Python

Intermediário

I

FUNÇÕES

Módulo Python (Intermediário)

Função

- Uma função é uma sequência de comandos que desempenham algum tipo de tarefa no seu programa
- Evita duplicação de código
- Facilita alterações: se quisermos mudar como a tarefa deve ser executada, teremos apenas um local para modificar

© Paradigma Internet LTDA

Função (Exemplo)

```
def imprime_uma_mensagem():
    print("Olá mundo!")
```

def => indica o início da definição de uma função

imprime_uma_mensagem => nome da função

() => a definição de parâmetros é feita entre os parênteses

: (dois pontos) => indica o início do corpo da função, tudo que estiver identado faz parte desse corpo

Para chamar a função:

```
imprime_uma_mensagem()
```

© Paradigma Internet LTDA

Função (Exemplo)

```
def imprime_uma_mensagem(msg):
    print(msg)
```

© Paradigma Internet LTDA

Parâmetros padrão (Exemplo)

```
def cumprimentar(trat, nome, sobrenome, formal=True):
    if formal:
        return "Olá, %s %s!" % (trat, sobrenome)
    return "Olá, %s!" % nome
```

```
print(cumprimentar("Sr", "Jose", "Silva"))
print(cumprimentar("Sr", "Jose", "Silva", False))
```

© Paradigma Internet LTDA

Funções (Exercícios)

1. Defina uma função chamada maior() que receba dois números como argumento e retorne o maior deles. (Não vale usar a função max())
2. Defina a função maior_de_tres() que receba três números e retorne o maior deles
3. Defina uma função chamada largura() que retorne o tamanho de uma string ou lista. (Não vale usar a função len())

© Paradigma Internet LTDA

Funções (Exercícios)

4. Define uma função chamada eh_vogal(), que receba um caractere (string de tamanho 1) e retorne True se for uma vogal e False caso contrário.

5. Defina as funções soma() e multiplica() que recebam uma lista e retornem o somatório e produtório dessa lista, respectivamente. Exemplo:

   soma([1,2,3,4]) retorna 10

   multiplica([1, 2, 3, 4]) retorna 24

© Paradigma Internet LTDA

II

Classes

Módulo Python (Intermediário)

Classe

- É uma forma de agrupar dados e funções que trabalham sobre esses dados
- Define um um novo tipo de dado, como string, inteiro, lista, etc.
- Os valores dos dados que salvamos em uma classe são chamados atributos, e as funções associadas à classe são chamadas de métodos

© Paradigma Internet LTDA

Classe

- Em Python, tudo é objeto. Ou seja, tudo é uma instância de alguma classe
- Para saber o tipo (ou classe) de determinado objeto, basta:

  type(objeto)

- Exercício: verifique o tipo de alguns objetos (exemplo: strings, inteiros, funções, etc)

© Paradigma Internet LTDA

Classes (Exemplo)

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade
    def ano_nascimento(self):
        return 2015 - self.idade
```

© Paradigma Internet LTDA

Classes (Exercícios)

1. Defina uma classe chamada Animal com as seguintes características:
   - Atributos: especie (string), nome (string), idade (inteiro)
   - Métodos:
     - __init__(self, especie, nome, idade)
     - se_apresentar(self):
       - print("Eu sou um(a) %s, meu nome é %s e tenho %s anos de idade" % (self.especie, self.nome, str(self.idade)))
     - ano_nascimento(self):
       - Retorna o ano em que o animal nasceu

© Paradigma Internet LTDA

Classes (Exercícios)

2. Defina uma classe chamada Dono com as seguintes características:
   - Atributos: nome (string), pets (lista de Animais)
   - Métodos:
     - __init__(self, especie, nome, idade)
     - adicionar_pet(self, animal):
       - adiciona um Animal à lista de pets do Dono.

© Paradigma Internet LTDA

Classe (Exercícios)

- Vamos melhorar nossa classe pessoa:

```
import datetime # objetos de data
...
        def __init__(self, nome, nascimento):
                self.nome = nome
                self.nascimento = nascimento
    def idade(self):
                hoje = datetime.date.today()
                idade = hoje.year - self.nascimento.year
                if hoje < datetime.date(hoje.year, self.nascimento.month,
                self.nascimento.day):
                   idade -= 1
                return idade
```

© Paradigma Internet LTDA

III

Orientação à objetos

Módulo Python (Intermediário)

Introdução

- Não é obrigatório organizar seu código em classes ao programar em Python. Você pode usar apenas funções, e essa abordagem é chamada de procedural.
- O paradigma procedural funciona bem para programas simples.
- Mas para programas maiores e complexos, a Programação Orientada a Objetos (POO) se mostra uma abordagem mais adequada.

© Paradigma Internet LTDA

Vantagens

- Código mais organizado
- Dados e funções (métodos no caso) relacionados são colocados em uma única estrutura (a classe)
- Mais intuitiva, pois já aprendemos a pensar naturalmente sobre objetos e os relacionamentos entre eles
- Mais fácil de codificar usando boas práticas
- Vejamos alguns conceitos de POO e suas implementações em Python...

© Paradigma Internet LTDA

Composição

- É uma forma de relacionamento de objetos onde um objeto é colocado como atributo de outro
- Se pudermos expressar um relacionamento entre duas classes através do termo tem-um (ou tem-uma), então esse é um relacionamento de composição
- Exemplo:
  - Um objeto da classe Pessoa que possui um campo chamado nascimento que armazena um objeto da classe datetime

© Paradigma Internet LTDA

Composição (Exercício)

- Implemente as seguintes classes:

Classe	Atributos	Métodos
Estudante	nome (string) numero (inteiro) disciplinas (lista - Disciplina)	__init__ matricular(self, disciplina)
Departamento	nome (string)	__init__ adicionar_disciplina(self, disciplina)
Disciplina	nome (string) codigo (string) creditos (inteiro) departamento (Departamento)	__init__ adicionar_estudante(self, estudante)

© Paradigma Internet LTDA

Composição (Exercício)

1. Descreva brevemente um possível conjunto de classes que poderiam ser usadas para representar uma coleção de músicas (exemplos de classes: Musica, Artista, Album, Playlist)
2. Escreva uma implementação simples do modelo que mostre como as diferentes classes estariam relacionadas

© Paradigma Internet LTDA

Herança

- É uma forma de arranjar objetos em uma hierarquia
- Um objeto que herda de outro objeto é considerado como um subtipo daquele objeto
- Se pudermos expressar um relacionamento entre duas classes através do termo é-um (ou é-uma), então esse é um relacionamento de herança

```
class Animal:

...

class Cachorro(Animal):

...
```

© Paradigma Internet LTDA

Herança

- Herança nos ajuda a representar objetos que possuem semelhanças na maneira como funcionam
- De forma que podemos implementar funcionalidades comuns em uma classe pai (ou base) e reaproveitar nas classes filhas (ou subclasses)

© Paradigma Internet LTDA

Herança (Exemplo)

```
class Animal:
    def __init__(self, nome, peso):
        self.nome = nome
        self.peso = peso
class Cachorro(Animal):
    def latir(self):
        print('%s:  Au, au, au!' % self.nome)
class Gato(Animal):
    def miar(self):
        print('%s: Miau Miau!' % self.nome)
```

```
um_cao = Cachorro('Bob', 12)

um_cao.latir()
```

```
um_gato = Gato('Nik', 3)

um_gato.miar()
```

© Paradigma Internet LTDA

Herança (Exercício)

- Modele e implemente utilizando Herança as seguintes classes:
  - Pessoa
  - Aluno
  - Professor
- Para testar, crie atributos e métodos adequados a cada tipo de classe

© Paradigma Internet LTDA

III

Erros e Exceções

Módulo Python (Intermediário)

Erros

- Erros em um programa são frequentemente chamados de *bugs*
- Quase sempre são causados uma falha do programador
- O processo de encontrar e eliminar *bugs* é chamado de *debugging*
- Erros podem ser categorizados em 3 grupos:
  - Erros de sintaxe
  - Erros em tempo de execução
  - Erros de lógica

© Paradigma Internet LTDA

Erros de sintaxe

- Erros na utilização da linguagem de programação
- São análogos aos erros ortográficos e gramaticais da lingua portuguesa
- Erros de sintaxe comuns no Python:
  - Escrever uma palavra reservada de forma errada (exemplo: clas)
  - Esquecer de colocar um símbolo, como vírgula, ponto, parênteses
  - Identação incorreta

© Paradigma Internet LTDA

Erros de sintaxe (Exercício)

- Provoque alguns erros de sintaxe e observe como o interpretador se comporta:

```
myfunction(x, y):
    return x + y
```

```
else:
    print("Hello!")
```

```
if mark >= 50
    print("You passed!")
```

```
if arriving:
    print("Hi!")
esle:
    print("Bye!")
```

```
if flag:
print("Flag is set!")
```

© Paradigma Internet LTDA

V

DESAFIOS

Módulo Python (Intermediário)

Funções

1. Defina uma função chamada reverte() que reverte uma string. Exemplo:

   reverte('Teste') retorna 'etseT'

2. Defina uma função chamada eh_palindromo() que reconhece palíndromos (palavras ou frases iguais ao seu reverso, exemplo: radar). Exemplo:

   eh_palindromo('sopapos') retorna True

© Paradigma Internet LTDA

Funções

3. Escreva uma função chamada eh_membro() que recebe um valor (string ou número) e uma lista de valores e verifica se o valor existe na lista. (Não vale usar o operador in). Exemplo:

eh_membro(2, [1,2,5]) retorna True

© Paradigma Internet LTDA

Funções

4. Defina uma função chamada intersecao() que recebe duas listas e retorna os elementos que fazem parte das duas listas. Exemplo:

   intersecao([1,2,3], [1,3,5]) retorna [1,3]

5. Defina uma função chamada histograma() que recebe uma lista de inteiros e imprime um histograma no terminal. Exemplo:

   histograma(3, 6, 4) imprime:

```
* * *

* * * * * *

* * * *
```

© Paradigma Internet LTDA

Classes

1. Defina uma classe chamada Triangulo com as seguintes características:
    - Atributos: angulo1, angulo2 e angulo3 (todos números)
    - Métodos:
        - __init__(self, a1, a2, a3)
        - verificar_angulos(self):
            - Verifica que a soma dos três triângulos dá 180
            - Retorne **True** se a soma der 180, e **False** caso contrário.

© Paradigma Internet LTDA

Classes

2. Defina uma classe chamada Musica com as seguintes características:
    - Atributos: letra (lista de versos)
    - Métodos:
        - __init__(self, letra)
        - cantar(self):
            - Imprime a música, um verso por linha
- Exemplo de chamada:
    - feliz_aniversario = Musica([‘Parabéns pra você’, ‘Nesta data querida’, ‘Muitas felicidades’, ‘Muitos anos de vida!’])
    - feliz_aniversario.cantar()

© Paradigma Internet LTDA

Classes

3. Defina uma classe chamada Utilidades e coloque, como métodos, as funções criadas nos desafios de 1 a 5 (Funções):
    - reverte
    - eh_palindromo
    - eh_membro
    - intersecao
    - histograma
- Testar cada uma delas

© Paradigma Internet LTDA