**Python Orientado à Objetos**

*“Python já nasceu sendo uma linguagem de programação multi-paradigma, isto é: é possível programar em Python de maneira Imperativa, Funcional e também no paradigma utilizando conceitos da Programação Orientada a Objetos.” (PythonAcademy, 2023)*

A Programação Orientada a Objetos (POO) é um paradigma de programação que se baseia no conceito de classes e objetos. Essa abordagem de programação permite que os desenvolvedores consigam modelar o mundo real em seus programas de computador, representando entidades, atributos e comportamentos como objetos.

As **classes** podem conter dados na forma de campos que podemos chamar de atributos ou propriedades e código na forma de procedimentos, que podemos chamar de métodos. Uma característica fundamental dos **objetos** é que eles têm a capacidade de acessar e muitas vezes modificar seus próprios campos de dados; os objetos mantêm uma referência a si mesmos, frequentemente denotada como o atributo "self" em Python.

Na Programação Orientada a Objetos (POO), os programas são concebidos com base na interação entre objetos. Este paradigma coloca ênfase nos objetos que os desenvolvedores desejam manipular, em vez de focar principalmente na lógica necessária para efetuar essas manipulações. Essa abordagem de programação é particularmente adequada para o desenvolvimento de programas extensos, complexos e que precisam ser constantemente atualizados ou mantidos.

**Bibliotecas, Módulos e Pacotes em Python**

*“a biblioteca padrão Python é um conjunto de módulos disponíveis em Python para que você possa importá-los e usar as funcionalidades deles quando bem quiser.” (Algoritmos em Python)*

Com base na citação acima, concluímos que as bibliotecas em Python são conjuntos de módulos e funções pré-desenvolvidos que aumentam a funcionalidade da linguagem. Elas são essenciais para o desenvolvimento de aplicativos Python, pois oferecem várias funcionalidades prontas para uso.

*“A biblioteca padrão do Python é muito extensa, oferecendo uma ampla gama de recursos” (Python Org)*

Temos dois tipos de bibliotecas, sendo eles:

- Bibliotecas Padrão:

*“Python possui bibliotecas padrão, coleções de módulos e scripts acessíveis que facilitam os processos de criação” (Geo Sem Fronteiras)*

Alguns exemplos de tarefas são: manipulação de arquivos, comunicação em rede, processamento de strings, cálculos matemáticos, manipulação de datas, etc.

Esses módulos estão prontamente disponíveis e não exigem instalação adicional.

-Bibliotecas Externas:

“*A instalação de bibliotecas externas é uma etapa importante para qualquer projeto em Python que necessite de recursos adicionais para executar tarefas específicas.” (Gustavo Soares, Dev, 2023)*

Além da biblioteca padrão, Python possui uma ampla coleção de bibliotecas externas, também chamadas de pacotes ou módulos de terceiros. Essas bibliotecas são desenvolvidas e mantidas pela comunidade Python e abrangem vários domínios, incluindo ciência de dados, aprendizado de máquina, desenvolvimento web, automação, jogos, etc.

Ao falarmos sobre as bibliotecas em python, ouvimos também sobre o “gerenciamento de pacotes”, que é uma parte fundamental do desenvolvimento de software desta linguagem.

*“não é viável que a instalação dessas bibliotecas seja feita de forma manual, já* *que o processo de cada uma delas podem ser, no mínimo, complicadas.” (Fagner Pinheiro, Treina Web)*

O gerenciamento de pacotes facilita a instalação, atualização e distribuição de bibliotecas, módulos e pacotes de terceiros que podem ser usados em seus projetos. O principal sistema de gerenciamento de pacotes em Python é o "pip", que é amplamente utilizado pela comunidade de desenvolvedores Python.

Em relação a documentação, a maioria das bibliotecas possui documentação oficial detalhada que oferece informações sobre como utilizar suas funções e classes. Ela é uma fonte valiosa de informações para programadores que trabalham com essas bibliotecas.

**Exemplos de Bibliotecas Populares:**

-NumPy: Usado para computação numérica e manipulação de arrays multidimensionais.

-Pandas: Ideal para análise e manipulação de dados tabulares.

-Matplotlib e Seaborn: Utilizados para criação de visualizações de dados.

-SciPyGenericName: Oferece recursos avançados para a computação científica no ambiente Python.

-TensorFlow: Disponibiliza recursos para realizar aprendizado de máquina avançado e implementar técnicas de aprendizado profundo.

No exemplo de biblioteca padrão do python, usamos a “random" ela fornece funções para gerar números aleatórios e é comumente usada em muitos programas para introduzir aleatoriedade em processos, como simulações, jogos e experimentos estatísticos.

Texto

Descrição gerada automaticamente

Fonte: Grupo

Em relação aos módulos, concluímos que em Python, um módulo representa um arquivo que contém código incluindo definições de funções, classes e variáveis que podem ser utilizadas em outros programas Python.

“Um módulo nada mais é do que um arquivo .py com instruções e definições em Python” (Algoritmos em Python)

Os módulos possuem um papel importante na organização do código em Python, permitindo a divisão de um programa em partes menores e reutilizáveis.

Para utilizar um módulo em um programa Python, você precisa importá-lo usando a palavra-chave “import”. Por exemplo, se você possui um módulo chamado “meu\_modulo.py”, pode importá-lo da seguinte maneira:

Texto

Descrição gerada automaticamente

Fonte: Pense Python: “Módulos e Obtendo Ajuda”, 2023

Seguindo esse código, teríamos o seguinte resultado:

Forma

Descrição gerada automaticamente com confiança baixa

Fonte: Pense Python: “Módulos e Obtendo Ajuda”, 2023

Python também inclui uma ampla biblioteca de módulos padrão que oferecem funcionalidades prontas para uso, como math, random, os, datetime, entre outros. O uso de módulos ajuda a organizar seu código de maneira mais estruturada e modular.

*“Lembre-se que uma vez que o módulo é importado, podemos utilizar as coisas* *que estão definidas dentro dele.” (Pense Python, 2023)*

Módulos permitem que você reutilize código em diferentes projetos. Depois de definir e testar um módulo, você pode usá-lo em diferentes programas sem a necessidade de reescrever o código.

Quando um diretório contém um arquivo chamado \_\_init\_\_.py, ele é considerado um pacote Python. Isso possibilita a organização de módulos relacionados em pacotes e a importação deles como pacote.módulo.

Em continuidade à esse último parágrafo, agora, falaremos em relação aos pacotes em Python.

Dizemos que é uma maneira de organizar módulos relacionados em diretórios. Isso ajuda a estruturar e gerenciar seu código de maneira mais eficaz quando você tem um grande número de módulos.

*“Ao criarmos um pacote, além de organizarmos nosso programa, temos a possibilidade de importarmos os módulos de forma bastante flexível.” (DevFuria, 2023)*

Um pacote é essencialmente um diretório que contém um arquivo especial chamado “\_\_init\_\_.py” (usaremos esse arquivo ao decorrer do trabalho).

Esse arquivo indica ao Python que o diretório deve ser tratado como um pacote. O arquivo “\_\_init\_\_.py” pode estar vazio ou conter código de inicialização para o pacote.

*“Isso é feito para evitar diretórios com o mesmo nome, de ocultar involuntariamente módulos válidos que ocorrem mais tarde no caminho de pesquisa do módulo.” (Candido, School Of Net, 2018)*

Os pacotes geralmente contêm módulos relacionados e para importa-los você precisa usar a notação de ponto. Podem conter subpacotes, que são diretórios dentro de diretórios.

*“Um pacote pode conter vários outros módulos, chamamos de submódulos.” (DevFuria, 2023)*

Resumindo, pacotes em Python são uma maneira eficaz de organizar e estruturar projetos de código maiores, tornando o código mais legível e fácil de manter, ao mesmo tempo em que facilitam a reutilização de funcionalidades específicas em diferentes partes do seu programa.

**Aplicação de Banco de Dados com Python**

Um banco de dados é um repositório organizado de informações ou dados estruturados, geralmente mantido em formato eletrônico em um sistema de computador.

“é um repositório para armazenar informações (dados) de qualquer natureza. Ele retrata aspectos do mundo real — ou seja, o conceito visto anteriormente de minimundo — em que qualquer mudança que se faça no minimundo é diretamente replicada no banco de dados.” (Fundação Bradesco, Escola Virtual, 2023).

Suas vantagens são: velocidade, produtividade, precisão, redução de riscos, proteção,

Os bancos de dados relacionais desempenham um papel fundamental no ecossistema de desenvolvimento Python, permitindo que aplicativos armazenem e gerenciem dados de forma estruturada e eficaz. Python oferece diversas opções para interagir com bancos de dados relacionais, e aqui estão as principais:

-Biblioteca SQLite da Biblioteca Padrão: Python inclui a biblioteca SQLite, que é um sistema de gerenciamento de banco de dados SQL incorporado. É adequado para aplicativos menores e locais, sendo uma escolha conveniente para prototipagem rápida e para situações em que não é necessário um servidor de banco de dados separado.

**-ORMs (Mapeamento Objeto-Relacional):** Python oferece diversos ORMs populares, como SQLAlchemy, Django ORM e Peewee. Essas ferramentas permitem mapear objetos Python para tabelas em bancos de dados, simplificando o trabalho com dados por meio de uma abordagem mais orientada a objetos, em vez de escrever SQL manualmente.

**-Bibliotecas de Conexão Direta:** Existem bibliotecas de terceiros que possibilitam a conexão direta com bancos de dados relacionais usando SQL nativo. Exemplos incluem psycopg2 para PostgreSQL, mysql-connector-python para MySQL e pyodbc para SQL Server.

**-Frameworks Web:** Alguns frameworks web, como Flask e Django, fornecem suporte integrado para lidar com bancos de dados relacionais. Isso simplifica o desenvolvimento de aplicativos da web que interagem com bancos de dados.

**-Bancos** de Dados NoSQL: Embora não sejam bancos de dados relacionais, as opções NoSQL, como MongoDB e Cassandra, também são populares na comunidade Python. Existem bibliotecas e pacotes que facilitam a integração desses bancos de dados com Python.

No geral, a capacidade do Python de se integrar facilmente com bancos de dados relacionais é uma das razões pelas quais a linguagem é tão amplamente adotada para o desenvolvimento web e a análise de dados. A escolha da abordagem e da biblioteca depende das necessidades particulares de cada projeto e do conhecimento prévio do desenvolvedor.

**Princípios de POO**

*“No paradigma orientado à objetos, uma classe é a representação de algo do mundo real. No Python, o uso de classes é algo constante no desenvolvimento de programas.” (TreinaWeb, 2021)*

**Classes**

São estruturas de dados criadas pelo desenvolvedor que funcionam como um blueprint (plano detalhado ou modelo que descreve como algo deve ser construído, desenvolvido ou executado) para objetos.

Exemplo:

Uma imagem contendo relógio, escuro, bola, medidor

Descrição gerada automaticamente

Fonte: Grupo

Uma classe é composta por atributos e métodos. Os atributos de uma classe refletem as suas características, enquanto os métodos descrevem o comportamento associado à classe.

Para declarar um atributo em uma classe em Python, é necessário especificar o nome do atributo no método especial chamado “\_\_init\_\_”. Este método é responsável por definir o construtor da classe, ou seja, é onde determinamos como criar uma nova instância da classe em nosso programa.

Exemplo:

Interface gráfica do usuário, Site

Descrição gerada automaticamente

Fonte: Grupo

Agora, estamos estabelecendo que toda pessoa criada em nosso programa, com base na classe Pessoa, deve obrigatoriamente possuir um nome.

Agora, vamos definir um método de saudação para a classe “pessoas”. Pra isso, usamos a palavra “def”, isso indica que estamos definindo um método para a classe, incluindo o nome do método e, se houver, seus atributos.

Exemplo:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança média

Fonte: Grupo

**Objetos**

Os objetos são instâncias de classes. Eles são criados com base no modelo fornecido pela classe e podem armazenar dados e realizar ações. Por exemplo, um objeto específico pode ser um carro com uma marca e um modelo específicos.

*“Sempre que precisamos criar algo com base em uma classe, dizemos que estamos instanciando objetos.” (TreinaWeb, 2021)*

Essa citação acima relata que estamos gerando a representação de uma classe dentro do nosso programa.

Para criar uma instância de um objeto em Python, com base em uma classe que foi previamente definida, é preciso especificar a classe que queremos usar como modelo e, se necessário, fornecer os valores para seus atributos.

Exemplo:

Texto

Descrição gerada automaticamente

Fonte: Grupo

Quando executamos a linha “pessoa1 = Pessoa("Thiago")”, estamos gerando uma instância de um objeto do tipo Pessoa, fazendo receber o nome “Thiago”.

Por fim, criamos a linha que faz a execução desse processo todo.

Exemplo:

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Fonte: Grupo

Agora é só executar o programa! Abaixo temos a imagem completa dos exemplos acima.

Texto

Descrição gerada automaticamente

Fonte: Grupo

**Encapsulamento**

Esse princípio envolve a ideia de que os detalhes internos de uma classe devem ser ocultados do mundo exterior. Todos os atributos e comportamentos dessa entidade são agregados à sua classe, geralmente são definidos como privados e só podem ser acessados por métodos públicos dessa mesma classe.

*“O princípio do Encapsulamento também afirma que informações importantes devem ser contidas dentro do objeto de maneira privada e apenas informações selecionadas devem ser expostas publicamente.” (Python Academy, 2023)*

Essa característica de ocultação de dados oferece maior segurança ao programa e previne alterações não autorizadas nos dados.

-**Private (Privado):** Os elementos de uma classe (métodos ou variáveis) marcados como privados não estão disponíveis para acesso fora da classe em que estão definidos. Eles são acessíveis apenas internamente na classe.

-**Protected (Protegido):** Os membros marcados como protegidos podem ser acessados tanto dentro da própria classe quanto por classes derivadas (subclasses) da classe que os engloba.

-**Public (Público):** Os membros públicos podem ser acessados de qualquer lugar, tanto dentro como fora da classe. Eles definem a interface pública da classe.

Exemplo de Encapsulamento Simples:

1. Criamos a classe “Lista de Tarefas” com uma variável privada para armazenar as tarefas e o método construtor “\_\_init\_\_” que vimos anteriormente.

Texto

Descrição gerada automaticamente com confiança média

Fonte: Grupo

Aqui, estamos definindo a variável “\_\_tarefas” como uma lista vazia dentro do construtor da classe.

1. Adicione o método “adicionar\_tarefa” para permitir adicionar tarefas à lista. Também usamos o método “append” que é um recurso em Python que permite incluir um novo elemento no final de uma lista já existente.

*“Essa função é extremamente útil quando você precisa expandir uma lista com mais itens dinamicamente, sem precisar criar uma nova lista a cada vez.” (Awari, 2023.)*

Texto

Descrição gerada automaticamente

Fonte: Grupo

Esse método recebe uma tarefa como argumento e a adiciona à lista de tarefas.

1. Adicionamos o método “listar\_tarefas” para listar todas as tarefas na lista. Usamos o método “enumerate” nessa etapa.

*“A função "enumerate()" é útil quando precisamos acessar tanto o valor quanto o índice de cada elemento da lista.” (Emmanuel Adrade, Dio, 2023)*

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Fonte: Grupo

Aqui já temos uma classe base com encapsulamento que pode adicionar e listar tarefas.  
Utilizamos a estrutura “for” que executa o(s) código(s) contidos nela repetidamente, e “in” que é utilizada para determinar se um valor está incluso ou não em um conjunto de valores.

**Herança**

*“A Herança é um conceito do paradigma da orientação à objetos que determina que uma classe (filha) pode herdar atributos e métodos de uma outra classe (pai) e, assim, evitar que haja muita repetição de código.” (TreinaWeb, 2023)*

De acordo com a citação acima, a “classe filha” é chamada de:

-classe derivada

-subclasse

Enquanto a “classe pai” é chamada de:

-classe base

-superclasse

Isso facilita a reutilização de código e a criação de hierarquias de classes.

1. Aplicando a Herança, vamos criar uma classe ListaPrioritaria que herda da ListaDeTarefas. A herança é estabelecida pela colocação da classe pai entre parênteses. Utilizamos a função “super()” que é usada para chamar métodos da classe pai (superclasse) em uma hierarquia de herança.

Texto

Descrição gerada automaticamente

É importante ressaltar que estamos sobrescrevendo o método “listar\_tarefas” para incluir informações de prioridade.

**Polimorfismo**

*“é a capacidade que uma subclasse tem de ter métodos com o mesmo nome de sua superclasse, e o programa saber qual método deve ser invocado, especificamente (da super ou sub).” (Samuel O. N. M. Filho, Forum Alura, 2022)*

Isso significa que várias classes podem ter métodos com o mesmo nome, mas com comportamentos diferentes, o que facilita a criação de sistemas flexíveis e extensíveis.

Aplicando o Polimorfismo, criaremos uma segunda subclasse chamada “ListaComData” que também herda de “ListaDeTarefas”. Esta classe adiciona a capacidade de atribuir datas de vencimento às tarefas.

Texto

Descrição gerada automaticamente

Fonte: Grupo

Nessa etapa, usamos função “zip()” que é usada para juntar duas ou mais listas em uma única lista de tuplas, onde cada tupla contém um elemento de cada uma das listas fornecidas.

Novamente, estamos sobrescrevendo o método “listar\_tarefas”, mas agora para incluir informações sobre a data de vencimento.

1. Até aqui, já temos as três classes. Agora criaremos as instâncias e aplicar o polimorfismo. Criamos instâncias de cada classe e usando os métodos para adicionar e listar tarefas. Apesar das instâncias terem comportamentos diferentes, todas estão sendo tratadas do mesmo jeito, aqui encontramos o polimorfismo.

Texto

Descrição gerada automaticamente

Fonte: Grupo

Segue abaixo o código completo:

Texto

Descrição gerada automaticamente

Fonte: Grupo

Texto

Descrição gerada automaticamente

Fonte: Grupo

A POO é amplamente utilizada em muitas linguagens de programação, como Java, C++, Python, C#, e muitas outras, devido à sua capacidade de organizar o código de forma mais modular, facilitando a manutenção e o desenvolvimento de software mais escalável e reutilizável.

Por fim, usando um pouco do que vimos anteriormente, temos mais um exemplo de como usar a POO.

Tela de computador com letras e números em fundo preto

Descrição gerada automaticamente

Fonte: Grupo

Neste exemplo, criamos a classe e o construtor, definimos o método e criamos a instância da classe e por fim chamamos o método para impressão da imagem.

**Referências**

ANDRADE, Ana Paula de. “Orientação a Objetos em Python”; Treina Web, 2021

Acesso em: 27/10/2023

Disponível em:

<https://www.treinaweb.com.br/blog/orientacao-a-objetos-em-python>

RAMOS, Vinícius. “Programação Orientada a Objetos no Python: Introdução”; Python Acadamy, 2023

<https://pythonacademy.com.br/blog/introducao-a-programacao-orientada-a-objetos-no-python>

ANDRADE, Emmanuel, “3 Formas de Iterar uma Lista em Python”; Dio, 2023.

<https://www.dio.me/articles/3-formas-de-iterar-uma-lista-em-python>

AWARI; “Explorando o Método Append em Python: Dicas e Truques”; 2023

<https://awari.com.br/metodo-append-em-python/#:~:text=com%20a%20Awari-,O%20que%20%C3%A9%20M%C3%A9todo%20Append%20em%20Python,nova%20lista%20a%20cada%20vez>.

PINHEIRO, Fagner; “Utilizando Herança no Python”; Treina Web, 2023

<https://www.treinaweb.com.br/blog/utilizando-heranca-no-python>

ALURA; “Herança e Polimorfismo”, 2023

<https://www.alura.com.br/apostila-python-orientacao-a-objetos/heranca-e-classes-abstratas?utm_term=&utm_campaign=%5BSearch%5D+%5BPerformance%5D+-+Dynamic+Search+Ads&utm_source=adwords&utm_medium=ppc&hsa_acc=7964138385&hsa_cam=1560195067&hsa_grp=63243218150&hsa_ad=401937480958&hsa_src=g&hsa_tgt=aud-409949667324:dsa-758241041973&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gclid=Cj0KCQjwvL-oBhCxARIsAHkOiu0gLsB5QACb7-flAn7Sw2EVUCqlAKEnE40ktoTd10f5xGt6psRhGZIaAkCGEALw_wcB>

Pytho Org; “A Biblioteca Padrão do Python”;

<https://docs.python.org/pt-br/3/library/>

FUNDAÇÃO BRADESCO; “Modelagem de dados”; Escola Virtual, 2023

ORG, Python; “10. Um breve passeio pela biblioteca padrão”

[**https://docs.python.org/pt-br/3/tutorial/stdlib.html**](https://docs.python.org/pt-br/3/tutorial/stdlib.html)

SOARES, Gustavo; “Python: Imports e Bibliotecas”; DevFuria, 2023

[**https://dev.to/iugstav/python-imports-e-bibliotecas-42c0**](https://dev.to/iugstav/python-imports-e-bibliotecas-42c0)

PINHEIRO, Fagner; “Gerenciando Pacotes em Projetos Python com o PIP”; TreinaWeb, 2018

<https://www.treinaweb.com.br/blog/gerenciando-pacotes-em-projetos-python-com-o-pip>

ORG, Python; “A Biblioteca Padrão do Python”

<https://docs.python.org/pt-br/3/library/>

**Pense Python; “Modulos e Obtendo Ajuda”; Panda, USP**

<https://panda.ime.usp.br/pensepy/static/pensepy/04-Modulos/modulos.html#:~:text=Um%20m%C3%B3dulo%20%C3%A9%20um%20arquivo,deles%20intensamente%2C%20o%20m%C3%B3dulo%20turtle%20>.

Geo Sem Fronteiras; “5 Bibliotecas Python Para Conhecer”

[**https://geosemfronteiras.org/blog/5-bibliotecas-python-para-conhecer/**](https://geosemfronteiras.org/blog/5-bibliotecas-python-para-conhecer/)

Algoritmos Em Python; “Módulos e Bibliotecas”

<https://algoritmosempython.com.br/cursos/programacao-python/modulos-bibliotecas/#:~:text=Portanto%2C%20a%20biblioteca%20padr%C3%A3o%20Python,funcionalidades%20deles%20quando%20bem%20quiser>

CANDIDO; “Função \_\_init\_\_py”, School Of Net

<https://schoolofnet.com/forum/topico/qual-a-necessidadefuncao-do-__init__py-4308>