



Faculdade

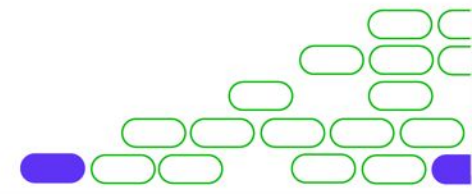


Fundamentos - Desenvolvedor Python

Capítulo 01 - Introdução ao Python

Aula 01 - Características da Linguagem

Antônio Carlos de Nazaré Júnior



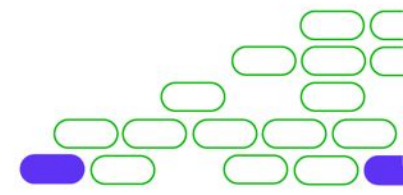
Quem sou eu?

- Formação
 - Graduação em Ciência da Computação (UFOP - 2012)
 - Mestrado em Ciência da Computação (UFMG - 2014)
 - Doutorando em Ciência da Computação (UFMG)
- Atuação Profissional
 - Pesquisa Aplicada (Smart Sense Laboratory - UFMG, 2014)
 - Pesquisa Aplicada (Coteminas, 2020)
 - Cientista de Dados (Localiza, 2021)
 - Cientista de Dados (isaac, 2022)
 - Professor (IGTI, 2022)



Quem sou eu?

- Áreas de Interesse
 - Ciência e Análise de Dados
 - Inteligência Artificial
 - Aprendizado de Máquina
 - Visão Computacional
 - Sistemas Biométricos
 - Computação Paralela
 - Sistemas Distribuídos



Sobre o Curso

- Curso básico e prático de introdução ao Python
- Principais conceitos e características da linguagem
- Vamos aprender desde a declaração de variáveis até conceitos mais avançados como manipulação de arquivos
- O curso é dividido em capítulos, cada um com suas aulas
- Esperamos ao final do curso que o aluno esteja apto a desenvolver o seu primeiro programa
- A partir de então vocês poderão continuar a caminhada para se tornar um desenvolvedor Python



Conteúdo da Aula

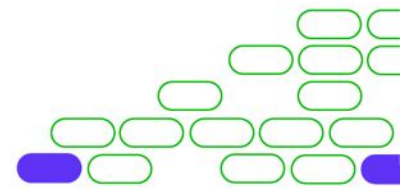
- O que são linguagens de programação
- Histórico e objetivos do Python
- Os principais motivos para aprender Python
- Classificação do Python como linguagem de programação





O que é uma linguagem de programação?

É uma linguagem formal, composta por um conjunto de símbolos e regras, que permitem os humanos controlarem o comportamento de uma máquina.



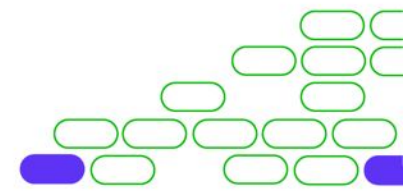
Linguagens de Programação

Linguagens Naturais

- É o principal meio de comunicação entre os humanos
- Não foram criadas por uma pessoa, pois se desenvolveram naturalmente
- Os idiomas (português, inglês, espanhol, francês etc.) são os maiores exemplos de linguagens naturais

Linguagens Formais

- São linguagens criadas pelas pessoas para uma aplicação em específico
- A notação matemática e as fórmulas químicas são exemplos de linguagens formais
- Linguagens Formais possuem um conjunto bem definido de:
 - **Símbolos e palavras-chaves**
 - **Regras sintáticas e semânticas**
- Linguagens de programação, são linguagens formais criadas para expressão de instruções para o computador



Linguagens de Programação

C++

```
3 #include <windows.h>
4 #include <stdlib.h>
5
6 int main()
7 {
8     struct tm *tmp;
9     time_t s;
10
11     for(;;)
12     {
13         s = time(NULL);
14         tmp = localtime(&s);
15
16         printf("%d:%d:%d\n", tmp->tm_hour, tmp->tm_min, tmp->tm_sec);
17
18         Sleep(1000);
19
20         system("cls");
21     }
22 }
23
24 return 0;
```



Java

```
1 public class Minesweeper
2 {
3     private int[][] myTruth;
4     private boolean[][] myShow;
5
6     public void cellPicked(int row, int col)
7     {
8         if( inBounds(row, col) && myShow[row][col] )
9         {
10             myShow[row][col] = true;
11
12             if( myTruth[row][col] == 0 )
13             {
14                 for(int r = -1; r <= 1; r++)
15                 {
16                     for(int c = -1; c <= 1; c++)
17                     {
18                         cellPicked(row + r, col + c);
19                     }
20                 }
21             }
22         }
23     }
24
25     public boolean inBounds(int row, int col)
26     {
27         return 0 <= row && row < myTruth.length && 0 <= col && col < myTruth[0].length;
28     }
29 }
```



JavaScript

```
<!DOCTYPE html>
<title>My Example</title>

<time id="date"></time>

<script>
/*
Create a JavaScript Date object for the current date and time,
then extract the desired parts, then join them again in the desired format.
*/
var currentDate = new Date(),
    day = currentDate.getDate(),
    month = currentDate.getMonth() + 1,
    year = currentDate.getFullYear(),
    date = day + "/" + month + "/" + year;

// Output the date to the above HTML element
document.getElementById("date").innerHTML = date;
</script>

<!-- via http://www.quackit.com -->
```



Python

```
import urllib
import re

print "we will try to open this url, in order to get IP Address"

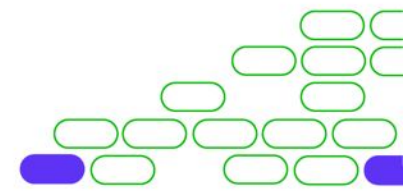
url = "http://checkip.dyndns.org"

print url

request = urllib.urlopen(url).read()

theIP = re.findall(r"d{1,3}.d{1,3}.d{1,3}.d{1,3}", request)

print "your IP Address is: ", theIP
```

A Linguagem Python

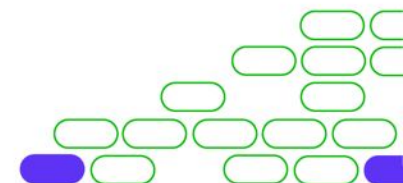
- Criada em 1991, por Guido van Rossum
 - Uma evolução da linguagem ABC
 - Foi o responsável pela evolução da linguagem
 - Ele continuou sendo o líder do projeto até 2018
- Apesar da similaridade, o nome da linguagem não tem relação com a cobra “Píton”
- O nome Python foi uma homenagem ao seu programa de humor favorito: “Monty Python's Flying Circus”



Imagens extraídas de:

<https://education.ti.com/pt/t3-europe-sites/t3-europe/edublogs/interview-guido-rossum>

<https://www.amazon.com/Monty-Pythons-Flying-Circus-Season/dp/B001NXTG4E>



A Linguagem Python

- Em 1999, Guido von Rossum definiu quais seriam os objetivos do Python:



Intuitiva e Poderosa

Ser uma uma linguagem fácil e intuitiva, e ao mesmo tempo poderosa

Open Source

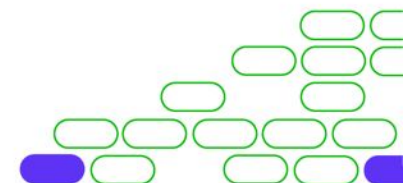
O código deve ser aberto, para que qualquer pessoa possa contribuir

Simples

Escrever um código em Python deve ser tão simples quanto escrever um texto em inglês

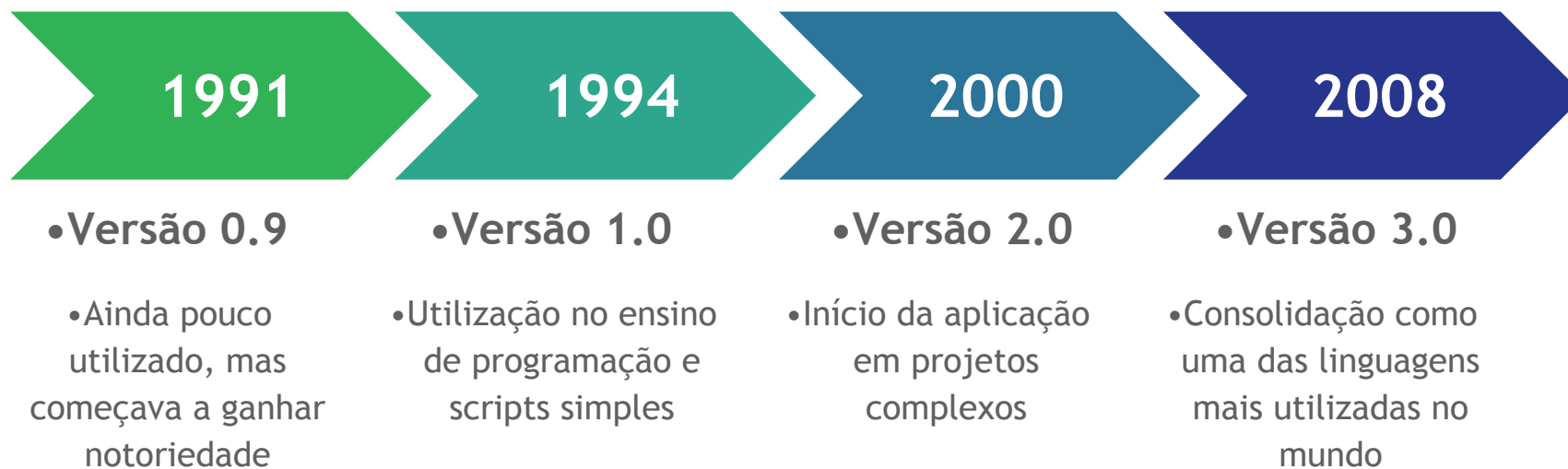
Versátil

Versátil para utilização desde tarefas simples até soluções completas



A Linguagem Python

- Evolução das versões do Python:



Motivos para Aprender Python

- Principais motivações para se aprender Python atualmente:

1 Fácil de Aprender		3 Suporte		5 Aplicações
	2 Versatilidade		4 Popularidade	



Motivos para Aprender Python

- Python é a linguagem de programação mais popular¹ do mundo...











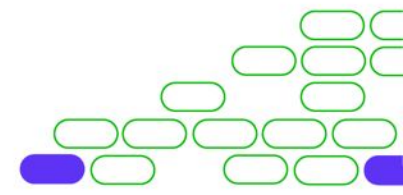
Mar 2022	Mar 2021	Change	Programming Language	Ratings	Change
1	3	▲	 Python	14.26%	+3.95%
2	1	▼	 C	13.06%	-2.27%
3	2	▼	 Java	11.19%	+0.74%
4	4		 C++	8.66%	+2.14%
5	5		 C#	5.92%	+0.95%
6	6		 Visual Basic	5.77%	+0.91%
7	7		 JavaScript	2.09%	-0.03%
8	8		 PHP	1.92%	-0.15%
9	9		 Assembly language	1.90%	-0.07%
10	10		 SQL	1.85%	-0.02%

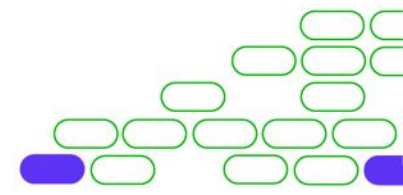
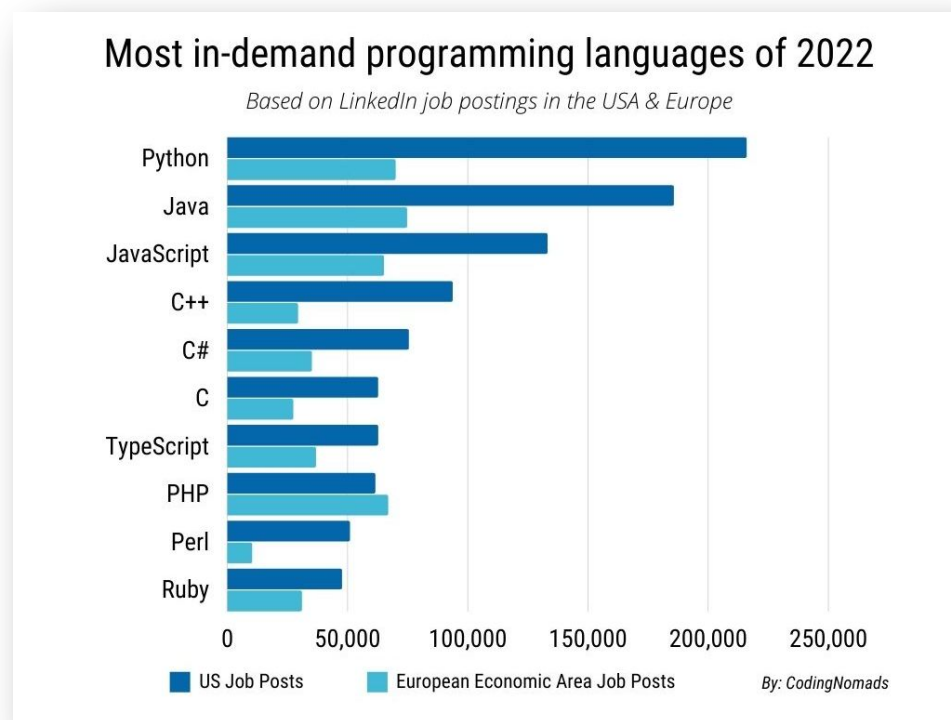
Imagem capturada de: <https://www.tiobe.com/tiobe-index/>

1. Fonte: TIOBE index - o ranking criado pela empresa TIOBE é um indicador da popularidade das linguagens de programação.



Motivos para Aprender Python

- Existe uma grande demanda por desenvolvedores Python...



Motivos para Aprender Python

- Python permite construir diferentes tipos de soluções...

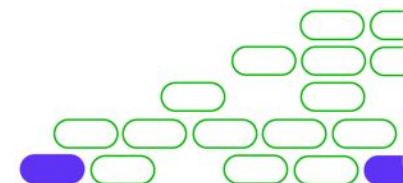


Motivos para Aprender Python

- Várias empresas, mundialmente conhecidas, utilizam Python...

 netguru

Which companies use **Python**?



Características da Linguagem

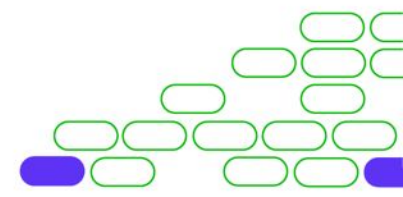
- As linguagens de programação podem ser classificadas em diversas categorias, de acordo com as suas funcionalidades e características
- Python pode receber as seguintes classificações:

✓ Linguagem de Alto Nível

✓ Linguagem Interpretada

✓ Tipagem Dinâmica

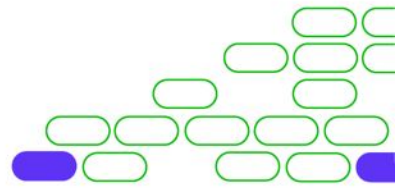
✓ Multiparadigma



Conclusão



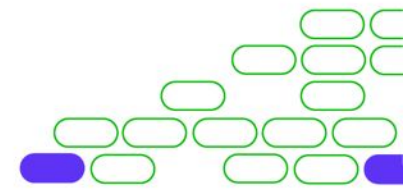
- ✓ O que é uma linguagem de programação
- ✓ Histórico e a importância do Python
- ✓ Motivos para se aprender Python
- ✓ Características do Python como uma linguagem de programação



Próxima Aula



- Vantagens e Desvantagens do Python
- Quando Devemos Escolher o Python?





Faculdade

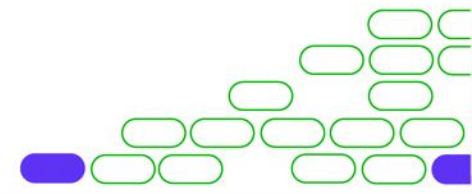


Fundamentos - Desenvolvedor Python

Capítulo 01 - Introdução ao Python

Aula 02 - Vantagens e Desvantagens do Python

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- As vantagens e as desvantagens do Python
- Escolha do Python para o desenvolvimento de um projeto



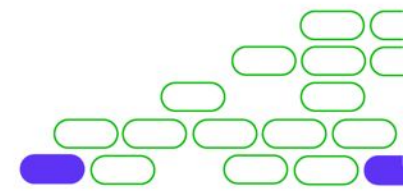
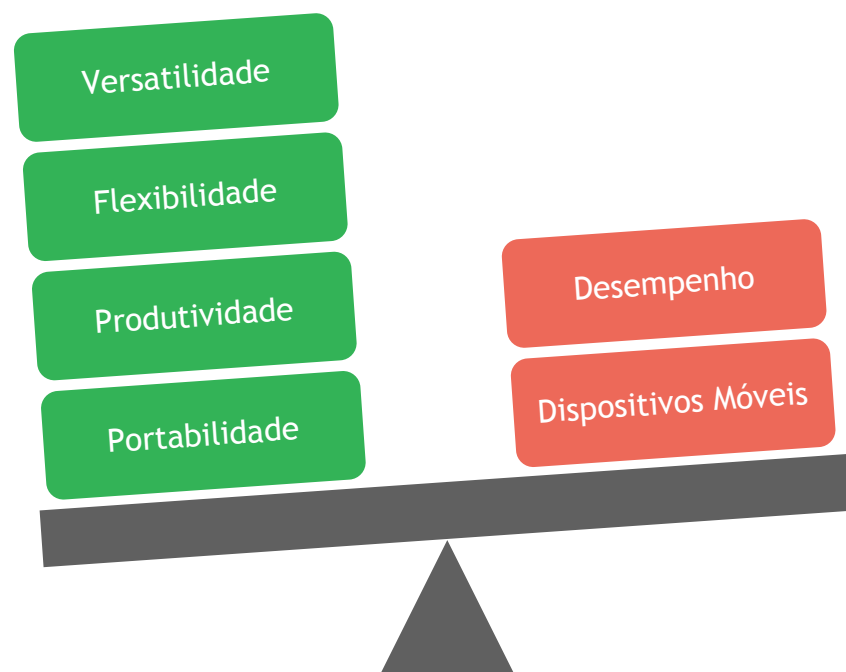
Python é a melhor escolha para o desenvolvimento de uma aplicação?

A resposta é... Depende! Para tomar esta decisão, devemos antes conhecer suas vantagens e suas limitações!



Vantagens e Desvantagens do Python

- Como toda linguagem, o Python tem suas vantagens e desvantagens....



Vantagens e Desvantagens do Python

Vantagens

- Facilidade de Aprendizado
 - Python se concentra na legibilidade do código
 - Código bem estruturado e com poucas linhas
 - Utilizado por diversas universidades como linguagem de ensino
- Versatilidade e Flexibilidade
 - Permite solucionar tarefas básicas ou construir projetos complexos
 - Diferentes paradigmas de programação
 - É possível começar com um script simples e evoluir para uma solução completa.



Vantagens e Desvantagens do Python

Vantagens

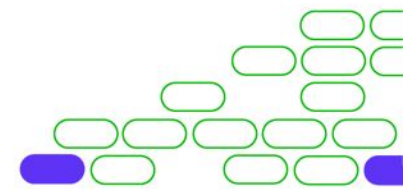
- Prototipação e Produtividade
 - “Fazer mais, com menos código!”
 - Construção de protótipos e testes de ideias em minutos
 - Economia de tempo e aumento da produtividade
- Diversidade de Bibliotecas
 - Existem bibliotecas para as mais diversas aplicações do mundo real
 - Frameworks WEB, IA, manipulação de dados, mercado financeiro, desenvolvimento de jogos e muitas outras
 - E se não houver, é muito fácil criar sua própria biblioteca

Vantagens e Desvantagens do Python

Vantagens

- Open Source
 - É possível baixar o Python e sair utilizando em questão de minutos
 - Você pode utilizar e modificar o seu código livremente¹
 - Por ser livre, a comunidade do Python é extensa e muita ativa
 - Muito fácil de obter ajuda para problemas
- Portabilidade
 - Python é compatível com todos os sistemas operacionais
 - Execução do mesmo código em diferentes plataformas, sem necessidade de adaptação

¹ - Entretanto, é necessário se atentar às licenças de utilização de códigos de terceiros.



Vantagens e Desvantagens do Python

Desvantagens

- Limitação de Desempenho e Alto Consumo de Memória
 - Por ser interpretada e dinâmica, possui um desempenho inferior quando comparada com outras linguagens
 - O consumo de memória também é maior em relação a outras linguagens
 - No geral, essas limitações não interferem no desempenho final da maioria das aplicações, mas é sempre recomendado analisar cada caso
 - Existem diferentes estratégias para otimizar o desempenho e diminuir o consumo de memória
- Dispositivos Móveis
 - Android e iOS não suportam Python nativamente
 - Existem frameworks que permitem a utilização, mas requerem um esforço adicional

Vantagens e Desvantagens do Python

Desvantagens

- Problemas com Paralelização
 - Paralelismo está longe de ser um ponto forte em Python
 - Técnicas de computação paralela permitem a execução de múltiplos fluxos de código ao mesmo tempo
 - Redução considerável do tempo de execução
 - Python restringe que apenas um fluxo de código seja executado por vez
 - Existem bibliotecas de multiprocessamento que contornam esta limitação, mas requerem um conhecimento mais avançado



Quando Devemos Escolher o Python?

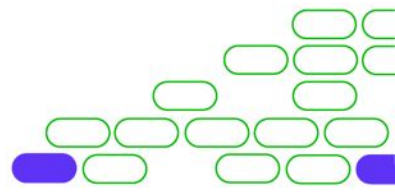
- Não existe uma resposta única e 100% correta para essa pergunta
- Devemos sempre considerar a natureza da aplicação a ser desenvolvida
- Perguntas como estas são sempre úteis:
 - Minha aplicação será executada onde? Em um servidor? Em um smartphone?
 - Quais as limitações eu devo considerar?
 - É um protótipo? É um código simples? É um projeto complexo?
 - Qual o tamanho da equipe estará envolvida no projeto?
- No geral, considerando as exceções causadas pelas limitações apresentadas, o Python será uma boa escolha para o projeto



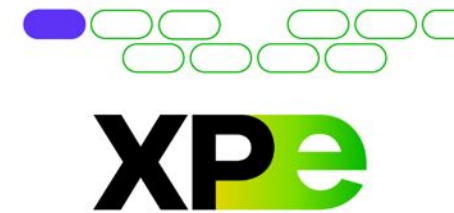
Conclusão



- ✓ Aprendemos quais são as vantagens e limitações do Python
- ✓ A natureza da aplicação deve ser considerada ao escolher uma linguagem
- ✓ Existem situações onde não é recomendável a utilização do Python



Próxima Aula



- Como instalar e configurar um ambiente para desenvolvimento em Python
- Como acessar o Google Colab, nossa ferramenta durante o curso





Faculdade

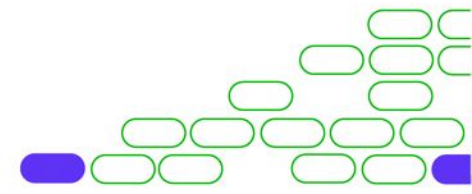


Fundamentos - Desenvolvedor Python

Capítulo 01 - Introdução ao Python

Aula 03 - Preparação do Ambiente Python

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula

- Como instalar e configurar um ambiente para desenvolvimento em Python
- Como acessar o Google Colab, nossa ferramenta durante o curso






Preparação do Ambiente Python

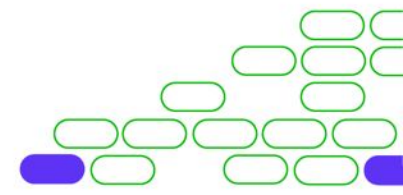
- Enquanto a maioria das distribuições Linux e o MacOS já possuem o Python configurado, o Windows exige a instalação por parte do usuário
- Mesmo com uma versão pré-configurada, é sempre recomendado instalar e atualizar para a versão mais recente do Python
- Para o desenvolvimento do curso, iremos utilizar uma ferramenta on-line e gratuita com o Python já configurado
- Existem diversos tutoriais na internet com instruções atualizadas de como instalar o Python
- A seguir serão listadas sugestões de tutoriais para aqueles que desejam realizar a instalação



Preparação do Ambiente Python

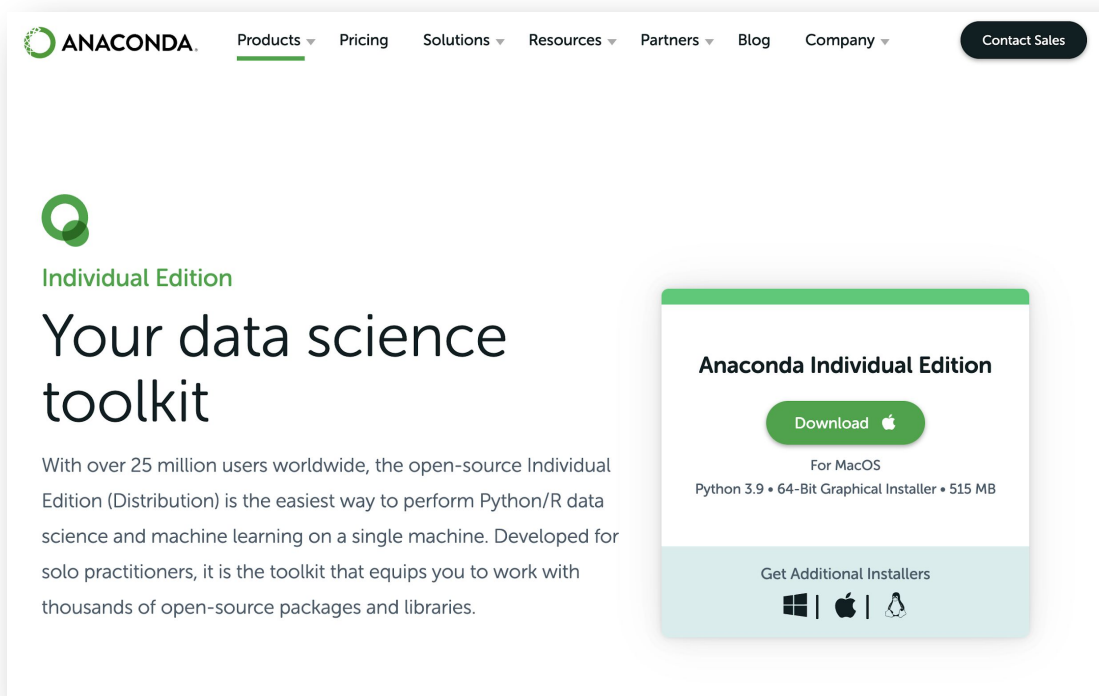
- Sugestões de tutoriais para instalação em diferentes sistemas operacionais

Sistema	Tutoriais
 Windows	<p>How to Install Python on Windows (inglês): https://realpython.com/installing-python/#how-to-install-python-on-windows</p> <p>Instalando o Python 3 no Windows (português): https://python.org.br/instalacao-windows/</p>
 Linux	<p>How to Install Python on Linux (inglês): https://realpython.com/installing-python/#how-to-install-python-on-linux</p> <p>Instalando o Python no Linux (português): https://python.org.br/instalacao-linux/</p>
 macOS	<p>How to Install Python on macOS (inglês): https://realpython.com/installing-python/#how-to-install-python-on-macos</p> <p>Instalando o Python no Mac OS (português): https://python.org.br/instalacao-mac/</p>



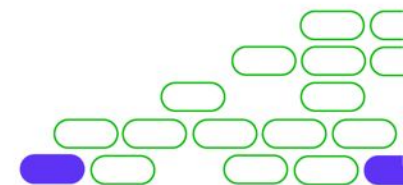
Preparação do Ambiente Python

- Existe também o Anaconda, que é um “pacotão para o Python”



O Anaconda é gratuito para utilização individual.

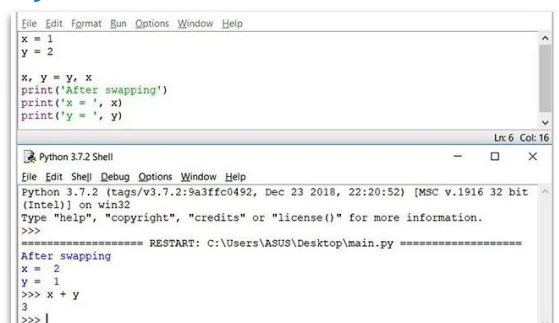
O download, assim como as instruções de instalação, para cada sistema operacional está disponível na página do projeto: <https://www.anaconda.com>



Preparação do Ambiente Python

- Podemos também escolher diferentes IDE's para o desenvolvimento

Python Iddle



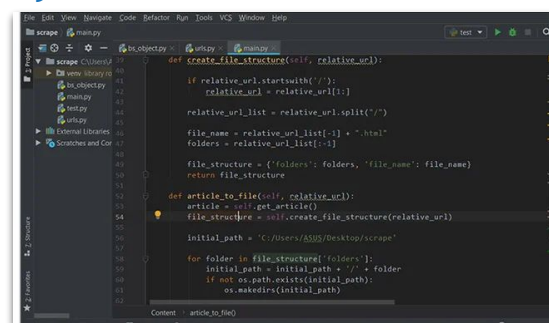
The screenshot shows the Python Iddle IDE interface. The top window contains a simple Python script for swapping variables x and y. The bottom window shows the Python 3.7.2 Shell, which has executed the script and printed the output: 'After swapping', 'x = 2', 'y = 1', and '3'.

```
x = 1
y = 2

x, y = y, x
print('After swapping')
print('x = ', x)
print('y = ', y)
```

```
Python 3.7.2 Shell
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ASUS\Desktop\main.py =====
After swapping
x = 2
y = 1
>>> x + y
3
>>>
```

PyCharm

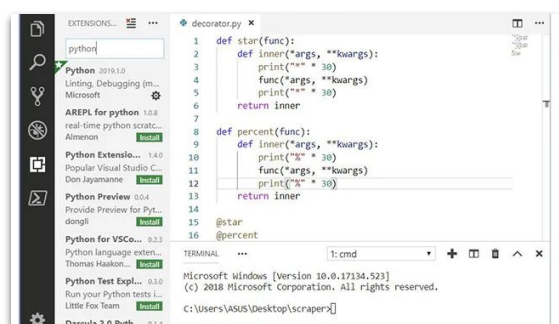


The screenshot shows the PyCharm IDE interface. The main editor displays a Python script for creating a file structure. The left sidebar shows the project structure with folders like 'src' and 'tests'.

```
def create_file_structure(self, relative_url):
    if relative_url.startswith('/'):
        relative_url = relative_url[1:]
    relative_url_list = relative_url.split('/')
    file_name = relative_url_list[-1] + ".html"
    folders = relative_url_list[:-1]
    file_structure = {'folders': folders, 'file_name': file_name}
    return file_structure

def article_to_file(self, relative_url):
    article = self.get_article()
    file_structure = self.create_file_structure(relative_url)
    initial_path = 'C:/Users/ASUS/Desktop/scrape'
    for folder in file_structure['folders']:
        initial_path = initial_path + '/' + folder
        if not os.path.exists(initial_path):
            os.makedirs(initial_path)
```

Visual Studio Code



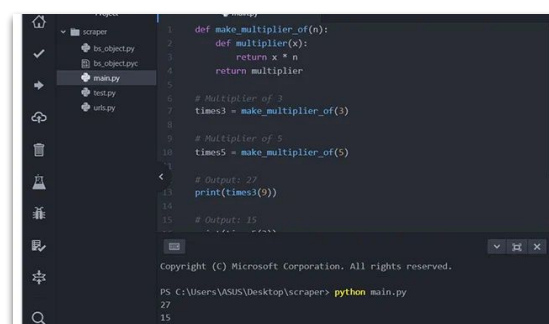
The screenshot shows the Visual Studio Code IDE interface. The left sidebar shows the Extensions view with various Python-related extensions installed. The main editor displays a Python script for a decorator function.

```
def star(func):
    def inner(*args, **kwargs):
        print("*" * 30)
        func(*args, **kwargs)
        print("*" * 30)
    return inner

def percent(func):
    def inner(*args, **kwargs):
        print("%" * 30)
        func(*args, **kwargs)
        print("%" * 30)
    return inner

@star
@percent
```

Atom



The screenshot shows the Atom IDE interface. The main editor displays a Python script for a multiplier function. The bottom status bar shows the file path: 'C:\Users\ASUS\Desktop\scrapers\main.py'.

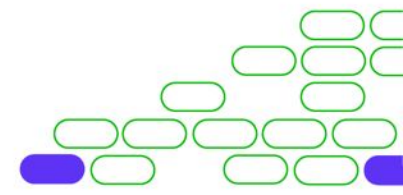
```
def make_multiplier_of(n):
    def multiplier(x):
        return x * n
    return multiplier

# Multiplier of 3
times3 = make_multiplier_of(3)

# Multiplier of 5
times5 = make_multiplier_of(5)

# Output: 27
print(times3(9))

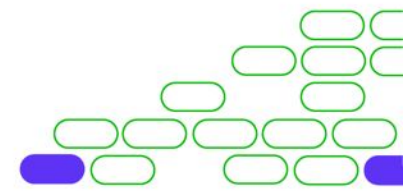
# Output: 45
print(times5(9))
```



Como Acessar o Google Colab

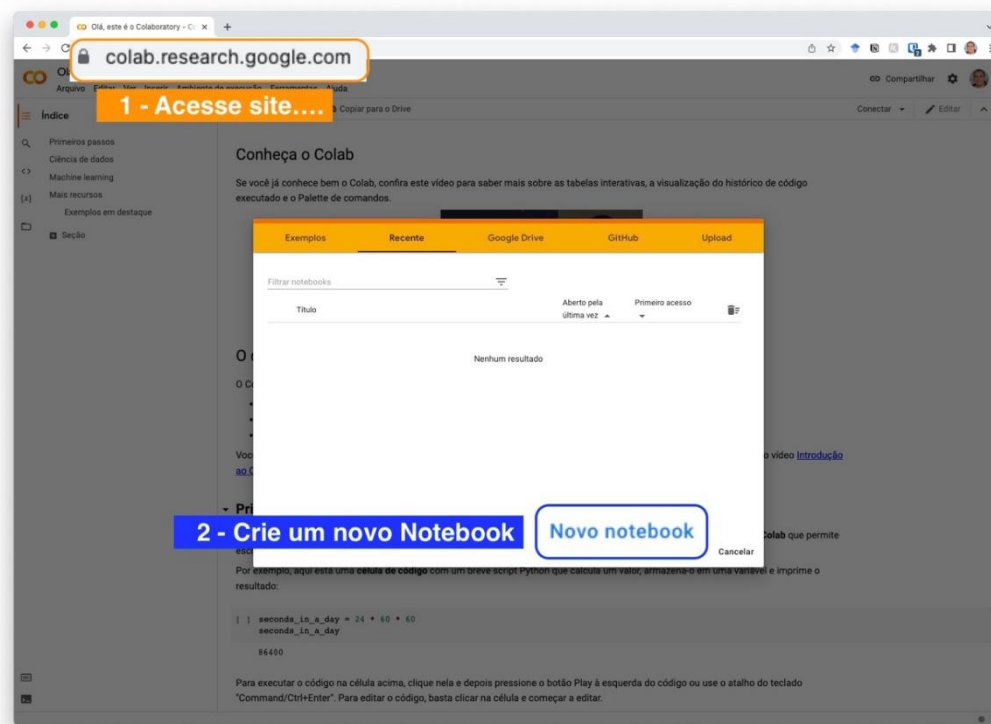
- O Google Colaboratory (Colab)¹ é uma ferramenta gratuita hospedada na própria nuvem da Google
- Possibilita a escrita e execução de códigos Python sem a necessidade de nenhuma configuração. Basta entrar e codificar!
- Ele utiliza o conceito de notebooks, que permite que intercalemos o código fonte com a saída de cada execução
- É possível salvar o notebook e voltar a editá-lo quando quiser, como se fosse um documento on-line do Google Docs
- Vamos utilizá-lo para executarmos todos os códigos de exemplos do curso.

¹ - <https://colab.research.google.com/>



Como Acessar o Google Colab

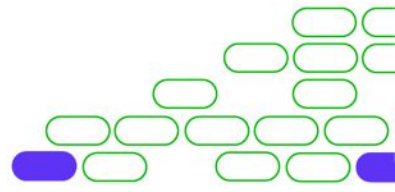
- Basta ter uma conta do Google e acessar: colab.research.google.com/



Conclusão



- ✓ Aprendemos que existem diferentes meios de obter o Python
- ✓ Podemos ter nossa própria instalação ou utilizar uma ferramenta on-line
- ✓ Nossa ferramenta durante o curso será o Google Colab



Próxima Aula



- Chega de teoria, queremos executar o nosso primeiro código!





Faculdade

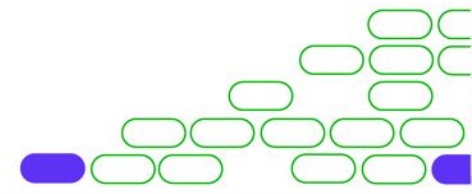


Fundamentos - Desenvolvedor Python

Capítulo 01 - Introdução ao Python

Aula 04 - Executando o Primeiro Código

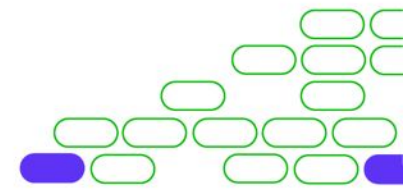
Antônio Carlos de Nazaré Júnior




Conteúdo da Aula



- Vamos partir para a prática e executar o nosso primeiro código Python



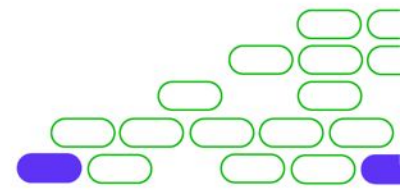


Conteúdo Prático no Google Colab

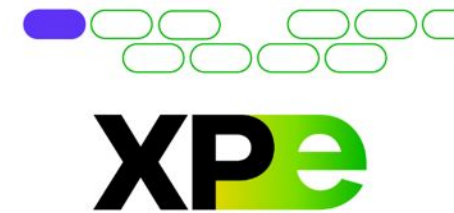
Conclusão



- ✓ Escrevemos e executamos o nosso primeiro código em Python



Próxima Aula



- Vamos começar a escrever nossos primeiros códigos
- Elementos da sintaxe do Python
- Como estruturar um código em Python





Faculdade

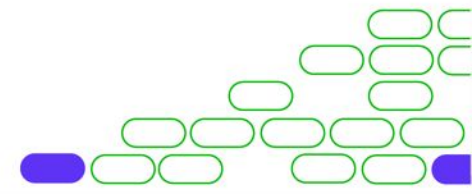


Fundamentos - Desenvolvedor Python

Capítulo 02 - Escrita de Códigos em Python

Aula 01 - A Sintaxe da Linguagem

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Sintaxe e Semântica de uma Linguagem
- A Sintaxe da Linguagem Python
- Elementos da Sintaxe do Python



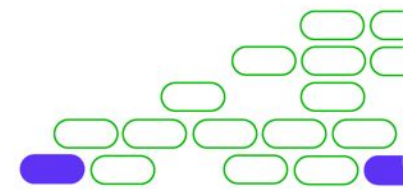
Sintaxe e Semântica de uma Linguagem

Sintaxe

- Refere-se à sua estrutura de escrita
- Não considera o significado das palavras
- Ela é composta por um conjunto de regras, que valida a sequência de palavras, símbolos e/ou instruções que é utilizada

Semântica

- Trata-se da análise do significado das palavras, expressões, símbolos e instruções da linguagem.
- A semântica é importante para que os desenvolvedores saibam precisamente o que as instruções fazem

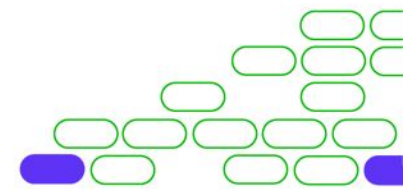


A Sintaxe da Linguagem Python

- Vamos considerar o seguinte exemplo de código...

```
>>> # define o valor do limiar↵
... limiar = 5↵
... ↵
... menores = [] # cria lista menores↵
... maiores = [] # cria lista maiores↵
... ↵
... # divide os números de 1 a 10 em maiores e menores↵
... for i in range(10): ↵
...     if (i < limiar):↵
...         menores.append(i)↵
...     elif (i > limiar):↵
...         maiores.append(i)↵
...     ↵
... # imprime na tela os valores das listas↵
... print('Resultado final')↵
... print('menores:', menores)↵
... print('maiores:', maiores)↵
```

```
Resultado final
menores: [0, 1, 2, 3, 4]
maiores: [6, 7, 8, 9]
```



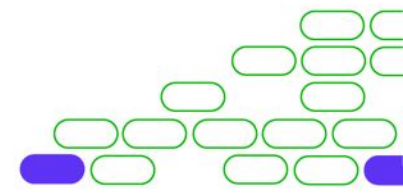
Elementos da Sintaxe do Python

- Vamos considerar o seguinte exemplo de código...

```
>>> # define o valor do limiar
... limiar = 5
...
... # divide os números de 1 a 10 em maiores e menores
... for i in range(10):
...     if i < limiar:
...         menores.append(i)
...     elif i > limiar:
...         maiores.append(i)
...
... # imprime na tela os valores das listas
... print('Resultado final')
... print('menores:', menores)
... print('maiores:', maiores)

Resultado final
menores: [0, 1, 2, 3, 4]
maiores: [6, 7, 8, 9]
```

- Comentários
- Quebras de Linhas
- Indentação
- Espaços em Branco
- Parênteses



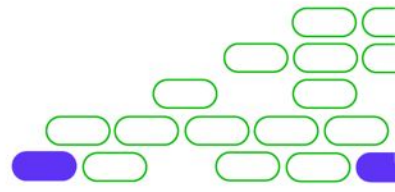


Conteúdo Prático

Conclusão



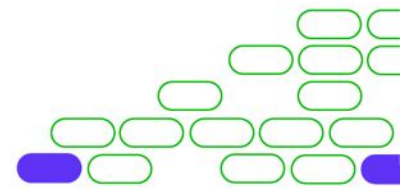
- ✓ Diferença entre sintaxe e semântica de uma linguagem de programação
- ✓ Estrutura da sintaxe do Python
- ✓ Principais elementos da sintaxe do Python



Próxima Aula



- Como declarar variáveis em Python
- Regras para nomear variáveis
- Palavras-chave do Python





Faculdade

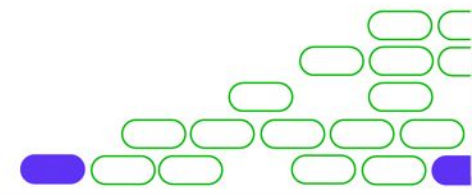


Fundamentos - Desenvolvedor Python

Capítulo 02 - Escrita de Códigos em Python

Aula 02 - Variáveis e Keywords

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula

- Como declarar variáveis e atribuir valores em Python
- Regras para nomeação de variáveis
- Palavras-chave do Python





Conteúdo Prático

Regras para Nomeação de Variáveis

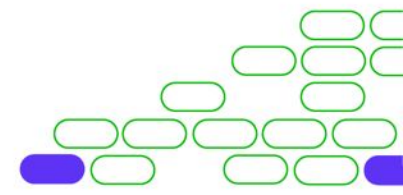
- Uma variável pode conter letras e números, mas não pode começar com um número
- É permitido utilizar letras minúsculas e maiúsculas para nomes de variáveis. Entretanto, é recomendável utilizar apenas letras minúsculas
- Não é permitido utilizar caracteres especiais, espaço em branco e pontuação
 - @ # \$ % * ^ ' " [] () { } (. ! : ? ;)
- O único símbolo permitido é o caractere *underline*/sublinhado (_), e as variáveis podem começar com ele também
- Versões recentes do Python suportam letras acentuadas, mas é extremamente recomendável não utilizá-las



Palavras-Chave do Python

- Existe um conjunto de palavras reservados que não podemos utilizar como nome de variáveis:

False	None	True	and	as	assert	async
await	break	class	continue	def	del	elif
else	except	finally	for	from	global	if
import	in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with	yield



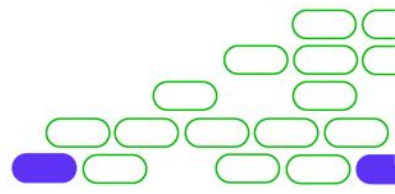


Conteúdo Prático

Conclusão



- ✓ Como declarar uma variável e atribuir a ela um valor
- ✓ Regras para nomeação das variáveis
- ✓ Palavras-chave do Python que não podemos utilizar



Próxima Aula



- Tipos de Erro em Python
- Como Interpretar os Erros do Python



Faculdade

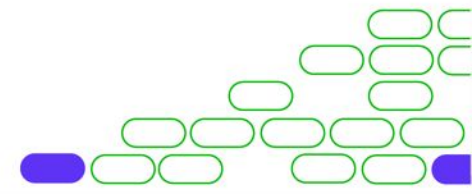


Fundamentos - Desenvolvedor Python

Capítulo 02 - Escrita de Códigos em Python

Aula 03 - Tipos de Erros do Python

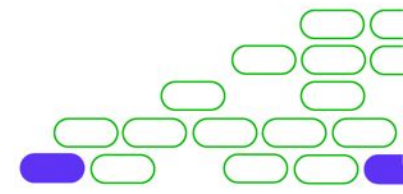
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Vamos conhecer os tipos de erros em Python
- Vamos interpretar e localizar estes erros no código



Tipos de Erros do Python

- Quando estamos desenvolvendo ou executando o nosso código, podemos deparar com erros
- Os erros são causados por violações das regras de escrita ou por inconsistência durante as operações realizadas
- Os três principais tipos de erro no Python são:
 - Erros de Sintaxe
 - Erros em Tempo de Execução
 - Erros Lógicos
- Conhecer e saber interpretar estes erros é muito importante para o aprendizado da linguagem e também durante o desenvolvimento

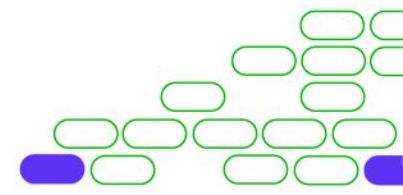


Erros de Sintaxe

- São os erros ocasionados pela violação da sintaxe do Python

```
>>> x = 3
... y = 5
... print x + y

File "<ipython-input-6-04df4e714b51>", line 3
    print "Esse é um tipo de erro"
      ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(x + y)?
```



Erros em Tempo de Execução

- Surgem quando o interpretador não consegue executar uma instrução devido a alguma inconsistência

```
>>> # utilização de uma variável inexistente (não definida)
... x = 0
... print(xx)

NameError                                Traceback (most recent call last)
<ipython-input-3-eab9121ba877> in <module>()
      1 # utilização de uma variável não existente
      2 x = 0
----> 3 print(xx)

NameError: name 'xx' is not defined

>>> # declaração incorreta da variável dia, como Dia
... dia = 'Segunda Feira'
... print(Dia)

NameError                                Traceback (most recent call last)
<ipython-input-2-2db379d425a4> in <module>()
      1 # declaração incorreta da variável dia, como Dia
      2 dia = 'Segunda Feira'
----> 3 print(Dia)

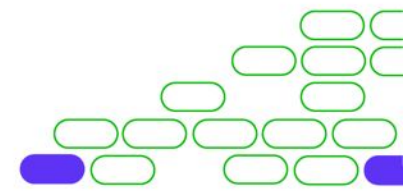
NameError: name 'Dia' is not defined
```

Erros Lógicos

- É o tipo de erro mais difícil de lidar, pois o erro não é de fato detectado

```
>>> x = 5.0
... y = 7.0
... z = 12.0
... media = x + y + z / 3
... print(media)
16.0
```

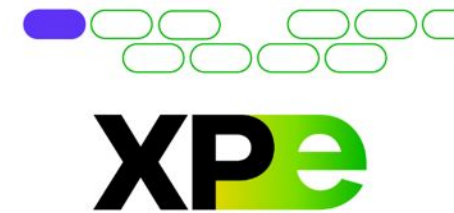
```
>>> x = 5.0
... y = 7.0
... z = 12.0
... media = (x + y + z) / 3
... print(media)
8.0
```





Conteúdo Prático

Conclusão



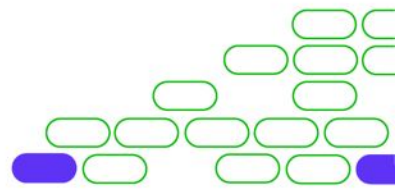
- ✓ Conhecemos os tipos de erro do Python
- ✓ Aprendemos a interpretar estes erros e usá-los a nosso favor



Próxima Aula



- Tipos de Dados em Python





Faculdade

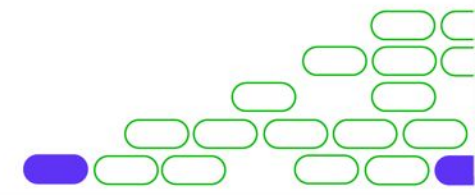


Fundamentos - Desenvolvedor Python

Capítulo 03 - Tipos Primitivos de Dados e Operadores

Aula 01 - Tipos de Primitivos de Dados em Python

Antônio Carlos de Nazaré Júnior



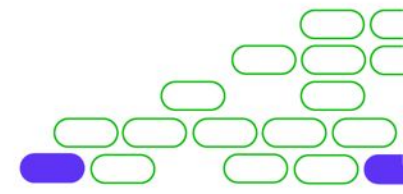
Conteúdo da Aula

- O que é um tipo de dado?
- Os tipos primitivos `int`, `float`, `bool` e `str`
- O tipo especial `NoneType`
- A função `type()`



Tipos de Dados em Python

- Um programa manipula diferentes informações como: números inteiros e racionais, palavras, textos e outras estruturas mais complexas.
- Para categorizar a natureza destas informações, utilizamos uma determinada classificação, chamada de *tipos de dados*.
- O tipo de dado também é responsável por definir quais operações podemos realizar com aquele dado em específico.
 - Por exemplo, podemos multiplicar números entre si, mas não podemos multiplicar uma palavra com outra.
- As linguagens de programação, normalmente, possuem alguns tipos de dados pré-definidos, chamados de *tipos primitivos de dados*.

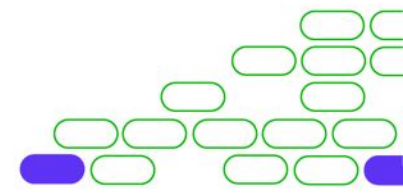


Tipos Primitivos de Dados

- O Python oferece quatro tipos primitivos (`int`, `float`, `bool` e `str`)

Tipo	Descrição	Exemplos
<code>int</code>	Representa números inteiros	0, 99, -3, -87
<code>float</code>	Representa números racionais	3.5, -9.1, 3.141592
<code>bool</code>	Assume apenas dois valores, <i>verdadeiro</i> ou <i>falso</i>	True, False
<code>str</code>	É uma sequência/cadeia de caracteres utilizada para representar palavras, frases ou textos.	'a', 'São Paulo', "céu"

- Também existe o tipo `NoneType`, que assume o valor *nulo* `None`.



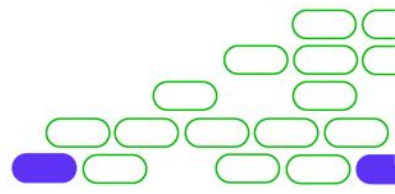


Conteúdo Prático

Conclusão



- ✓ Conhecemos os tipos de dados primitivos do Python
- ✓ Aprendemos sobre o valor nulo **None**
- ✓ Utilizamos a função **type()** para descobrir o tipo de uma variável



Próxima Aula



- Operações com os tipos primitivos de dados
- Operadores aritméticos, de comparação e lógicos





Faculdade

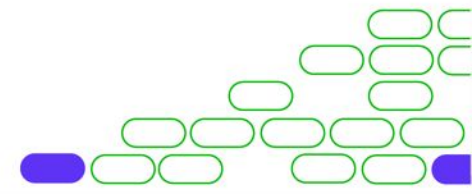


Fundamentos - Desenvolvedor Python

Capítulo 03 - Tipos Primitivos de Dados e Operadores

Aula 02 - Operadores dos Tipos Primitivos de Dados

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula

- Operações com os tipos primitivos de dados
- Operadores aritméticos
- Operadores de comparação
- Operadores lógicos



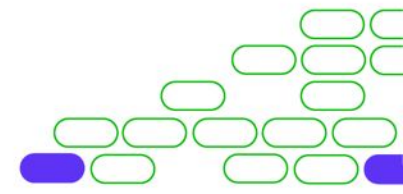
Operadores em Python



Aritméticos	
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto)
//	Divisão inteira
**	Exponenciação


Comparação	
>	Maior que
>=	Maior ou igual que
<	Menor que
<=	Menor ou igual que
==	Igual
!=	Diferente

Lógicos	
and	Conjunção
or	Disjunção
not	Complemento

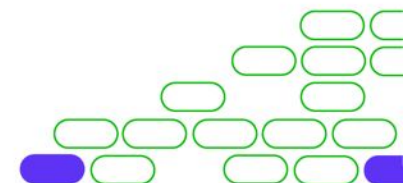


Precedência dos Operadores Aritméticos

- O Python possui regras de precedência que regem a ordem de execução das operações dentro de uma expressão aritmética.
- É a mesma convenção tradicional da matemática, conhecida como *PEMDAS*:

A blue vertical bar with a white downward-pointing arrow at the bottom, indicating the order of operations from top to bottom.

P	Parênteses
E	Exponenciação
M	Multiplicação
D	Divisão
A	Adição
S	Subtração



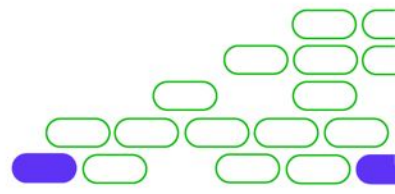


Conteúdo Prático

Conclusão



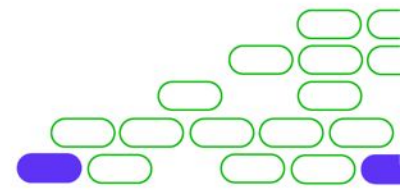
- ✓ Aprendemos a utilizar os operadores aritméticos
- ✓ Realizamos comparações, utilizando os operadores de comparação
- ✓ Aprendemos também sobre os operadores lógicos



Próxima Aula



- Operações com *strings*





Faculdade

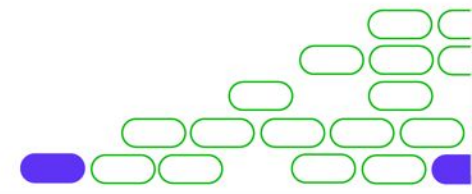


Fundamentos - Desenvolvedor Python

Capítulo 03 - Tipos Primitivos de Dados e Operadores

Aula 03 - Operações com Strings

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula

- Operador de acesso das *strings*
- Operações de concatenação e repetição
- Operações de filiação
- Métodos das *strings*



Representação de uma String

- Uma *string* é formada por uma sequência de caracteres...

c	o	n	s	o	l	a	ç	ã	o
0	1	2	3	4	5	6	7	8	9
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1





Conteúdo Prático

Conclusão



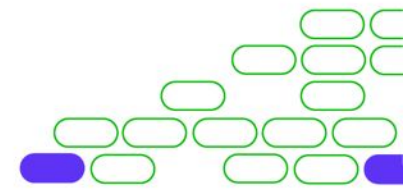
- ✓ Aprendemos como as *strings* são representadas no Python
- ✓ Utilizamos os operadores e métodos das *strings*



Próxima Aula



- Conversão e formatação dos tipos de dados





Faculdade

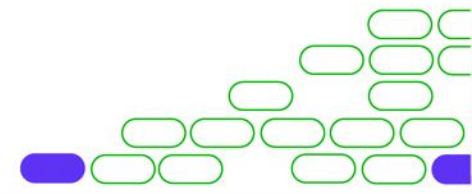


Fundamentos - Desenvolvedor Python

Capítulo 03 - Tipos Primitivos de Dados e Operadores

Aula 04 - Conversão dos Tipos de Dados

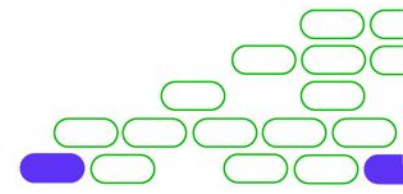
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



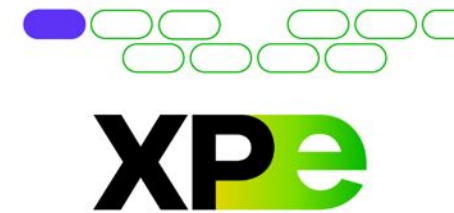
- Conversão entre os tipos de dados
- Entender as particularidades das conversões de tipos
- Formatação dos tipos





Conteúdo Prático

Conclusão



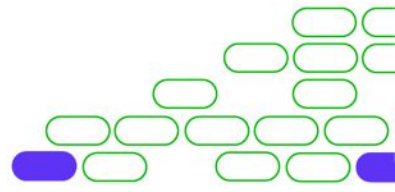
- ✓ Realizamos conversões entre os tipos de dados
- ✓ Formatação dos valores em uma *string* única



Próxima Aula



- Estruturas Condicionais





Faculdade

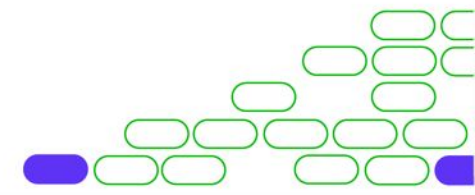


Fundamentos - Desenvolvedor Python

Capítulo 04 - Fluxos de Controle em Python

Aula 01 - Estruturas Condicionais

Antônio Carlos de Nazaré Júnior

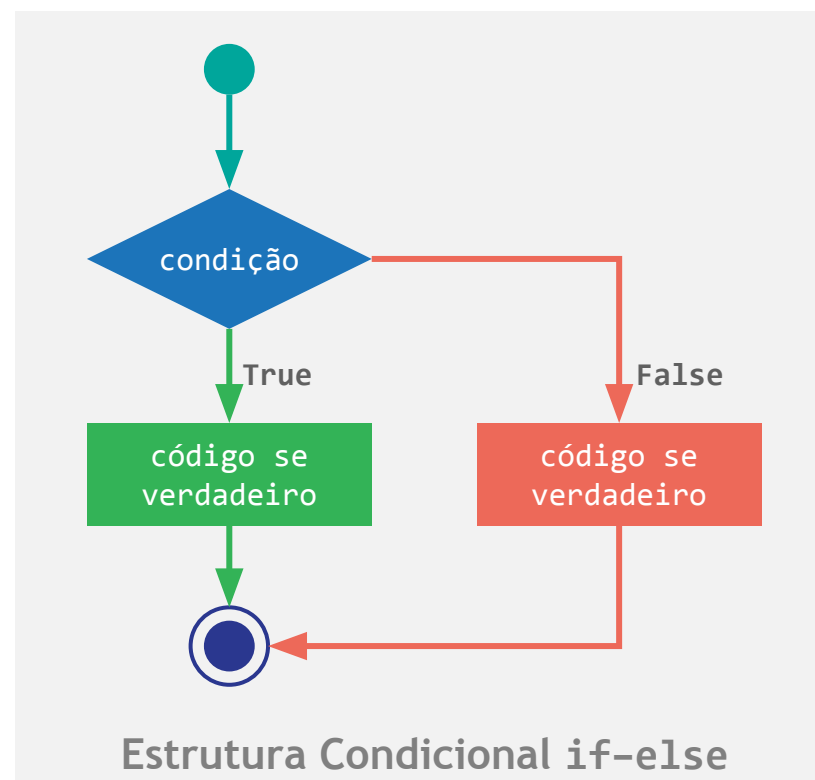
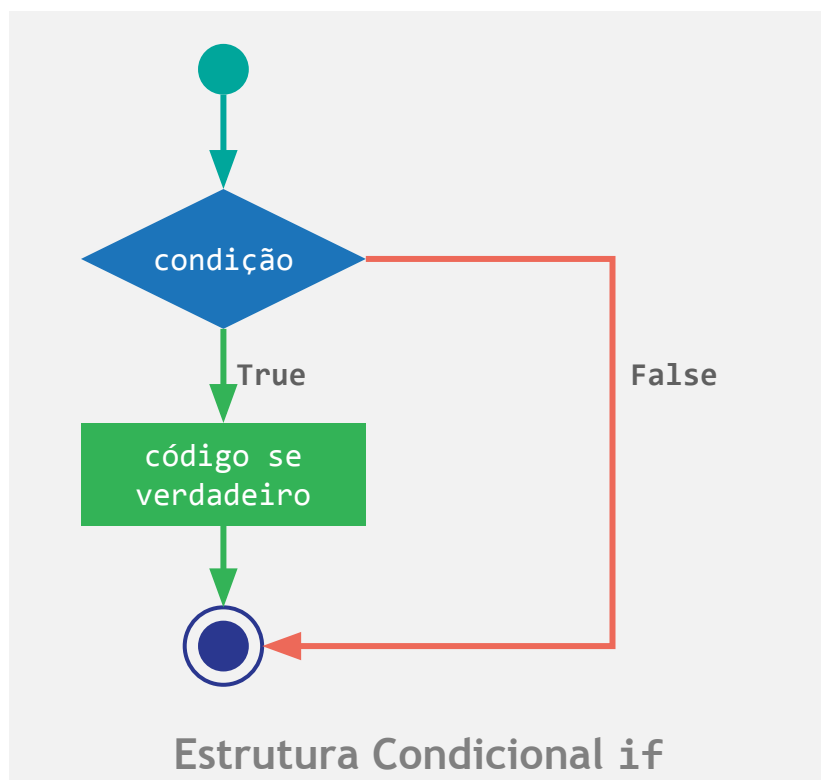


Conteúdo da Aula

- Estruturas Condicionais
- Estrutura Condicional `if`
- Estrutura Condicional `if - else`
- Estrutura Condicional `if - elif - else`

Estruturas Condicionais

- Situações em que é necessário executar um trecho de código específico com base em uma determinada condição.



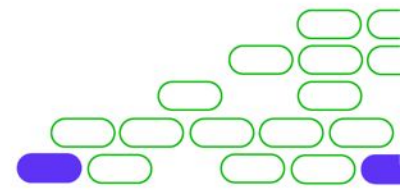


Conteúdo Prático

Conclusão



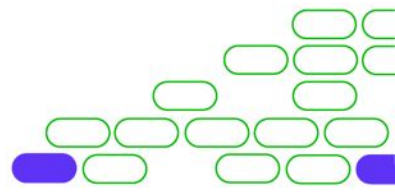
- ✓ Aprendemos como utilizar as estruturas condicionais



Próxima Aula



- Estruturas de Repetição





Faculdade

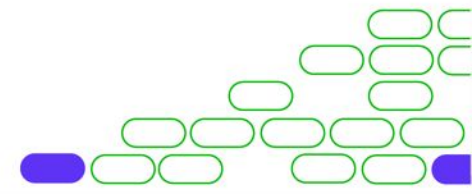


Fundamentos - Desenvolvedor Python

Capítulo 04 - Fluxos de Controle em Python

Aula 02 - Estruturas de Repetição

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula

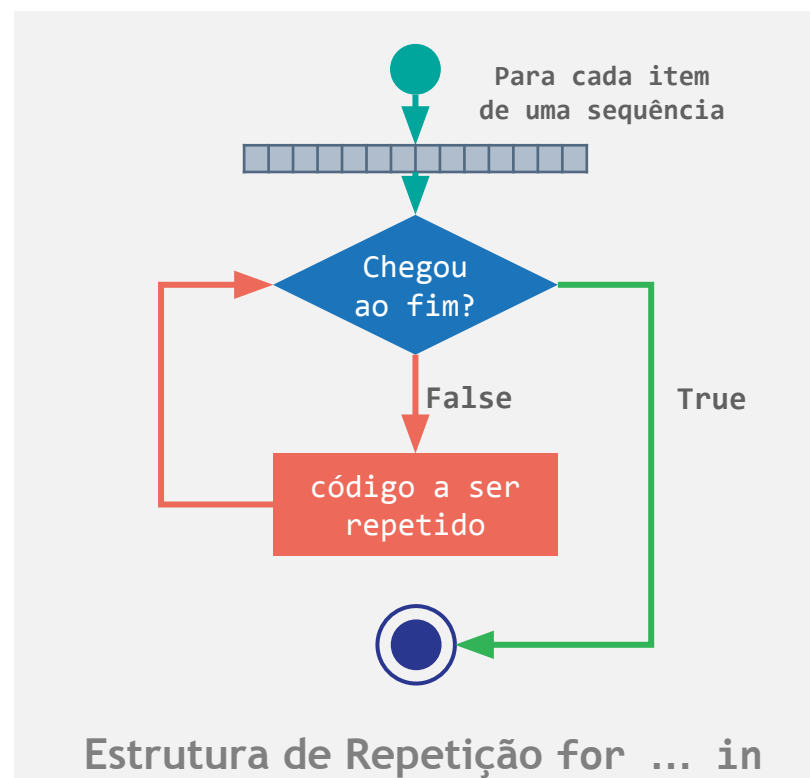
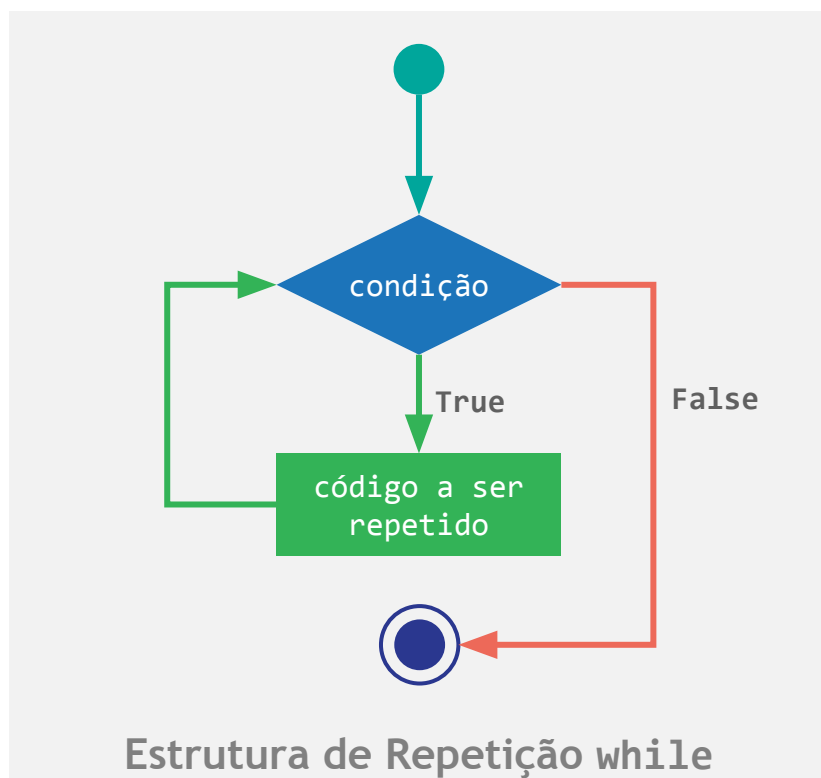


- Estruturas de Repetição
- Estrutura de Repetição **while**
- Estrutura de Repetição **for ... in**



Estruturas de Repetição

- Situações em que é necessário executar um trecho de código específico repetidas vezes.



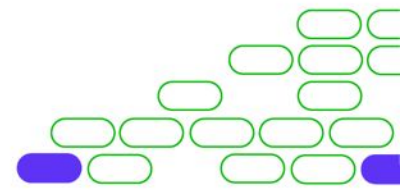
A black computer monitor with a silver stand, displaying the text 'Conteúdo Prático' on its screen.

Conteúdo Prático

Conclusão



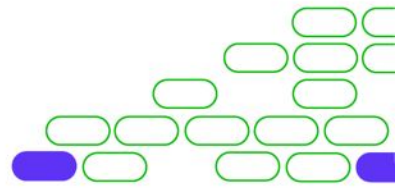
- ✓ Aprendemos como utilizar as estruturas de repetição



Próxima Aula



- Estruturas de Dados





Faculdade

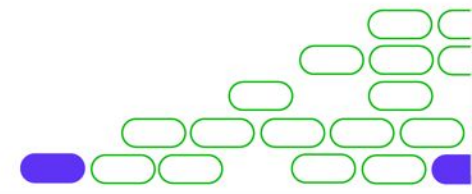


Fundamentos - Desenvolvedor Python

Capítulo 05 - Estruturas de Dados em Python

Aula 01 - Estruturas de Dados

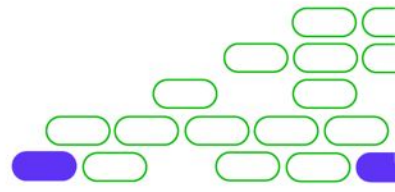
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Estruturas de dados

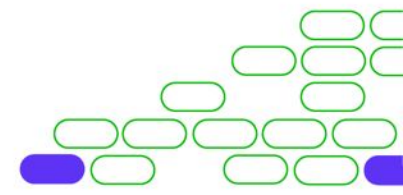


Estruturas de Dados

- Dada as idades de um grupo de pessoas, qual a idade da pessoa mais velha?

```
>>> # cria as variáveis com as idades
... p1 = 27
... p2 = 49
... p3 = 12
...
... # verifica qual a maior idade
... if p1 >= p2: 1
...     if p1 >= p3: 2
...         print('Maior idade:', p1)
... elif p2 >= p3: 3
...     print('Maior idade:', p2)
... else:
...     print('Maior idade:', p3)
Maior idade: 49
```

3 Pessoas
3 Comparações

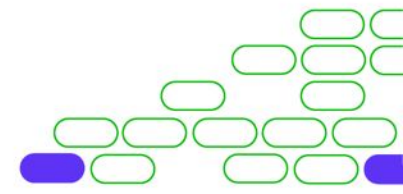


Estruturas de Dados

- Dada as idades de um grupo de pessoas, qual a idade da pessoa mais velha?

```
if p1 >= p2: 1
    if p1 >= p3: 2
        if p1 >= p4: 3
            print('Maior idade:', p1)
        else:
            print('Maior idade:', p4)
    elif p3 >= p4: 4
        print('Maior idade:', p3)
    else:
        print('Maior idade:', p4)
if p2 >= p3: 5
    if p2 >= p4: 6
        print('Maior idade:', p2)
    else:
        print('Maior idade:', p4)
elif p3 >= p4: 7
    print('Maior idade:', p3)
else:
    print('Maior idade:', p4)
```

4 Pessoas
7 Comparações



Estruturas de Dados

- Dada as idades de um grupo de pessoas, qual a idade da pessoa mais velha?

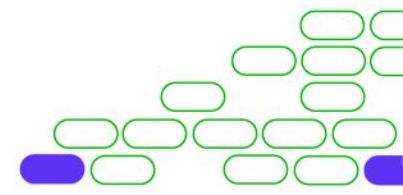
Quantidade de Pessoas	Número de Comparações
3	3
4	7
5	15
6	31
10	511
20	524.287
35	17.179.869.183

Se conseguíssemos escrever
uma comparação por segundo
seriam necessários 550 ANOS!

Estruturas de Dados

- Dada as idades de um grupo de pessoas, qual a idade da pessoa mais velha?

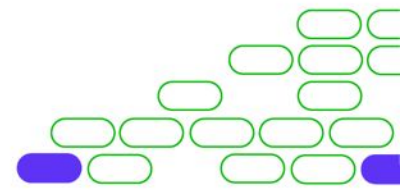
Estrutura	Tipo	Exemplo	Características
Lista	<code>list</code>	<code>[1, 2, 3, 4]</code>	Ordenada e mutável.
Tupla	<code>tuple</code>	<code>(1, 2, 3, 4)</code>	Ordenada e imutável.
Conjunto	<code>set</code>	<code>{1, 2, 3, 4}</code>	Não-ordenada, mutável e valores únicos.
Dicionário	<code>dict</code>	<code>{'a':1, 'b':2, 'c':3}</code>	Mapeamento (key, value), não-ordenado e mutável.



Conclusão



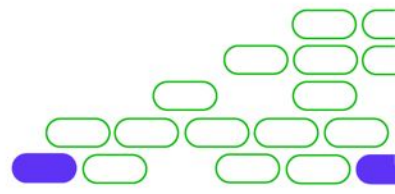
- ✓ Aprendemos a importância das estruturas de dados



Próxima Aula



- Tipo lista (`list`)





Faculdade

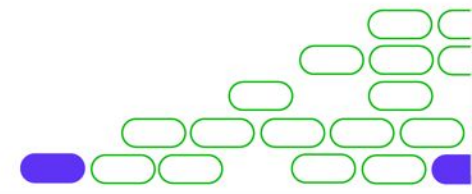


Fundamentos - Desenvolvedor Python

Capítulo 05 - Estruturas de Dados em Python

Aula 02 - Listas

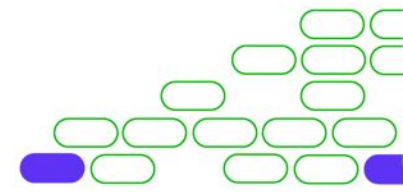
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Tipo lista (`list`)



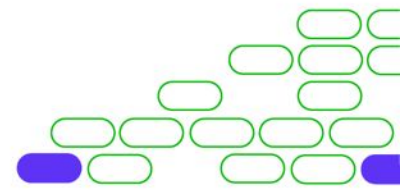


Conteúdo Prático

Conclusão



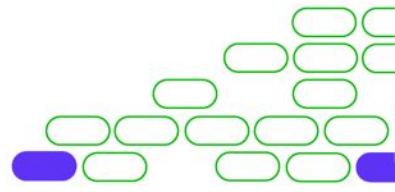
- ✓ Criação e operações do tipo lista
- ✓ Aplicações práticas do tipo lista



Próxima Aula



- Tipo tupla (tuple)





Faculdade

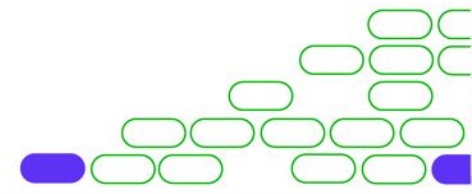


Fundamentos - Desenvolvedor Python

Capítulo 05 - Estruturas de Dados em Python

Aula 03 - Tuplas

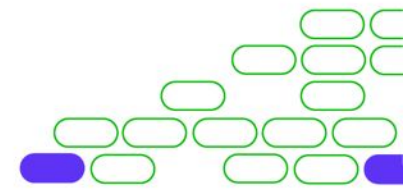
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Tipo tupla (tuple)



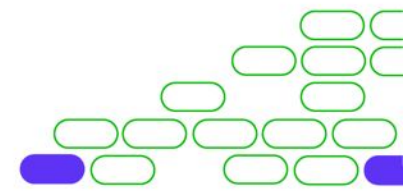


Conteúdo Prático

Diferenças entre Listas e Tuplas

- Em seu livro¹, Mark Pilgrim enumera três pontos interessantes sobre tuplas:
- *Tuplas são mais rápidas que listas - Se você está definindo uma sequência constante de valores e você vai ter que iterar sobre ele, utilize uma tupla ao invés de uma lista.*
- *Tuplas tornam o seu código mais seguro - Uma vez que eles protegem, contra gravações, os dados que não precisam ser alterados. Usar uma tupla em vez de uma lista é como ter uma declaração implícita de que esses dados são constantes e que uma função específica será necessária para sobrescrevê-los.*
- **Tuplas podem ser utilizadas como chaves de dicionários** - As listas nunca podem ser utilizadas como chaves de dicionário, porque as listas não são imutáveis."

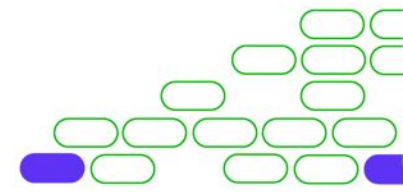
1 - PILGRIM, M. Dive Into Python 3. 2. ed. [S.l.]: Apress, 2009. 360 p. ISBN 1430224150.



Conclusão



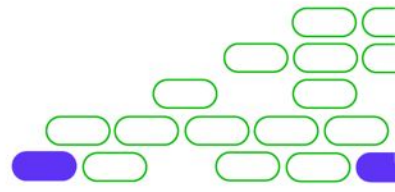
- ✓ Criação e operações do tipo tupla
- ✓ Diferença entre tuplas e listas



Próxima Aula



- Tipo conjunto (set)





Faculdade

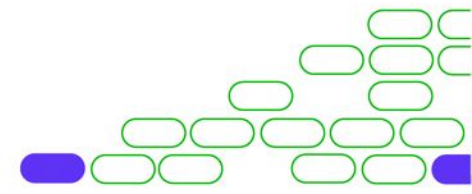


Fundamentos - Desenvolvedor Python

Capítulo 05 - Estruturas de Dados em Python

Aula 04 - Conjuntos

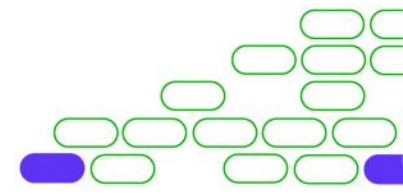
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



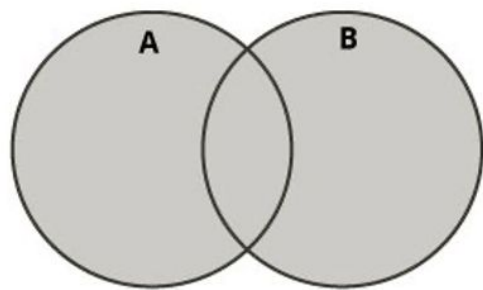
- Tipo conjunto (set)



Operações com Conjuntos

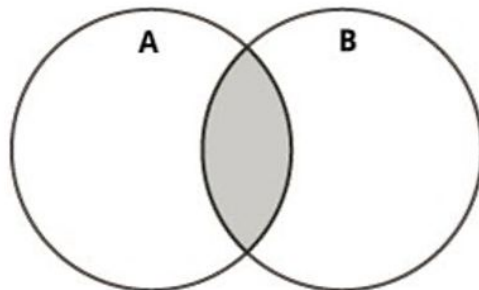
- Antes da prática, vamos relembrar alguns conceitos sobre conjuntos

União - $A \cup B$



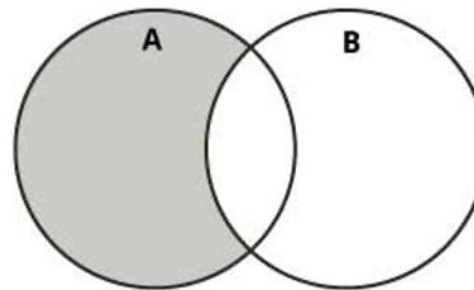
O resultado será o conjunto formado por todos os elementos de **A** e **B**

Interseção - $A \cap B$

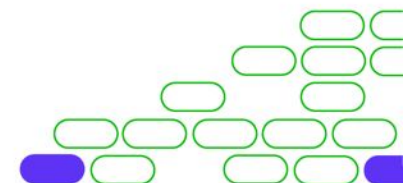


O resultado será o conjunto formado apenas pelos elementos que estejam em **A** e **B**, simultaneamente

Diferença - $A - B$



O resultado será o conjunto formado pelos elementos que estejam em **A**, mas que não estejam em **B**





Conteúdo Prático

Conclusão



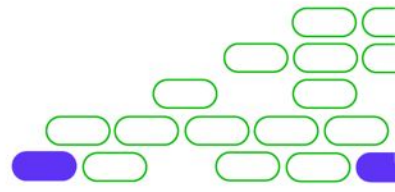
- ✓ Criação e operações do tipo conjunto
- ✓ Aplicações práticas do tipo conjunto



Próxima Aula



- Tipo dicionário (dict)





Faculdade

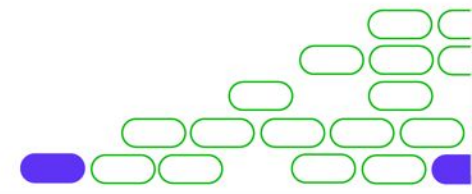


Fundamentos - Desenvolvedor Python

Capítulo 05 - Estruturas de Dados em Python

Aula 05 - Dicionários

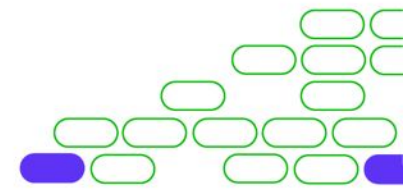
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Tipo dicionário (dict)





Conteúdo Prático

Conclusão

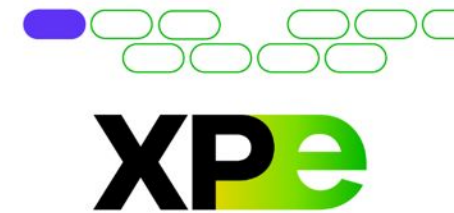


- ✓ Criação e operações do tipo dicionário
- ✓ Aplicações práticas do tipo dicionário



Próxima Aula

- Funções





Faculdade

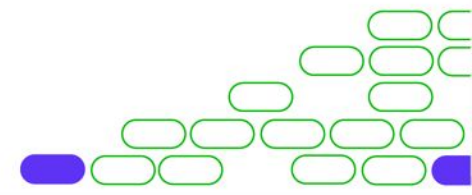


Fundamentos - Desenvolvedor Python

Capítulo 06 - Funções em Python

Aula 01 - Declaração de Funções

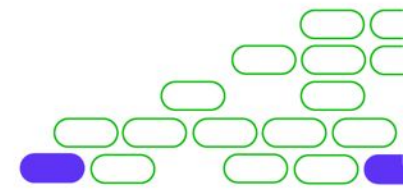
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Importância das Funções
- Reusabilidade de Código
- Declaração de Funções



Reutilização de Códigos

- Durante o desenvolvimento, será necessário utilizar uma mesma computação múltiplas vezes, em diferentes partes do código.
- Operações semelhantes em um programa de gerenciamento empresarial
 - **Soma:** Vendas diárias, pagamentos aos fornecedores, gastos com frete etc.
 - **Ordenação:** pagamentos por vencimento, clientes por região e nome etc.
- O ideal é que tenhamos um procedimento capaz de somar uma série de valores e um outro que ordene os itens de uma lista
 - Estes procedimentos podem ser utilizados diversas vezes, sem a necessidade de reescrever o mesmo código
- **Ou seja, desejamos ter partes do código que sejam reutilizáveis!**



Funções em Python

- Em Python, assim como em outras linguagens de alto nível, a reutilização de códigos é realizada por meio de **funções**
 - Uma função é basicamente um bloco de código, que realiza uma determinada tarefa e que pode ser reutilizado várias vezes
 - As funções auxiliam na divisão do programa em partes menores e modulares
- Nós já utilizamos funções diversas vezes nos capítulos anteriores:
 - `print()`: para exibição/impressão de valores
 - `range()`: que gera uma sequência de inteiros
 - `len()`: que retorna o tamanho de uma sequência
- **Agora vamos aprender a criar nossas próprias funções!**





Conteúdo Prático

Conclusão



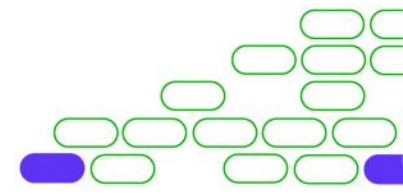
- ✓ Aprendemos sobre reusabilidade de código e funções
- ✓ Criamos nossas primeiras funções



Próxima Aula



- Utilização das funções
- Conceitos importantes sobre funções em Python





Faculdade

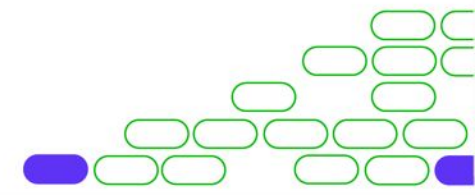


Fundamentos - Desenvolvedor Python

Capítulo 06 - Funções em Python

Aula 02 - Utilização de Funções

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Aplicação das funções no nosso código
- Conceitos importantes sobre funções





Conteúdo Prático

Conclusão



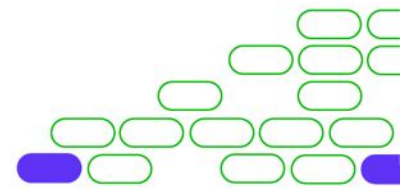
- ✓ Aprendemos a utilizar nossas funções
- ✓ Vimos conceitos importantes sobre funções



Próxima Aula



- Argumentos das Funções





Faculdade

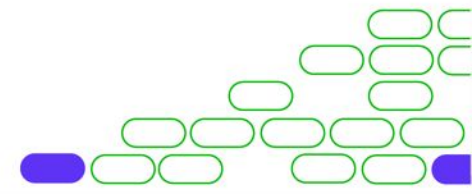


Fundamentos - Desenvolvedor Python

Capítulo 06 - Funções em Python

Aula 03 - Argumentos das Funções

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Declaração explícita de argumentos
- Argumentos padrão (*default*)





Conteúdo Prático

Conclusão



- ✓ Aprendemos a utilizar diferentes maneira de declarar argumentos



Próxima Aula

- Módulos





Faculdade

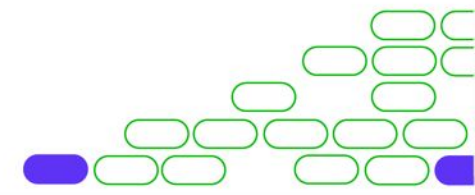


Fundamentos - Desenvolvedor Python

Capítulo 07 - Módulos

Aula 01 - Criação e Importação de Módulos

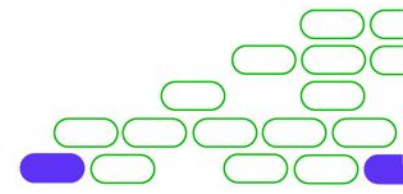
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Entender o que são os módulos em Python
- Criar e importar nossos próprios módulos



Módulos em Python

- Um módulo em Python nada mais é do que um declarações de variáveis e ou funções.
 - Esses arquivos devem ser salvos com extensão `.py`
- Nós utilizamos módulos para poder organizar (modularizar) os nossos códigos de acordo com as funcionalidades de cada conjunto de funções.
 - Imaginem um programa para geometria, poderíamos ter:
 - Um módulo para cada figura geométrica (quadrado, triângulo, círculo etc.)
 - Ou um módulo para cada tipo de cálculo geométrico (área, volume, etc.)
- A organização e a divisão em módulos dependerão do seu objetivo
- São essenciais para a modularização dos programas
 - Principalmente aqueles com milhares de linhas de código.



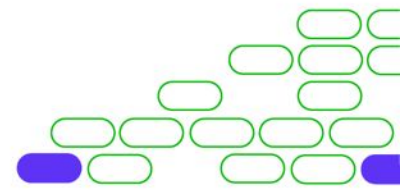


Conteúdo Prático

Conclusão



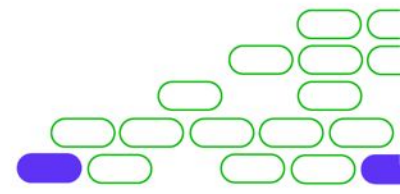
- ✓ Aprendemos a criar e importar nosso próprios módulos



Próxima Aula



- Principais módulos embutidos do Python





Faculdade

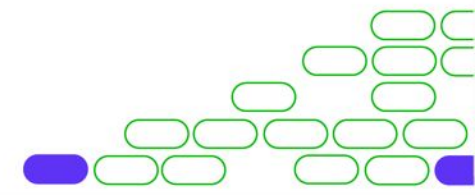


Fundamentos - Desenvolvedor Python

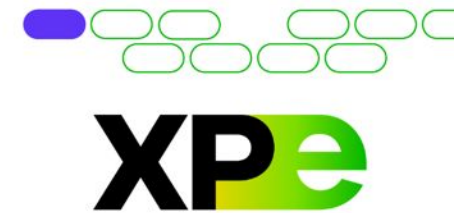
Capítulo 07 - Módulos

Aula 02 - Módulos Embutidos do Python

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Conhecer os principais módulos embutidos do Python
- Utilizar estes módulos para resolver situações práticas



Principais Módulos Embutidos do Python

- Nativamente, o Python oferece mais de 150 módulos

Módulo	Funcionalidades
<code>collections</code>	Estruturas de dados com diferentes funcionalidades
<code>csv</code>	Manipulação de arquivos CSV
<code>datetime</code>	Manipulação de datas e timestamps
<code>json</code>	Manipulação de arquivos json
<code>math</code>	Funções matemáticas
<code>multiprocessing</code>	Possibilita o processamento paralelo
<code>os</code>	Funções de interação com o sistema operacional
<code>random</code>	Geração de dados aleatórios
<code>sys</code>	Funções e parâmetros específicos do sistema



Conteúdo Prático

Conclusão



- ✓ Conhecemos e utilizamos alguns módulos embutidos do Python



Próxima Aula



- Instalação de novos módulos providos por terceiros



Faculdade

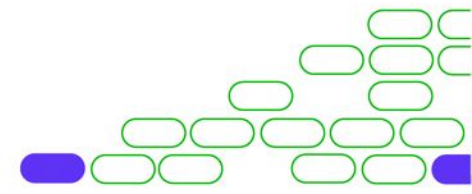


Fundamentos - Desenvolvedor Python

Capítulo 07 - Módulos

Aula 03 - Instalação de Novos Módulos

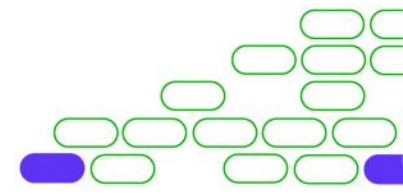
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Como instalar e importar novos módulos do Python



Módulos Externos do Python

- Qualquer desenvolvedor pode criar e disponibilizar módulos pra outras pessoas
- Os módulos são disponibilizados por meio de pacotes
- Repositório oficial do Python: pypi
 - 360 mil módulos!
 - Verificar sempre a procedência e popularidade do módulo
- Também existe os repositórios do conda
- O Google Colab já possui os principais módulos instalados, principalmente aqueles que são utilizados pelos profissionais de IA e Dados



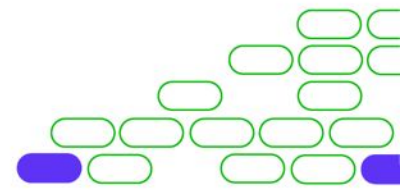


Conteúdo Prático

Conclusão



- ✓ Aprendemos a instalar e utilizar módulos desenvolvidos por terceiros



Próxima Aula



- Manipulação de arquivos em Python





Faculdade

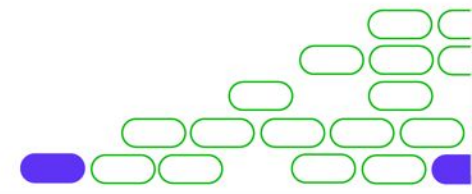


Fundamentos - Desenvolvedor Python

Capítulo 08 - Manipulação de Arquivos

Aula 01 - Criação, Abertura e Fechamento de Arquivos

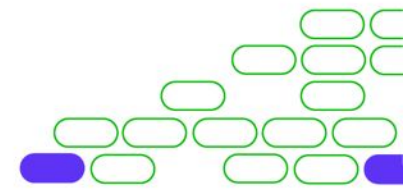
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Como criar arquivos em disco utilizando o Python



Arquivos em Disco

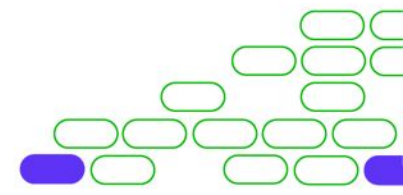
- Até o momento trabalhamos apenas com dados diretamente na memória do computador
- Em muitas situações teremos que ler e gravar dados que estarão armazenados em um arquivo do disco
- Existem diferentes formatos de arquivos como *CSV*, *JSON*, *XML*, *YML* e vários outros.
 - Para cada um destes formatos, o ideal é utilizar um pacote Python desenvolvido especificamente para a manipulação de seus dados.
- O conteúdo aqui apresentado é apenas introdutório, para que possamos entender os conceitos básicos



Criação e Abertura de Arquivos

- A criação e de arquivos é realizada por meio da função:
 - `open(caminho, modo)`
- Existem diferentes modos de uso, conforme a tabela

Modo	Descrição
'r'	Modo somente leitura (modo padrão).
'w'	Modo de escrita. Cria um arquivo, caso ainda não exista, ou substitui o arquivo atual.
'x'	Modo de escrita. Cria um arquivo e, se o arquivo já existir, retorna um erro.
'a'	Modo de escrita. Cria um arquivo, caso ainda não exista e adiciona dados ao final dele.
't'	Abre o arquivo no modo texto (modo padrão).
'b'	Abre o arquivo no modo binário.





Conteúdo Prático

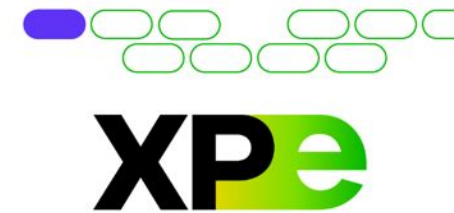
Conclusão



- ✓ Aprendemos a criar/abrir e fechar arquivos em Python



Próxima Aula



- Leitura de arquivos





Faculdade

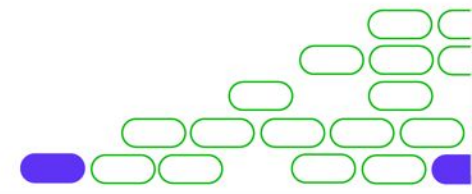


Fundamentos - Desenvolvedor Python

Capítulo 08 - Manipulação de Arquivos

Aula 02 - Leitura de Arquivos

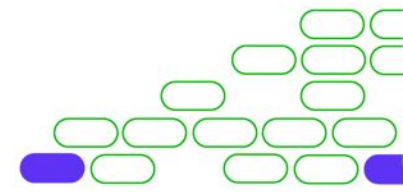
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Realizar a leitura de arquivos texto



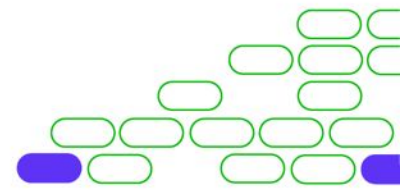


Conteúdo Prático

Conclusão



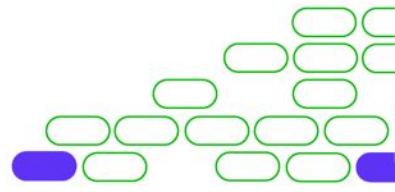
- ✓ Aprendemos a realizar a leitura de arquivos texto em Python



Próxima Aula



- Escrita de arquivos





Faculdade

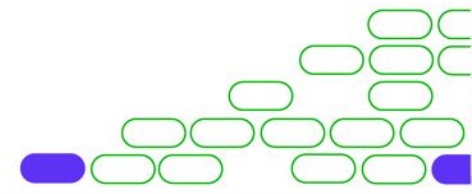


Fundamentos - Desenvolvedor Python

Capítulo 08 - Manipulação de Arquivos

Aula 03 - Escrita de Arquivos

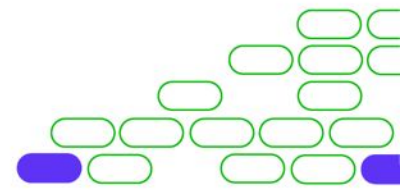
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Realizar a gravação de dados em arquivos texto





Conteúdo Prático

Conclusão



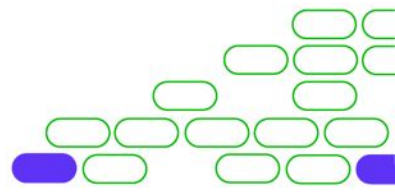
- ✓ Aprendemos a realizar a gravação de dados em arquivos texto com Python



Próxima Aula



- Recursos úteis da linguagem Python





Faculdade

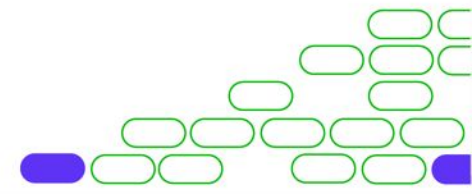


Fundamentos - Desenvolvedor Python

Capítulo 09 - Recursos Úteis da Linguagem

Aula 01 - Compreensão de Listas

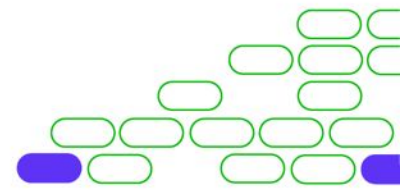
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Aprenderemos um recurso muito útil do Python, a compreensão de listas



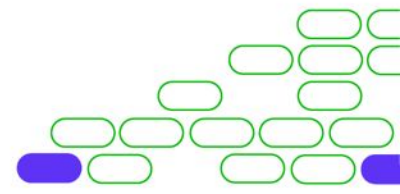


Conteúdo Prático

Conclusão



- ✓ Aprendemos a realizar e aplicar a compreensão de listas



Próxima Aula



- Compreensão de Dicionários





Faculdade

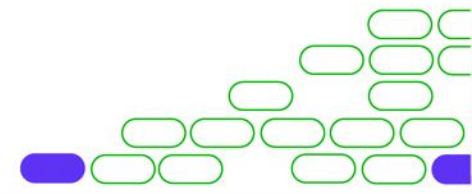


Fundamentos - Desenvolvedor Python

Capítulo 09 - Recursos Úteis da Linguagem

Aula 02 - Compreensão de Dicionários

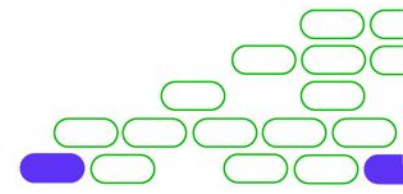
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Aprenderemos outro recurso útil do Python, a compreensão de dicionários



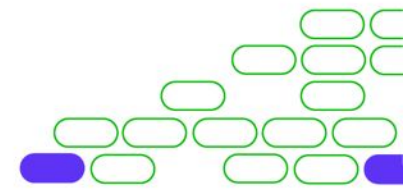


Conteúdo Prático

Conclusão



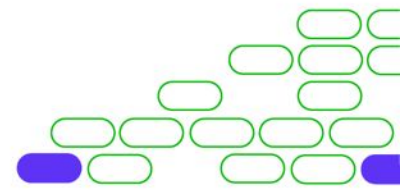
- ✓ Aprendemos a realizar e aplicar a compreensão de dicionários



Próxima Aula



- Funções Anônimas (lambda Functions)





Faculdade

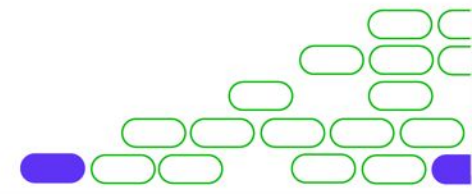


Fundamentos - Desenvolvedor Python

Capítulo 09 - Recursos Úteis da Linguagem

Aula 03 - Funções Anônimas (Funções *Lambda*)

Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



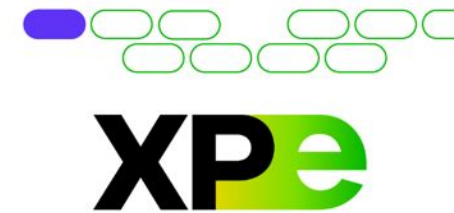
- Aprenderemos o conceito de funções *lambda* em Python
- Aplicaremos funções *lambda* em exemplos





Conteúdo Prático

Conclusão



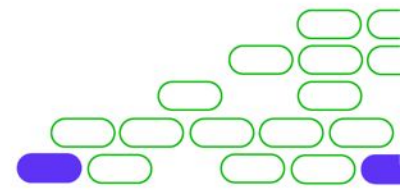
- ✓ Aprendemos a utilizar funções *lambdas* como recurso em Python



Próxima Aula



- Atribuição Condicional em Uma Linha





Faculdade

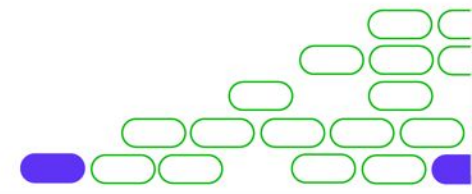


Fundamentos - Desenvolvedor Python

Capítulo 09 - Recursos Úteis da Linguagem

Aula 04 - Atribuição Condicional em Uma Linha

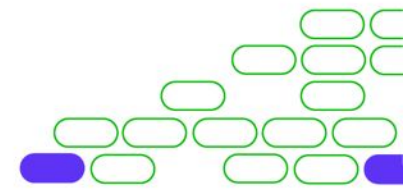
Antônio Carlos de Nazaré Júnior



Conteúdo da Aula



- Aprenderemos como realizar a atribuição condicional



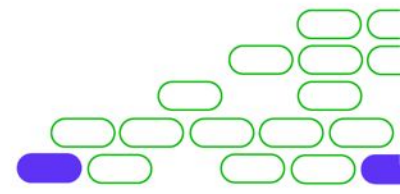


Conteúdo Prático

Conclusão



- ✓ Aprendemos a utilizar a atribuição condicional



Agora é com vocês...
É hora de voar com o Python!!!

