devaria

Programação – Python

Parte 1

Parte 1

- Python - Introdução
- Executável

Python

https://docs.python.org/pt-br/3/

É uma linguagem de alto nível, interpretada e multiparadigma, suporta o paradigma orientado a objetos, imperativo, funcional e procedural.

Foi lançada por Guido van Rossum em 1991.

É uma linguagem de programação que tem uma curva de aprendizado menor comparado a outras linguagens como Java, C#, etc...

Python

https://docs.python.org/pt-br/3/

- É uma linguagem de tipagem dinâmica forte. Possui uma sintaxe simples e com uma boa legibilidade.

- Possui uma comunidade muito ativa e acolhedora.

Python

https://docs.python.org/pt-br/3/

Dentre os principais pontos fortes do Python temos:

- Considerada pelo público a 3ª linguagem "mais amada“ em 2018. (Stack Overflow)
- Está entre as 5 linguagens mais populares. (RedMonk)
- Multiparadigma;
- Expressões Lambdas;
- Comunidade;
- Sintaxe simples;
- Open source.

Ascensão do Python

https://docs.python.org/pt-br/3/

- BI – Business Inteligence
  - Pandas
  - Numpy
  - Matplotlib

- Inteligência Artificial
  - Scikit Learn
  - Tensor Flow
  - Keras

Como o computador entende o Python?

O interpretador “traduz” o nosso código-fonte do arquivos .py em um formato conhecido como byte code, que logo em seguida irá enviar para a PVM (Python Virtual Machine) que é o motor de execução do Python.

A PVM irá executar esses byte codes e enviar para o sistema operacional em uma linguagem de máquina para processar as informações.

Café ou Dúvidas?

Para que a aula não fique cansativa, a cada conclusão de assunto teremos 15 minutos para que vocês possam tirar as dúvidas através dos comentários no vídeo e responderemos ao vivo.

Parte 2

- Primeiro programa
- Variáveis e inputs da linha de comando

Olá, mundo!

Como nosso jargão, não poderia faltar o famoso Hello World nosso. Nesse exercício vamos exercitar algumas coisas:

- Criar um novo repositório;
- Aprender os menus principais do PyCharm;
- Criar nosso primeiro programa;
- Executá-lo;
- Obter a resposta na linha de comando.

Variáveis

Já estamos falando de variáveis e constantes desde a lógica de programação, mas agora elas farão total sentido. Afinal, a partir de agora, elas receberão dados e serão manipuladas para que, ao fim do programa, o objetivo pelo qual ele foi criado seja alcançado.

Tipos de Dados

Dentre os tipos de valores podemos mencionar:

- str;
- list;
- tuple;
- dict;
- int;
- float;
- complex;
- bool;
- Qualquer classe definida no seu projeto ou proveniente de alguma biblioteca/framework importado para o projeto.

Variáveis

Agora que já sabemos os tipos, como seria a padronização de nome de variáveis e constantes no Python?

```
# Somente declaração de variáveis:
temperatura = 23
nome = “Filipe”
```

Variáveis dinâmicas

As variáveis no Python mudam de tipo dinamicamente


```
n = 20
print(n) # Irá printar na tela 20
n = “Teste”
print(n) # Irá printar na tela Teste
```

Inputs do console

Vamos agora ver como podemos capturar informações no momento em que executamos nosso programa, e utilizar essas informações dentro do programa em si.


```
if __name__ == '__main__' :

    # Recebe o valor digitado e salva em uma variável
    numero = input('Digite um número:')

    # Printa na tela o valor digitado
    print(f'O número digitado foi {numero}')
```

Variáveis e Inputs

Agora que já entendemos como criar nossos programas e exibir informações no console, vamos aprender como receber informações quando nosso programa é executado. Nesse exercício, vamos exercitar algumas coisas:

- Criar um novo programa;
- Executá-lo;
- Capturar as informações passadas na linha de comando;
- Exibir as variáveis criadas na saída do programa no nosso console.

Não se esqueça de commitar
os programas desenvolvidos no seu Git =D

Café ou Dúvidas?

Para que a aula não fique cansativa, a cada conclusão de assunto teremos 15 minutos para que vocês possam tirar as dúvidas através dos comentários no vídeo e responderemos ao vivo.

Parte 3

- Operadores lógicos
- Estrutura de seleção

Operadores lógicos

Seguindo no conhecimento do Python, agora vamos falar sobre os operadores lógicos. Estes são responsáveis por operações lógicas com operandos booleanos:

- Operador unário not (negação lógica);
- Operadores Binários and e or.

Operadores lógicos

O primeiro operador que utilizamos muito é o operador unário not.

```
# Operador unário - Serve para inverter o valor de um booleana a fim de facilitar a verificação.

valido = False
print(not valido) # resultado: True
print(not True) # resultado: False
```

Operadores lógicos

O próximo operador que utilizamos muito é o operador and (E). O resultado de x and y será True se ambos x e y forem avaliados como True.

Caso contrário, o resultado será False. Se x for avaliado como False, y não será avaliado.

Operadores lógicos

Operador binário and - Checa a primeira condição se ela for False, nem avalia a segunda, só será True se ambas forem True.

```
def SecondOperand():
        print(”Avaliando segundo operador.")
        return True

a = False and SecondOperand()
print(a) # resultado: False
b = True and SecondOperand()
print(b) # resultado: Avaliando segundo operador. True
```

Operadores lógicos

O próximo operador que utilizamos muito é o operador or (OU). O resultado de x and y será True se um dos dois x ou y forem avaliados como True.

Se x for avaliado como True, y não será avaliado.

Operadores lógicos

Operador binário or - Checa a primeira condição. Se ela for True, nem avalia a segunda, só será True se uma das duas forem True.

```
def SecondOperand():
        print(”Avaliando segundo operador.")
        return True

a = False or SecondOperand()
print(a) # resultado: True
b = True or SecondOperand()
print(b) # resultado: Avaliando segundo operador. True
```

Estrutura de seleção

Nosso próximo assunto em Python: agora vamos conhecer a estrutura de seleção. Esta é responsável por separar no código algumas seleções de instruções que só serão executadas quando a condição que as envolver for alcançada:

- if (verificador de condição);

Estrutura de seleção

if - verifica se a condição inteira passada entre parênteses é verdadeira. Se sim, executa o treixo, se não, verifica se foi definido uma cláusula else e a executa.

```
valido = False

if valido:
        print(valido) # resultado: True
else:
        print(valido) # resultado: False
```

Estrutura de seleção

if - verifica se a condição inteira passada entre parênteses é verdadeira. Se sim, executa o treixo, se não, verifica se foi definido uma cláusula elif ou else e a executa.

```
nome = ‘Filipe’

If n == ‘João’:
        print(f’O nome é João’)
elif n == ‘Filipe’:
        print(f’O nome é Filipe’)
else:
        print(f’Nenhum dos nomes.’)
```

Operadores e estrutura de seleção

Agora que já entendemos como funcionam os operadores lógicos e estrutura de seleção, vamos exercitar algumas coisas:

- Criar um novo programa;
- Capturar as informações passadas na linha de comando;
- Aprender a utilizar a depuração do Visual Studio;
- Retornar para o usuário o resultado de acordo com a informação passada por ele.

Operadores e estrutura de seleção

Desafio:

Escrever um programa que recebe o nome e a idade e de acordo com uma lista de convidados, validar se a pessoa está na lista e é maior de idade, e retornar a mensagem de acordo com as validações feitas.

Café ou Dúvidas?

Para que a aula não fique cansativa, a cada conclusão de assunto teremos 15 minutos para que vocês possam tirar as dúvidas através dos comentários no vídeo e responderemos ao vivo.

Parte 4

- Demais operadores

- Método

Operador Atribuição

Este operador é o mais comum no dia-a-dia da programação. Ele serve atribuir valor a uma variável/constante.

valido = False
idade = 18
mensagem = ”Favor informar dados válidos”

Operador Comparação

Assim como a atribuição, esse operador utiliza o =, porém nesse caso serão dois == fazendo uma comparação, e não uma atribuição


```
if valido == False:
        print(”Favor informar dados válidos”)
```

Operador Comparação

Existe também um operador para verificar se os dados comparados são diferentes - este operador é o not

```
if not (valido == True):
        print(”Favor informar dados válidos”)
```

Operador Comparação

Os demais operadores de comparação são:

> Este operador compara se o valor a esquerda do operador é maior que o da direta, retornando True/False;

>= Este operador compara se o valor à esquerda do operador é maior ou igual ao da direta, retornando True/False.

Operador Comparação

Os demais operadores de comparação são:

< Este operador compara se o valor à esquerda do operador é menor que o da direta, retornando True/False;

<= Este operador compara se o valor à esquerda do operador é menor ou igual ao da direta, retornando True/False.

Operadores Matemáticos

Já vimos bastante os operadores matemáticos, vamos apenas relembrá-los

+ Efetua uma adição entre dois valores;

- Efetua uma subtração entre dois valores;

* Efetua uma multiplicação entre dois valores;

/ Efetua uma divisão entre dois valores;

% Efetua um módulo entre dois valores.

** Eleva um número a alguma potência

Métodos / Funções

Assim como já falamos de variáveis, também já falamos dos nossos métodos e funções.

Basicamente são comportamentos que nossa classe deseja realizar separadamente.

Os métodos são formadas por:

- Um nome que auxilia a identificar a função e chamá-la;
- Uma lista de variáveis/parâmetros de entrada que a função precisa para funcionar;
- Um escopo onde serão executadas as instruções respectivas dessa função.

Métodos / Funções

O método inicia com o nome e, entre parênteses, a lista de parâmetros de entrada para executar essa operação.

```
def Validar (valido):
        if(valido == True):
                return ‘O valor da variável é True’
        else
                return ‘O valor da variável é False’


teste = False
print(Validar(teste))
```

Operadores e Métodos

Agora que já entendemos como funcionam os demais operadores e criação de métodos. Nesse exercícios vamos exercitar algumas coisas:

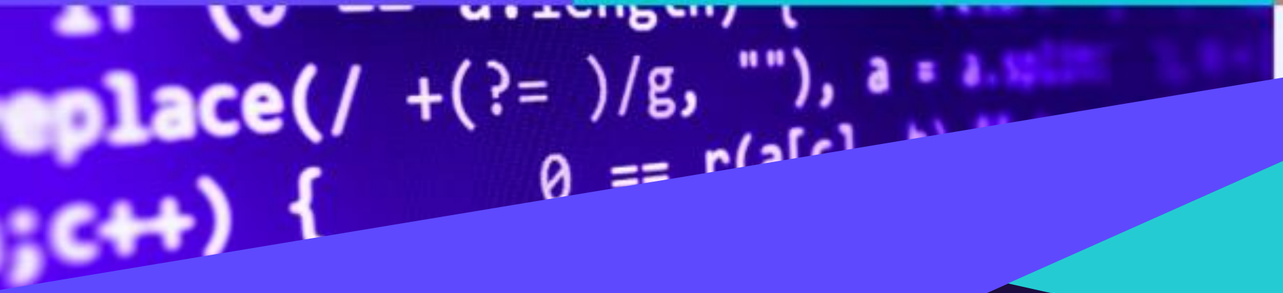
- Criar um novo programa;
- Capturar as informações passadas na linha de comando;
- Criar funções para desempenhar regras específicas;
- Realizar operações matemáticas;
- Retornar para o usuário o resultado de acordo com a informação passada por ele.

Operadores e Métodos

Desafio:

Escrever um programa que recebe dois números e um operador matemático e com isso executa o o calculo corretamente.

Não se esqueça de commitar
os programas desenvolvidos no seu Git =D



Obrigado pela participação

Esperamos que você tenha gostado, fique à vontade para nos enviar seus feedbacks sobre esta aula.

Esperamos você na próxima live, segunda-feira 03/05 as 19:30. Não se esqueça de consultar o calendário e anote na sua agenda.

Se inscreva no canal e siga-nos nas rede sociais:

www.youtube.com/c/Devaria

@devaria_oficial

@rafamazzucato

@castellodaniel

@oliveirandouglas

Referências Bibliográficas

- https://docs.python.org/pt-br/3/

Se inscreva no canal e siga-nos nas rede sociais:

www.youtube.com/c/Devaria

@devaria_oficial @rafamazzucato @castellodaniel @oliveirandouglas