
JORNADA PYTHON

FAIXA PRETA



ByLearn

Seja Bem-Vindo!



Felipe Cabrera

Graduado em Ciência da Computação e
Mestrando em Computação Aplicada – FFCLRP –
USP.

RESUMO DA AULA 1



CIÊNCIA DA COMPUTAÇÃO

Programação

Escrever um *código* que será *transformado* em um *programa*.

Algoritmo

A *sequência de instruções* lógicas que chegam a *solução de um problema*.

Linguagem de Programação

Conjunto de *regras, padrões* e *instruções* para comunicarmos com a *máquina* e gerarmos um *software*.

Código

A *implementação* de um *algoritmo* usando uma *linguagem de programação*.

PROGRAMAÇÃO

Compilar

Traduzir o código *escrito* (fonte) para o código lido pela *máquina* (alvo).

Interpretar

O código escrito é passado para uma *linguagem intermediária* e executada pelo *interpretador*.


Python: O que é

Linguagem de Programação interpretada criada por Guido van Rossum.

Python: Motivos

A que *mais cresce*, tem como objetivo ser *fácil*, *simples*, *gratuita* e *poderosa*. Alta *demanda* de empregos, *versátil* e com bons *salários*.

RESUMO DA AULA 2

The background of the slide is a teal-to-blue gradient. On the right side, there is a complex network of light blue dots connected by thin lines, creating a web-like or molecular structure that extends from the top right towards the bottom right.

PREPARAÇÃO

Instalação

Você pode instalar o *Python* através do *site oficial* ou através do *pacote Anaconda*.

Google Colab

Uma solução para utilizar o *Python na nuvem*!

O uso é totalmente online!

IDE

É um *programa* que *auxilia* na hora do *desenvolvimento* (programação)

IDE Escolhida

Nós escolhemos o *Google Colab*, para *ninguém precisar instalar* o Python durante a Jornada.

PYTHON

Sintaxe

Basicamente, a *sintaxe* está ligada na *forma como escrevemos*. Ela quem dita a *maneira correta* de se escrever

Sintaxe na Programação

Cada linguagem de programação possui a *sua sintaxe*, ou seja, a *sua forma de escrever*.

Pythonês

O Python é lido *da esquerda pra direita* e *de cima pra baixo*, com sintaxe *igual a de um texto normal*

Identação

No Python o *nível mais a direita pertence* ao *nível à sua esquerda*.

BÁSICO DA PROGRAMAÇÃO

Variáveis

Uma variável é um *identificador* que se *refere a um valor*

Tipos de Dados

Cada *dado* irá possuir um *tipo*.

É isso que definirá se ele é um *número*, *texto*, *booleano*...

Métodos de Entrada

São formas de *enviar dados do usuário* ao Python.

Métodos de Saída

São formas do Python de *mostrar dados ao usuário*.

BÁSICO DA PROGRAMAÇÃO

Dados Sequenciais

Para trabalharmos com valores sequenciais *podemos usar listas*

Elementos em Listas

Quando queremos usar um elemento da lista, *devemos utilizar o seu índice.*

Índices x Posições

Posições iniciam em 1. Já os índices, iniciam em 0.

Basicamente:
Índice = Posição - 1

Condições

Formas de realizar *tomadas* lógicas *de decisão.*

Se... Então
Senão, se... Então
Senão... Então

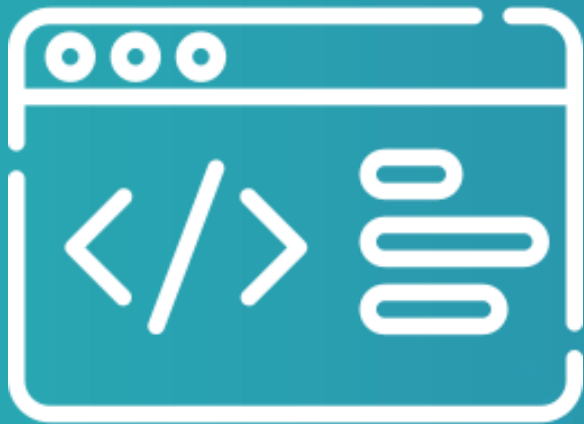
GITHUB

O que é o *GitHub*

É a principal plataforma para *hospedagem* e *compartilhamento* de *códigos* (de programação) do mundo!

Ótimo para *encontrar* projetos e *estudar*





O INTERMEDIÁRIO DA PROGRAMAÇÃO

*Bora continuarmos a aprender a
dominar o Python?*

Operadores Lógicos

Podemos usar operadores lógicos para criar *condicionais compostas*.

Por exemplo: Verificar se é de manhã *e* está chovendo ou se é sábado *ou* domingo.

Podemos usar *and* para que *todas* sejam verdade.

Podemos usar *or* para que *ao menos* uma seja verdade.

```
dia = 'sabado'
horario = 'manha'
chovendo = False

if horario == 'manha' and chuvendo == False:
    print("O sol está lindo")

if dia == 'sabado' or dia == 'domingo':
    print("É fim de semana")
```

Laços de Repetição

Excelente forma de evitar repetição de código.

Por exemplo: Mostrar todos os números até 10.

Para usar laços de repetição nós podemos usar o *for*.

PS: Como na programação iniciamos em 0, *range(10)* vai de 0 até 9.

```
for numero in range(10):  
    print(numero)  
    # De 0 até 9  
  
for numero in range(1,11):  
    print(numero)  
    # De 1 até 10
```

Tamanho de Listas

Podemos também pegar o tamanho de listas e usá-los no nosso código.

Por exemplo: Pegar quantos nomes tem na lista e mostrar um a um (pelo índice).

Sabermos o tamanho de uma variável torna nosso código mais dinâmico.

No Python nós usamos a função *len* para pegar o tamanho da sequência.

```
nomes = ['Felipe', 'Fabiana', 'Paulo', 'Katia']  
  
tamanho = len(nomes)  
  
for indice in range(tamanho):  
    print("O nome no índice", indice, "é", nomes[indice])
```

Repetições Dinâmicas

Além de usar o tamanho da variável, podemos também usar a própria sequência na repetição.

Por exemplo: Para todas as *espécies* da lista *animais* eu quero *printar* a *espécie*

Para isso, usamos o *for* com a *sequência* que queremos trabalhar, como:

for elemento in sequencia:

```
animais = ['Gato', 'Furão', 'Cachorro', 'Coelho', 'Pássaro', 'Peixe']

for animal in animais:
    print("Espécie:", animal)

numeros = [5, 10, 15, 20, 25, 30]

for numero in numeros:
    print("O valor", numero, "é múltiplo de 5!")
```


Funções

Outra forma de evitar repetição de código é através de funções, onde definimos uma rotina para ser executada.

Por exemplo: Definir o que fazer para verificar se um aluno foi aprovado.

Para definirmos uma função usamos o *def*.

Funções podem *retornar valores*.

Podemos *enviar valores* para funções.

```
# Definimos a funcao da media
def calcular_media(nota1, nota2):
    soma = nota1 + nota2
    media = soma / 2

    return media

# Definimos a função da Aprovacao
def verificar_aprovacao(media):

    if media >= 6:
        print('O Aluno foi aprovado')
    else:
        print('O Aluno foi reprovado')

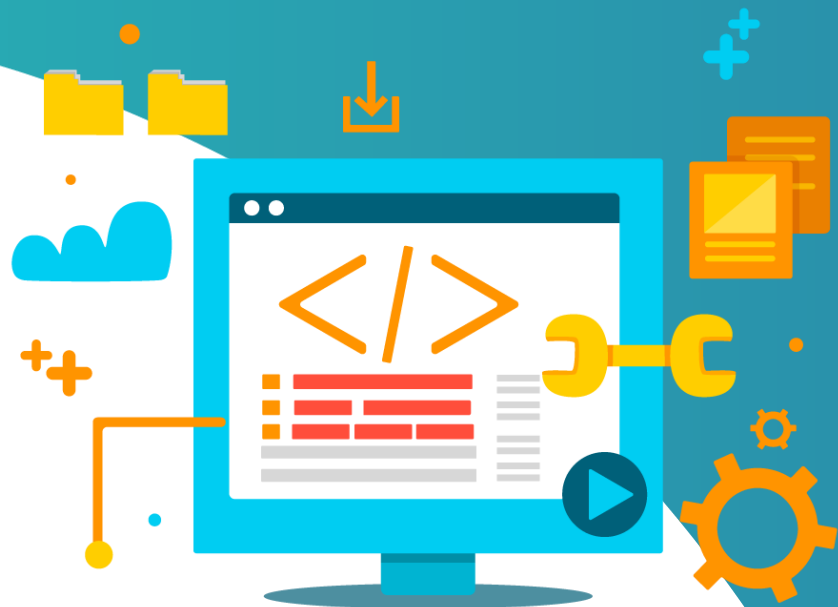
# Chamamos a função Calcular Média
# Enviamos as notas 7.75 e 4.5
# Pegamos o valor do seu retorno...
# ... e colocamos em media_aluno
media_aluno = calcular_media(7.75, 4.5)

# Chamamos a função Verificar Aprovação
# Enviamos a média calculada acima
verificar_aprovacao(media_aluno)
```



GIT

GIT



- É o *controlador de versões de código* mais utilizado no mundo!
- Basicamente, no lugar de “*projeto_v1*”, “*projeto_v2*”, “*projeto_v3*”...
- Nós usamos o *Git* para manter um projeto (*repositório*) só.
- Todas as *versões* ficam em *histórico estudar*.

GIT

Ferramenta que permite o controle de versões de código.

GITHUB

É uma plataforma (serviço) que permite o uso do GIT.



CRIANDO UM REPOSITÓRIO

Criar um repositório no Github é um processo extremamente simples e que não demora um minuto!

Saiba mais clicando [aqui](#).



EXEMPLOS PRÁTICOS

The background is a teal-to-blue gradient. On the right side, there is a complex network of light blue dots connected by thin lines, creating a web-like or molecular structure that extends from the top right towards the bottom right.



```
nota1 = 7.5
nota2 = 4.8

# Definimos o que é "Verificar uma Aprovação"
def verificar_aprovacao():
    media = calcular_media([nota1, nota2])

    if media >= 6:
        print("O Aluno Foi Aprovado!")
    else:
        print("O Aluno Foi Reprovado")


# Definimos o que é "Calcular a Média"
def calcular_media(notas):
    quantidade = len(notas)

    soma = 0
    for nota in notas:
        soma = soma + nota

    media = soma / quantidade

    return media

# Chamamos (executamos) a função de Verificar Aprovação
verificar_aprovacao()
```



```
frutas = ['Maça', 'Banana', 'Pera', 'Uva']
guloseimas = ['Bolacha', 'Batata', 'Fini', 'Chocolate']
comidas = ['Arroz', 'Feijão', 'Carne']
bebidas = ['Refrigerante', 'Suco de Laranja', 'Água']

categorias = ['Frutas', 'Guloseimas', 'Comidas', 'Bebidas']
compras = [frutas, guloseimas, comidas, bebidas]

for indice, categoria in enumerate(categorias):
    print('Você precisa comprar', len(compras[indice]), categoria+':')
    for compra in compras[indice]:
        print('-', compra)
```



```

def validar_idade(idade):
    if idade < 18:
        print('\nDesculpe, você não tem idade para prosseguir,', nome)
        return False
    else:
        print('\nÓtimo! Podemos prosseguir,', nome)
        return True

def escolher_carta():
    print("Digite uma das opções abaixo:")
    print("1 - Carro\n2 - Moto\n3 - Carro e Moto")

    return int(input())

def calcular_preco(escolha):
    valor_carro = 1500
    valor_moto = 1000

    if escolha == 1:
        return valor_carro
    elif escolha == 2:
        return valor_moto
    else:
        return valor_carro + valor_moto

def desconto(valor):
    return valor - (valor * 0.10)

nome = input('Digite o seu nome: ')
idade = int(input('Digite sua idade'))

if validar_idade(idade):
    escolha = escolher_carta()

    print('\nPerfeito! Vou calcular o valor')
    valor = calcular_preco(escolha)

    print('\n'+nome, 'o valor total é de', valor, 'reais')
    print('Mas vou ver com meu gerente se posso dar um desconto...')
    valor = desconto(valor)

    print('\nCom desconto eu consigo fazer por', valor, 'reais.')

    print('Te interessa?\n1 - Sim\n2 - Não')
    interesse = int(input())
    if interesse == 1:
        print('\nPerfeito! Começaremos amanhã!')
    else:
        print('\nTudo bem :(Me avise se mudar de ideia.')

```



```
animais = []

animal = input("Digite o nome do seu animal de estimação ou digite 0 se não tiver nenhum: ")

while animal != '0':
    especie = input("Digite a Espécie desse animal: ")
    animais.append([animal, especie])
    animal = input('Se tiver mais animais, digite o nome dele. Ou digite 0 se não tiver: ')

if len(animais) == 0:
    print('\n\nVocê não tem animais')
else:
    print("\n\nVocê tem os seguintes animais:")
    for animal in animais:
        print("- Nome:", animal[0], "| Espécie:", animal[1])
```



```
horario = 'manhã'
clima = 'ensolarado'
temperatura = 'quente'

if horario == 'manhã' or horario == 'tarde':
    if clima == 'ensolarado' and temperatura == 'quente':
        print("Uma piscina cairia bem")

    if (clima == 'ensolarado' or clima == 'nublado') and (temperatura == 'amena' or temperatura == 'frio'):
        print("Seria legal praticar algum esporte")

    if clima == 'chuvoso':
        print("Aproveite para treinar seu Python")
else:
    if clima == 'chuvoso':
        print("Que tal um filme, série ou jogatina?")
    else:
        print("Um jantar fora parece interessante...")
```



```
nome = input("Insira seu nome: ")

nota1 = float(input("Sua primeira nota: "))
nota2 = float(input("Sua segunda nota: "))

print("Olá", nome, "suas notas foram:")
print(nota1, "e", nota2)

soma = nota1 + nota2
media = soma / 2

print("Sua média é", media)
```

Saída:

*Olá Felipe suas notas foram:
8.25 e 7.75
Sua média é 8.0*

BÔNUS: O *PIP* E OS PACOTES EXTERNOS





PIP – O Que é

O PIP é o instalador de pacotes padrão do Python. Ele te permite instalar pacotes externos para auxiliar no seu desenvolvimento.

Você pode buscar esses pacotes no [Pypi](https://pypi.org/).

Pacotes do Python

Em resumo, todo arquivo (script) do Python é um *módulo*, ou seja, esses códigos que nós desenvolvemos fazem parte de um módulo.

Já o *projeto* como um todo, a junção de todos arquivos, é um *pacote*.

Ou seja... Uma *solução* que fizemos no Python, como uma de *Gerenciar Alunos* pode ser considerado um pacote Python, no caso, posso chamá-lo de *schoolmanager*.

PS: Guardem bem esse nome!



Pacotes Internos

O Python por conta própria já tem *vários pacotes prontos*, que chamamos de *built-ins*.

Esses pacotes, assim como vimos agora pouco, são *soluções completas*, certo?

Dessa forma, se nós utilizarmos esses pacotes, provavelmente iremos *acelerar o nosso desenvolvimento*.

Afinal, tudo lá já está *pronto para ser usado*!

Basta *importarmos* o pacote/módulo interno.



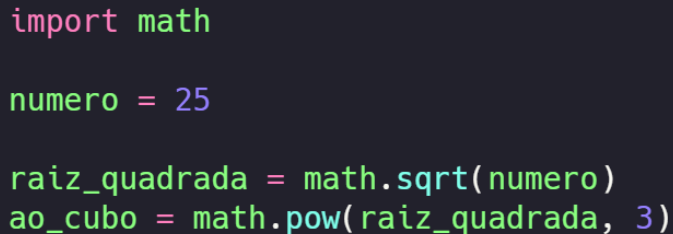
Importando

Nós podemos importar pacotes, módulos e funções no Python para facilitar nosso código.

Por exemplo: Calcular qual o cubo da raiz quadrada de um número? Importa o *math*!

No caso, usamos *import* para falar que estamos *importando* algo e na frente passamos o *nome*.

No caso, importamos o *math*.



```
import math

numero = 25

raiz_quadrada = math.sqrt(numero)
ao_cubo = math.pow(raiz_quadrada, 3)
```

Pacotes Externos

Assim como nós fazemos as *nossas próprias soluções*, muitos *outros programadores fazem as deles*, certo?

E se eu quiser *compartilhar* o *schoolmanager* com você? E se você quiser *compartilhar* sua solução comigo?

Como isso não vem por padrão no Python, nós chamamos esses pacotes de *pacotes externos*.

Ou seja: *Pacotes externos são soluções feitas por outros programadores*.



Instalando

Pacotes externos precisam ser *instalados* antes de ser *importados* no nosso código.

Para isso, usamos o *PIP*, lembra dele?

A sintaxe é simples:

pip install nome_do_pacote

Ex:

pip install schoolmanager



```
# Esse comando deve ser feito no terminal  
# No windows, você pode usar o CMD.exe
```

```
pip install nome_do_pacote
```

Exemplos:

```
pip install numpy  
pip install pandas  
pip install pillow  
pip install tensorflow  
pip install ...
```

Pacotes Externos

Sempre que você for iniciar um novo código é interessante buscar na internet se há algum pacote externo que pode te auxiliar.

Ex:

- Para requisições web, temos o *requests*.
- Para gerar pdf temos o *pdfkit*.
- Para criar gráficos temos o *matplotlib*.
- Para vetores e matrizes temos o *numpy*.
- Para imagens temos o *pillow (pil)*.



REQUESTS

```
# Importante o módulo do Requests e JSON
import requests
import json

# Definindo nosso Payload
payload = {'name': 'Felipe Cabrera', 'job': 'developer'}

# Enviando nosso Payload via POST
response = requests.post('https://reqres.in/api/users', data = json.dumps(payload))

# Vamos verificar o que enviamos
print(response.request.body)
# Output: {"name": "Felipe Cabrera", "job": "developer"}

# Vamos verificar o código de resposta
print(response.status_code)
# Output: 201

# Vamos verificar a resposta
print(response.text)
# Output: {"id":"683","createdAt":"2020-09-03T12:29:07.721Z"}
```

PDFKIT



```
import pdfkit

# PDF de um Site
pdfkit.from_url('http://google.com', 'out.pdf')

# PDF de um Arquivo
pdfkit.from_file('test.html', 'out.pdf')

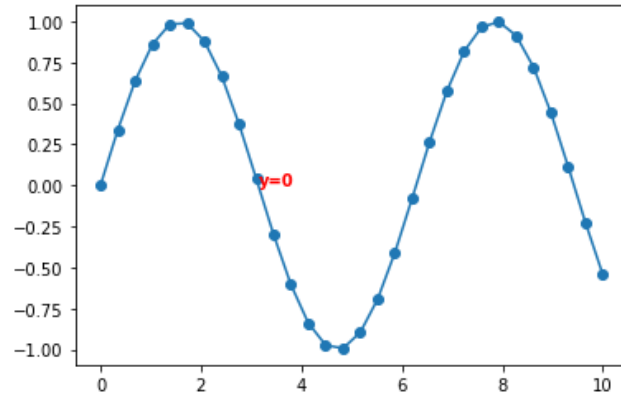
# PDF de um Texto
pdfkit.from_string('#ByLearnerFaixaPreta', 'out.pdf')
```

MATPLOTLIB



```
import matplotlib.pyplot as plt
import numpy as np

plt.plot(x, np.sin(x), '-o')
plt.text(np.pi, 0, 'y=0', weight='bold', c='r')
plt.show()
```



NUMPY

```
import numpy as np

# Criar matrizes randômicas
x1 = np.random.randint(10, size=6) # Matriz de uma dimensão
x2 = np.random.randint(10, size=(3, 4)) # Matriz bidimensional
x3 = np.random.randint(10, size=(3, 4, 5)) # Matriz tridimensional

print("Dimensões do x3: ", x3.ndim)
print("Shape (formato) do x3:", x3.shape)
print("Tamanho do x3", x3.size)

print('Primeira coluna do X2:', x2[:, 0])
print('Primeira linha do X2:', x2[0, :])
```


PILLOW (PIL)

```
from PIL import Image
from PIL import ImageFilter

# Manipula as imagens
imagem_original = Image.open('felipe.png')
imagem_com_filtro =
imagem_original.filter(ImageFilter.BLUR)
imagem_cinza = imagem_original.convert("LA")

# Mostra as imagens
imagem_original.show()
imagem_com_filtro.show()
imagem_cinza.show()

# Salva as imagens
imagem_original.save('original.png')
imagem_com_filtro.save('filtro.png')
imagem_cinza.save('cinza.png')
```



Original



Filtro



Cinza

EXTRA: RegEx

(Expressões Regulares)

Expressões Regulares



“Expressões Regulares são padrões utilizados para selecionar combinações de caracteres em uma string.”

De uma forma mais simples, podemos falar que as Expressões Regulares *são strings especiais para buscar padrões em textos.*

Algo como um *CTRL+F bem poderoso!*

Expressões Regulares



Seu uso é bem *amplo e importante*.

Na *grande maioria dos softwares* que trabalham com validação de dados e busca de padrões, nós *temos o RegEx sendo usado!*

Exemplos de uso:

- Web Scrapping
 - *A coleta/mineração de dados*
- Validar E-mails
- Encontrar Palavras Chaves
- Substituir Termos



```
import re

texto = "Venha se tornar um ByLearner!"
padrao = r"\bByLearn\w*\b"

resultado = re.search(padrao, texto)

if resultado:
    print("Encontramos o padrão no texto, entre os índices: "
          +f"{resultado.start()} e {resultado.end()}")
else:
    print("Não encontramos o padrão no texto")
```

Saída:

*Encontramos o padrão no texto,
entre os índices: 19 e 28*



```
import re

texto = "O rato roeu a roupa do rei de Roma"
padrao = r"rato|roupa|rei"

ocorrencias = re.finditer(padrao, texto)

for ocorrencia in ocorrencias:
    print(f"Encontrei: {ocorrencia.group()} entre os índices {ocorrencia.span()}")
```

Saída:

```
Encontrei: rato entre os índices (2, 6)
Encontrei: roupa entre os índices (14, 19)
Encontrei: rei entre os índices (23, 26)
```



```
import re

texto = "Um dois quatro quatro cinco"
padrao = r"quatro"
substituicao = r"tres"

novo_texto = re.sub(padrao, substituicao, texto, 1)
print(novo_texto)
```

Saída:

Um dois três quatro cinco



```
import re

texto = "O rato roeu a roupa do rei de Roma"
padrao = r"rato|roupa|rei"

ocorrencias = re.finditer(padrao, texto)

for ocorrencia in ocorrencias:
    print(f"Encontrei: {ocorrencia.group()} entre os índices {ocorrencia.span()}")
```

Saída:

```
Encontrei: rato entre os índices (2, 6)
Encontrei: roupa entre os índices (14, 19)
Encontrei: rei entre os índices (23, 26)
```




```
import re

texto = "O rato roeu a roupa do rei de Roma"
padrao = r"[rR](\w*)"
substituicao = r"g\1"

novo_texto = re.sub(padrao, substituicao, texto)
print(novo_texto)
```

Saída:

O gato goeu a goupá do gei de goma

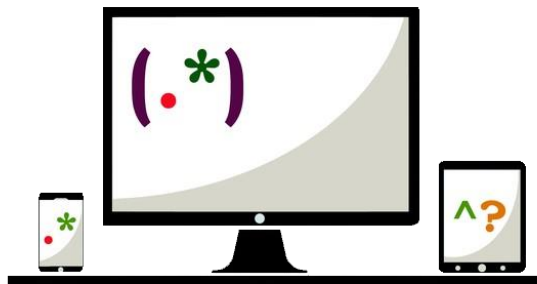
```
import re

texto = "Felipe Cabrera"
padrao = r"\w+\s\w+"

regex = re.fullmatch(padrao, texto)

if regex:
    print("Valido")
else:
    print("Inválido")
```

Saída:
Válido



EXPRESSÕES REGULARES

Agora que você já sabe a importância, que tal aprender de verdade as regras do RegEx?

Temos dois materiais fantásticos te esperando:

[1\) Aprenda sobre Expressões Regulares](#)

[2\) Aprenda usar o RegEx com o Python](#)



ByLearn

Obrigado por escolher nossos cursos!