da classe Pessoa. Para importar uma classe/ficheiro que esteja separado à parte, devemos fazer o seguinte código:

from nome_ficheiro **import** nome_ficheiro_classe



Relação da importação dos dados

- **From nome_ficheiro** → ir carregar os dados do ficheiro com a classe
- **Import nome_classe** → ir carregar qual a classe a utilizar (pois pode haver mais de 1 classe no ficheiro e pode haver necessidade de carregar mais do que uma classe ao mesmo tempo);

Exercícios

1. Faça a criação da classe com o nome **Pessoa**, do qual o **construtor de inicialização** deve receber as seguintes informações: ID, Nome, Idade, Genero e Cidade.
   - Deve fazer a **declaração do nome da classe**;
   - Fazer o **construtor de inicialização** da classe com os **dados referenciados mais acima**;
   - Fazer o **método** para **mostrar os dados da pessoa**;

No final crie 3 objetos com base na nova classe e mostre na consola.

2. Faça a criação da classe com o nome **Animal**, do qual o **construtor de inicialização** deve receber as seguintes informações: ID, Nome, Idade, Genero, Raça e Porte.
   - Deve fazer a **declaração do nome da classe**;
   - Fazer o **construtor de inicialização** da classe com os **dados referenciados mais acima**;
   - Fazer o **método** para **mostrar os dados do animal**;

No final crie 3 objetos com base na nova classe e mostre na consola.

3. Faça a criação da classe com o nome **Produto**, do qual o **construtor de inicialização** deve receber as seguintes informações: ID, Descrição e Preço.
   - Deve fazer a **declaração do nome da classe**;
   - Fazer o **construtor de inicialização** da classe com os **dados referenciados mais acima**;
   - Fazer o **método** para **mostrar os dados do produto**;

No final crie 3 objetos com base na nova classe e mostre na consola.

Exercícios Adicionais

4. Faça a criação da classe com o nome **Cafe**, do qual o **construtor com parâmetros** deve receber as seguintes informações: ID, Nome_Cafe, Origem, Tipo_Cafe (arábica, rustica, etc...), Preco e Quantidade;
   - Deve fazer a **declaração do nome da classe**;
   - Fazer o **construtor com parâmetros** da classe com os **dados referenciados mais acima**;

- Fazer o **método** para **mostrar os dados do café**;

No final crie 3 objetos com base na nova classe e mostre na consola.

5. Faça a criação da classe com o nome **Sobremesas**, do qual o **construtor de inicialização** deve receber as seguintes informações: ID, Nome e Calorias.

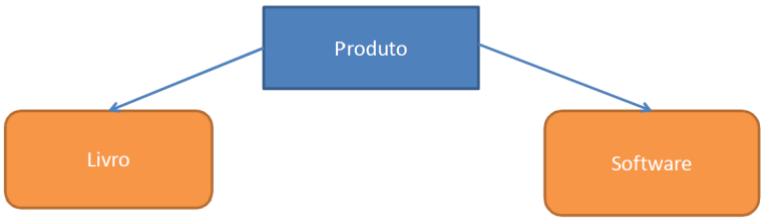
- Deve fazer a **declaração do nome da classe**;
- Fazer o **construtor de inicialização** da classe com os **dados referenciados mais acima**;
- Fazer o **método** para **mostrar os dados do animal**;

No final crie 3 objetos com base na nova classe e mostre na consola.

Heranças com classes em Python

A herança é um dos pilares fundamentais da programação orientada a objetos (POO), do qual, permite com que uma subclasse (ou chamada de classe filha) derive atributos e métodos de uma superclasse (chamada de classe pai).

Esse recurso é essencial para a reutilização do código e simplifica a manutenção, facilitando a criação de programas eficientes.



No exemplo anterior, existe uma superclasse "**Produto**" que vai servir de base à construção de diferentes classes de produtos;

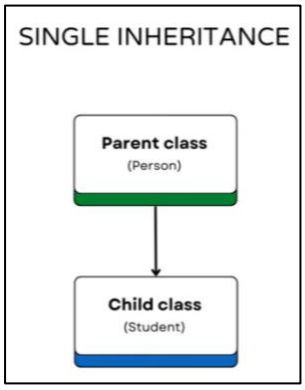
- As classes **Livro** e **Software** têm como ponto de partida a classe "**Produto**".
- As subclasses **Livro** e **Software** **herdam** todos os métodos públicos da classe "**Produto**";
- A subclasse **Livro** e **Software** definem **métodos específicos de cada classe**, que são únicos para **cada classe**.

Imaginem que está a criar uma aplicação com várias funções de utilizador, como por exemplo, alunos, professores e administradores.

Essas funções compartilham atributos comuns, como o nome e ID, mas também exigem funcionalidades que são exclusivas para cada classe.

Em vez de duplicar o código, a herança permite que defina o comportamento compartilhado em uma classe principal e o estenda em classes secundárias especializadas.

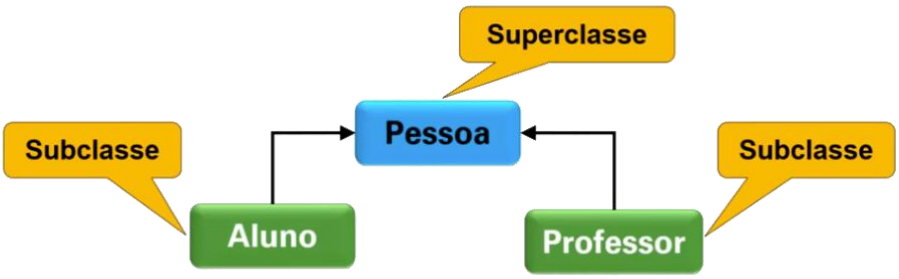
Herança Simples



- Ocorre quando uma classe herda os atributos e métodos de ***somente uma outra classe***.
- Isto permite criar hierarquias de classes de maneira organizada e lógica.
- Em Python, a herança simples é facilmente implementada utilizando parênteses () para ***especificar a superclasse***.

Vamos considerar o seguinte exemplo: Temos uma ***superclasse*** com o nome Pessoa e as ***subclasses*** Aluno e Professor

Esta superclasse vai ter atributos e métodos que depois irá passar para subclasses que herdam desta classe.



Superclasse Pessoa

- Como tal, vamos construir a ***classe Pessoa e criar o construtor*** da classe. No construtor, vamos indicar as **variáveis de classe**: ID, nome e Idade

```
#classe Pessoa que vai ter dados: ID, Nome e Idade
class Pessoa:
    #iniciar o construtor (com parametros de entrada)
    def __init__(self, id, nome, idade):
        self.ID = id
        self.Nome = nome
        self.Idade = idade
```

- Para além dessa informação, vamos criar um método da classe para retornar os dados das variáveis da classe com o nome Mostra_Dados:

```python
#metódo de instância (para retornar os dados)
def Mostra_Dados(self):
    return f"ID: {self.ID}, Nome: {self.Nome}, Idade: {self.Idade}"
```

Superclasse Pessoa (completo)

```python
#classe Pessoa que vai ter dados: ID, Nome e Idade
class Pessoa:
    #iniciar o construtor (com parametros de entrada)
    def __init__(self, id, nome, idade):
        self.ID = id
        self.Nome = nome
        self.Idade = idade

    #metódo de instância (para retornar os dados)
    def Mostra_Dados(self):
        return f"ID: {self.ID}, Nome: {self.Nome}, Idade: {self.Idade}"
```

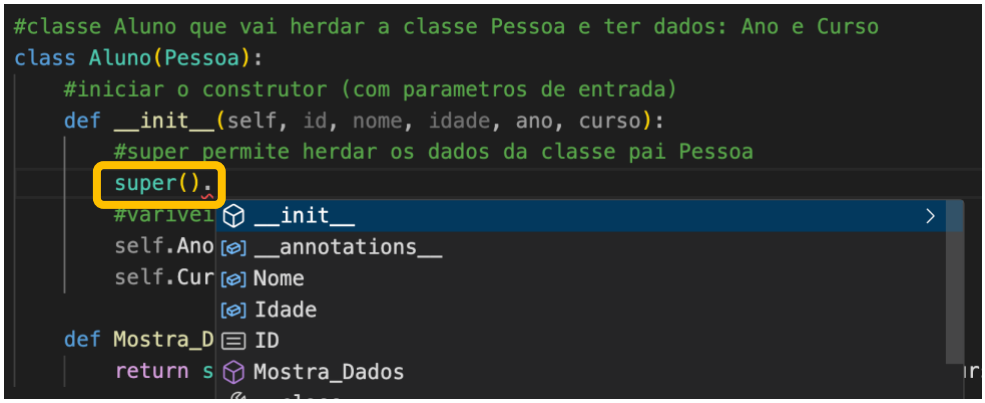
- Repare que construir a superclasse é a mesma coisa que construir uma classe normal (e não houve a preocupação de indicar alguma informação adicional);
- Quando criar as outras classes e se desejar fazer a herança, só aí é que vai ter de ter de indicar a herança;

Subclasse Aluno

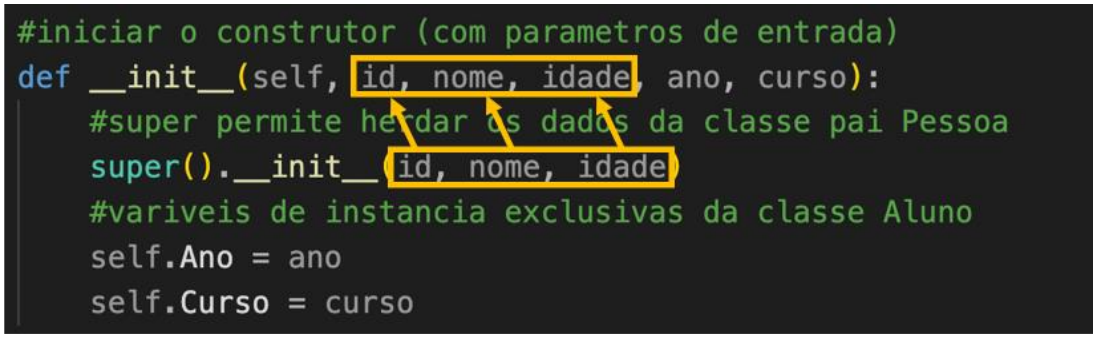
- Vamos começar por implementar a subclasse que irá herdar dados da superclasse **Pessoa**.
- Como tal, vamos começar por construir o início da classe e para dizer que esta classe vai herdar outra classe, logo a seguir ao nome da classe, vamos colocar entre parênteses o nome da superclasse (neste caso a classe Pessoa);
- De seguida, vamos fazer o construtor da classe e, como tal, o construtor vai ter de indicar as variáveis que vão ser herdadas da superclasse e depois disso é que refere às variáveis especificas da nova classe:
    - **Variáveis herdadas da superclasse Produto:** Id, Nome e Idade
    - **Variáveis exclusivas da subclasse Aluno:** ano e curso

```python
#classe Aluno que vai herdar a classe Pessoa e ter dados: Ano e Curso
class Aluno(Pessoa):
    #iniciar o construtor (com parametros de entrada)
    def __init__(self, id, nome, idade, ano, curso):
```

- Dentro do construtor, a primeira linha de código vai servir para chamar a herança da superclasse Pessoa e herdar três variáveis: ID, nome e idade.
- Para tal, vamos usar a palavra reservada **super** e logo a seguir colocamos **o ponto final** para ***aceder*** às propriedades e métodos da superclasse
- A função ***super()*** do Python é usada para ***dar acesso a métodos de uma classe herdada***.



- O **super()** é utilizado entre heranças de classes e vai disponibilizar métodos de uma **superclasse** (classe pai) para uma **subclasse** (classe filha).
- Através do **super()** definimos um novo comportamento para um determinado método construído na classe pai que vai ser **herdado pela classe filha**.



- No exemplo anterior, temos uma **subclasse Aluno** que **herda** os dados da **superclasse Pessoa** e utiliza o **super()** para chamar o método construtor da **superclasse** no seu método construtor.
- Logo a seguir ao **super()**, declaramos as **variáveis especificas da subclasse Aluno** que são o ano e o curso.
- **Após a criação do construtor** (com **herança da superclasse**), vamos criar um método da classe para mostrar os dados, do qual, vamos invocar o método do **super().MostraDados()** para ir buscar a mensagem a superclasse e juntar com os novos dados da subclasse:

```python
#Metodo de instancia com herança do Mostra_Dados da superclasse Produto
def Mostra_Dados(self):
    return super().Mostra_Dados() + f", Ano: {self.Ano}, Curso: {self.Curso}"
```

Subclasse Aluno (completa)

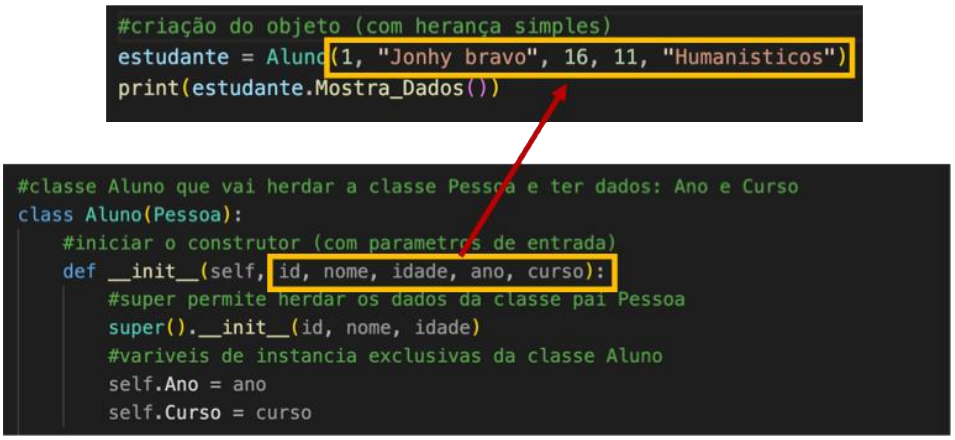
```python
#classe Aluno que vai herdar a classe Pessoa e ter dados: Ano e Curso
class Aluno(Pessoa):
    #iniciar o construtor (com parametros de entrada)
    def __init__(self, id, nome, idade, ano, curso):
        #super permite herdar os dados da classe pai Pessoa
        super().__init__(id, nome, idade)
        #variveis de instancia exclusivas da classe Aluno
        self.Ano = ano
        self.Curso = curso

    #Metodo de instancia com herança do Mostra_Dados da superclasse Produto
    def Mostra_Dados(self):
        return super().Mostra_Dados() + f", Ano: {self.Ano}, Curso: {self.Curso}"
```

Testar a herança simples Pessoa → Aluno

Para instanciar e construir um novo objeto vamos usar a subclasse Aluno (uma vez que está vai herdar os dados da superclasse Pessoa)

```python
#criação do objeto (com herança simples)
estudante = Aluno(1, "Jonhy bravo", 16, 11, "Humanisticos")
print(estudante.Mostra_Dados())
```

O número de parâmetros deve coincidir a entrada de dados no construtor da subclasse:



Resultado final:

```
#criação do objeto (com herança simples)
estudante = Aluno(1, "Jonhy bravo", 16, 11, "Humanisticos")
print(estudante.Mostra_Dados())
```

```
ID: 1, Nome: Jonhy bravo, Idade: 16, Ano: 11, Curso: Humanisticos
```

Exercício 1

1. Faça a criação da classe com o nome **Produto**, do qual o **construtor com parâmetros** deve receber as seguintes informações: ID, Descrição e Preço.
   - Deve fazer a **declaração do nome da classe**;
   - Fazer o **construtor com parâmetros** da classe com os **dados referenciados mais acima**;
   - Fazer o **método** para **mostrar os dados do produto**;

No final crie 3 objetos com base na nova classe e mostre na consola.

2. Faça a criação da classe com o nome **Livro**, do qual o **construtor com parâmetros** deve receber as seguintes informações: Autor e Numero de Páginas.
   - Deve fazer a **declaração do nome da classe** e referir que vai **herdar dados** da superclasse **Produto**;
   - Fazer o **construtor com parâmetros** da classe com os **dados referenciados mais acima**. Não esquecer de utilizar a palavra reservada super() para chamar a herança da superclasse Produto e herdar as variáveis do produto;
   - Fazer o **método** para **mostrar os dados do produto**, do qual, vamos **invocar o método da superclasse Produto para mostrar a mensagem das variáveis do produto** e **juntar** com os novos dados da subclasse;

No final crie 3 objetos com base na nova classe e mostre na consola.

3. Faça a criação da classe com o nome **Software**, do qual o **construtor com parâmetros** deve receber as seguintes informações: Empresa e Versão.
   - Deve fazer a **declaração do nome da classe** e referir que vai **herdar dados** da superclasse **Produto**;
   - Fazer o **construtor com parâmetros** da classe com os **dados referenciados mais acima.** Não esquecer de utilizar a palavra reservada super() para chamar a herança da superclasse Produto e herdar as variáveis do produto;
   - Fazer o **método** para **mostrar os dados do produto**, do qual, vamos **invocar o método da superclasse Produto para mostrar a mensagem das variáveis do produto** e **juntar** com os novos dados da subclasse;

No final crie 3 objetos com base na nova classe e mostre na consola.

Exercício 2

1. Faça a criação da classe com o nome **Analises**, do qual o **construtor com parâmetros** deve receber as seguintes informações: ID, Nome_Utente, Data.
   - Deve fazer a **declaração do nome da classe**;
   - Fazer o **construtor com parâmetros** da classe com os **dados referenciados mais acima**;
   - Fazer o **método** para **mostrar os dados da análise**;

2. Faça a criação da classe com o nome **Hemograma**, do qual o **construtor com parâmetros** deve receber as seguintes informações: **Hemácias** (globos vermelhos), **Leucócitos** (globos brancos) e **Trombócitos** (plaquetas);
   - Deve fazer a **declaração do nome da classe** e referir que vai **herdar dados** da superclasse **Analises**;
   - Fazer o **construtor com parâmetros** da classe com os **dados referenciados mais acima**. Não esquecer de utilizar a palavra reservada super() para chamar a herança da superclasse e herdar as variáveis das análises;
   - Fazer o **método** para **mostrar os dados**, do qual, vamos **invocar o método da superclasse para mostrar a mensagem das variáveis** e **juntar** com os novos dados da subclasse;

No final crie 3 objetos com base na nova classe e mostre na consola.

3. Faça a criação da classe com o nome **Bioquímica**, do qual o **construtor com parâmetros** deve receber as seguintes informações: **Ureia, Creatinina e PH**;

   - Deve fazer a **declaração do nome da classe** e referir que vai **herdar dados** da superclasse **Analises**;
   - Fazer o **construtor com parâmetros** da classe com os **dados referenciados mais acima**. Não esquecer de utilizar a palavra reservada super() para chamar a herança da superclasse e herdar as variáveis das análises;
   - Fazer o **método** para **mostrar os dados**, do qual, vamos **invocar o método da superclasse para mostrar a mensagem das variáveis** e **juntar** com os novos dados da subclasse;

No final crie 3 objetos com base na nova classe e mostre na consola.

Exercícios Adicionais Heranças

1. Faça a criação da classe com o nome **Sobremesas**, do qual o **construtor de inicialização** deve receber as seguintes informações: ID, Nome e Calorias.

   - Deve fazer a **declaração do nome da classe**;
   - Fazer o **construtor de inicialização** da classe com os **dados referenciados mais acima**;
   - Fazer o **método** para **mostrar os dados da sobremesa**;

2. Faça a criação da classe com o nome **Semifrio**, do qual o **construtor de inicialização** deve receber as seguintes informações: Tipo_Semifrio e Temperatura.

   - Deve fazer a **declaração do nome da classe** e referir que vai **herdar dados** da superclasse **Sobremesas**;
   - Fazer o **construtor de inicialização** da classe com os **dados referenciados mais acima.** Não esquecer de utilizar a palavra reservada super() para chamar a herança da superclasse **Sobremesas** e herdar as variáveis do produto;
   - Fazer o **método** para **mostrar os dados do produto**, do qual, vamos **invocar o método da superclasse Produto para mostrar a mensagem das variáveis do produto** e **juntar** com os novos dados da subclasse;

No final crie 3 objetos com base na nova classe e mostre na consola.