

PROGRAMAÇÃO WEB II

Curso Técnico Integrado em Informática
Lucas Sampaio Leite



Professor

Lucas Sampaio Leite
lucas.leite@ifbaiano.edu.br



Objetivo

- Capacitar os alunos no desenvolvimento de aplicações web, utilizando o modelo cliente-servidor, frameworks modernos e a integração eficiente com bancos de dados, garantindo a construção de sistemas dinâmicos, escaláveis e seguros.

Ementa

- Modelo cliente-servidor; Ambiente de Desenvolvimento Web; Frameworks; Integração com banco de dados.

Conteúdo

1. Fundamentos de Programação Orientada a Objetos (POO) em Python

- Revisão de conceitos básicos de Python
- Revisão dos princípios da Programação Orientada a Objetos (POO)
 - Classes, objetos, atributos e métodos
 - Encapsulamento, herança, polimorfismo e abstração
- Boas práticas Python

2. Modelo Cliente-Servidor e Introdução ao Desenvolvimento Web

- Arquitetura cliente-servidor e HTTP
- Protocolos e métodos HTTP (GET, POST, PUT, DELETE)
- Comunicação entre frontend e backend
- APIs RESTful: conceitos e boas práticas

Conteúdo

3. Gerenciamento de Pacotes em Python

- Uso de pip, pipenv e poetry

4. Banco de Dados Relacionais com Python DB API

- Introdução a bancos de dados relacionais
- Criação e conexão de banco de dados com Python DB API
- Manipulação de dados com Python DB API

5. Desenvolvimento Web com Flask

- Introdução ao Flask
- Manipulação de dados com Flask
- Autenticação e autorização em Flask
- Testando aplicações Rest em Flask
- Deploy de uma API

Conteúdo

6. Desenvolvimento Web com Django

- Introdução ao Django
- Modelos e banco de dados com Django
- Administração do Django Framework
- Views, templates e formulários com Python e Django
- Autenticação e Autorização em Django
- Testando projetos em Django
- Deploy de aplicações Python com Django

7. Desenvolvimento de APIs com FastAPI

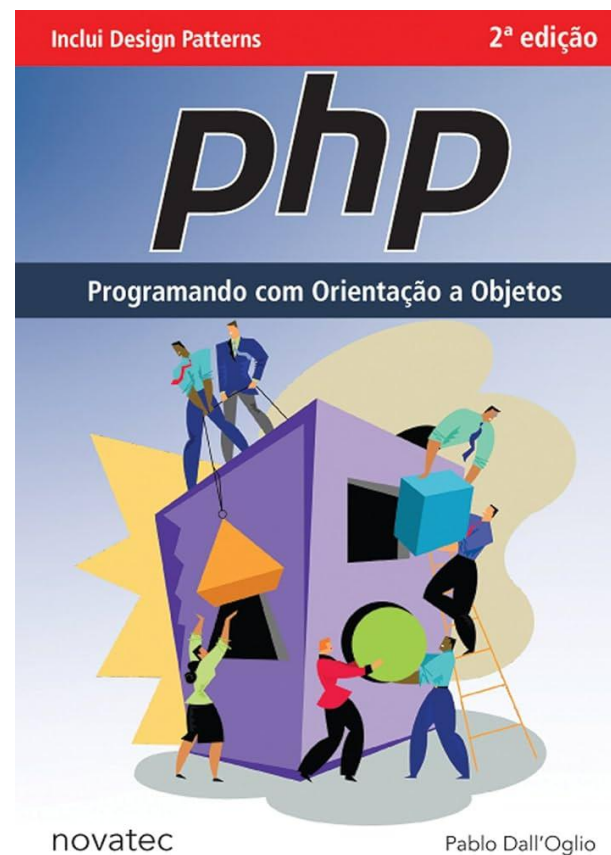
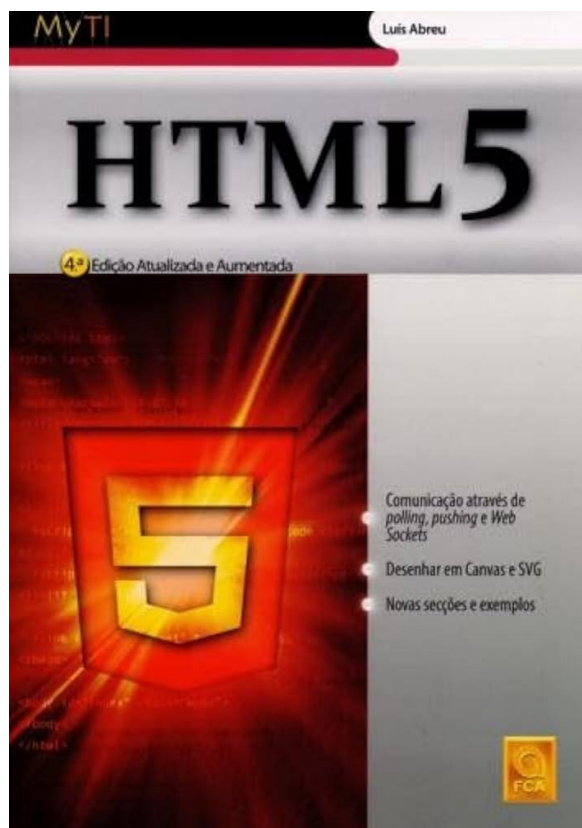
Metodologia

- Aulas síncronas e presenciais
 - Aulas expositivas e dialogadas
 - Abordagem prática
 - Aulas de acompanhamento de projeto
- Listas de Exercício / Provas / Projeto
- Com chamada

Avaliação

- Listas de Exercício / Provas / Projeto
- Avaliação 1: Listas de exercício 30% + Prova 70%
- Avaliação 2: A definir...
- Avaliação 3: A definir...
- Avaliação 4: A definir...

Bibliografia Básica



Bibliografia Básica

- LUÍS Abreu. HTML 5. 319. p. ISBN 9789727227396
- DALL’OGLIO, Pablo. PHP: Programando com Orientação a Objetos. 2. ed. São Paulo: Novatec, 2012. ISBN 9788575222003.
- SILVA, Maurício Samy. Construindo sites com CSS e (X) HTML: sites controlados por folhas de estilo em cascata. São Paulo: Novatec, 2008. 446 p. ISBN 9788575221396.

Bibliografia Complementar

- GILMORE, W. Jason. Dominando PHP e MySQL: do iniciante ao profissional. Rio de Janeiro: Alta Books, 2011. XXVIII, 769 p. ISBN 9788576083023.
- DUCKETT, Jon. HTML e CSS: projete e construa websites. Alta Books: 2016. ISBN 8576089394.
- DUCKETT, Jon. Javascript e JQuery: desenvolvimento de interfaces web interativas. Alta Books: 2016. ISBN 8576089459.

Outras Referências (gratuito)

- Documentação do Python: <https://docs.python.org/pt-br/3/>
- Material Panda USP:
https://panda.ime.usp.br/panda/static/pythonds_pt/index.html
- Documentação Flask: <https://flask.palletsprojects.com/en/stable/>
- Documentação Django: <https://docs.djangoproject.com/pt-br/5.1/>
- Documentação FastAPI: <https://fastapi.tiangolo.com/pt/>

Referência de exercícios Python



ListaDeExercicios


Esta é uma lista com sugestões de programas para iniciantes em programação. Os exercícios podem ser implementados em pseudo-código, Python, C, C++, Java, Pascal ou em qualquer outra linguagem. Os exercícios estão divididos em categorias e procuram obedecer uma ordem de dificuldade crescente. Cada exercício é independente do outro, sendo possível deixar exercícios para trás sem grandes prejuízos.

Lista de exercícios:

1. [EstruturaSequencial](#)
2. [EstruturaDeDecisao](#)
3. [EstruturaDeRepeticao](#)
4. [ExerciciosListas](#)
5. [ExerciciosFuncoes](#)
6. [ExerciciosComStrings](#)
7. [ExerciciosArquivos](#)
8. [ExerciciosClasses](#)
9. [ListaDeExerciciosProjetos](#)


<https://wiki.python.org.br/ListaDeExercicios>

Referência de exercícios Python (com árbitro digital)

 ENGLISH


[LOGIN](#) [REGISTER](#) [FORUM](#) [PROFESSORS](#) [CORPORATIONS](#)

BEECROWD




We are a global community of developers committed to keep evolving as students and professionals. Train algorithms and programming challenges and become the expert you always dreamed to be.

[CORPORATE PAGE](#)



USE SOCIAL SIGN IN



OR


EMAIL

PASSWORD

☐ REMEMBER ME (7 DAYS) [SIGN IN](#)


FIRST TIME HERE?
SIGN UP today to join one of the largest developer and competitive programming communities in the world!

COMPETITION AND RANKING




Join the brightest minds in competitive programming! Participate in competitions, contests and tournaments! Compare your knowledge with your peers. Level up, grow and shine in your career!

PROBLEM REPOSITORY



Our state-of-the-art competitive programming platform has 2,000+ analytical and programming tests available in more than 20 different programming languages. All tests are available in Portuguese and English.

BEECROWD ACADEMIC

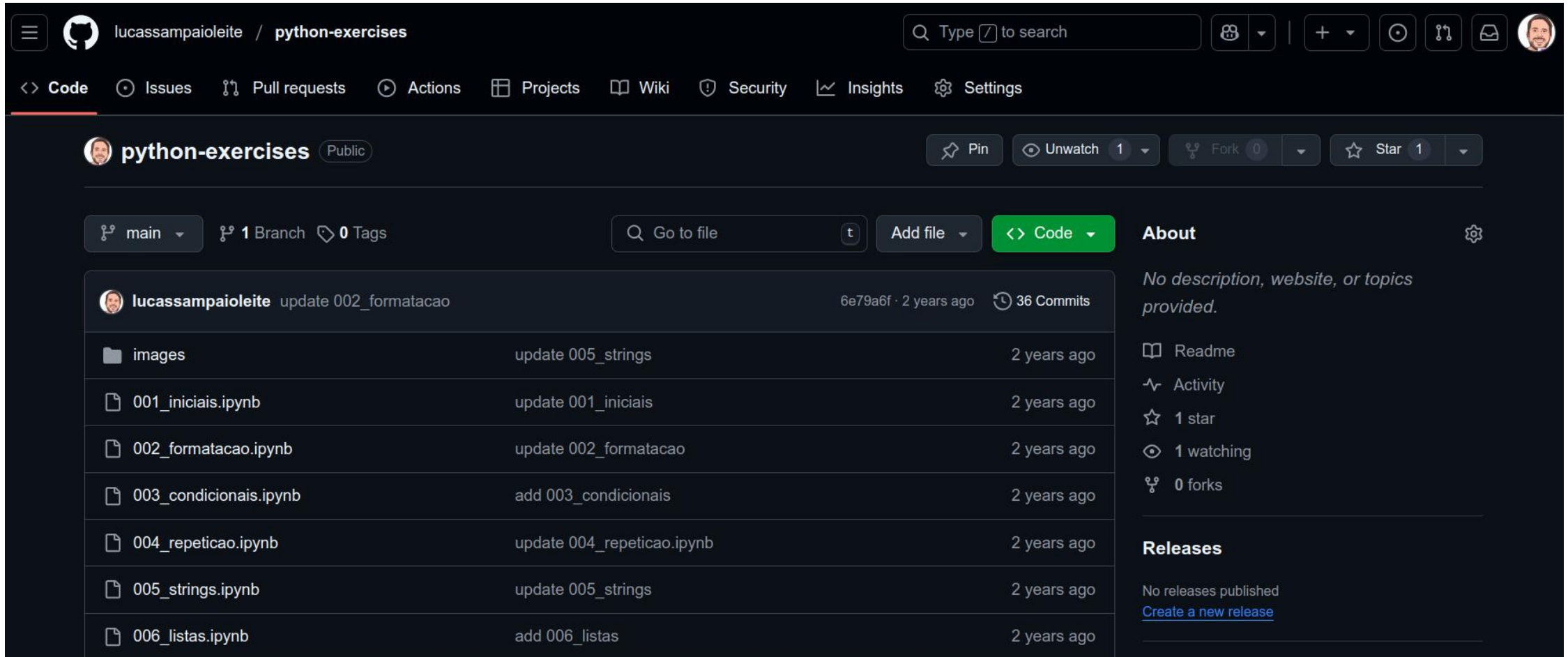


The beecrowd Academic is a module for Educational Institutions, Professors and Coaches. Here you can create courses, exercise lists and track your students progress giving them real-time feedback.

[ACCESS ACADEMIC](#)

<https://judge.beecrowd.com/>

Referência de exercícios resolvidos



The screenshot shows the GitHub repository page for 'python-exercises' by user 'lucassampaioleite'. The repository is public and has 1 star and 1 watcher. The main branch is 'main'. The repository contains several files and folders, including 'images', '001_iniciais.ipynb', '002_formatacao.ipynb', '003_condicionais.ipynb', '004_repeticao.ipynb', '005_strings.ipynb', and '006_listas.ipynb'. The repository also has 36 commits and 0 forks.

Repository Details:

- Repository: python-exercises (Public)
- Stars: 1
- Watchers: 1
- Forks: 0
- Commits: 36

Files and Commits:

File/Folder	Commit Message	Time Ago
images	update 005_strings	2 years ago
001_iniciais.ipynb	update 001_iniciais	2 years ago
002_formatacao.ipynb	update 002_formatacao	2 years ago
003_condicionais.ipynb	add 003_condicionais	2 years ago
004_repeticao.ipynb	update 004_repeticao.ipynb	2 years ago
005_strings.ipynb	update 005_strings	2 years ago
006_listas.ipynb	add 006_listas	2 years ago

About: No description, website, or topics provided.

Releases: No releases published. [Create a new release](#)

<https://github.com/lucassampaioleite/python-exercises>

Boas práticas

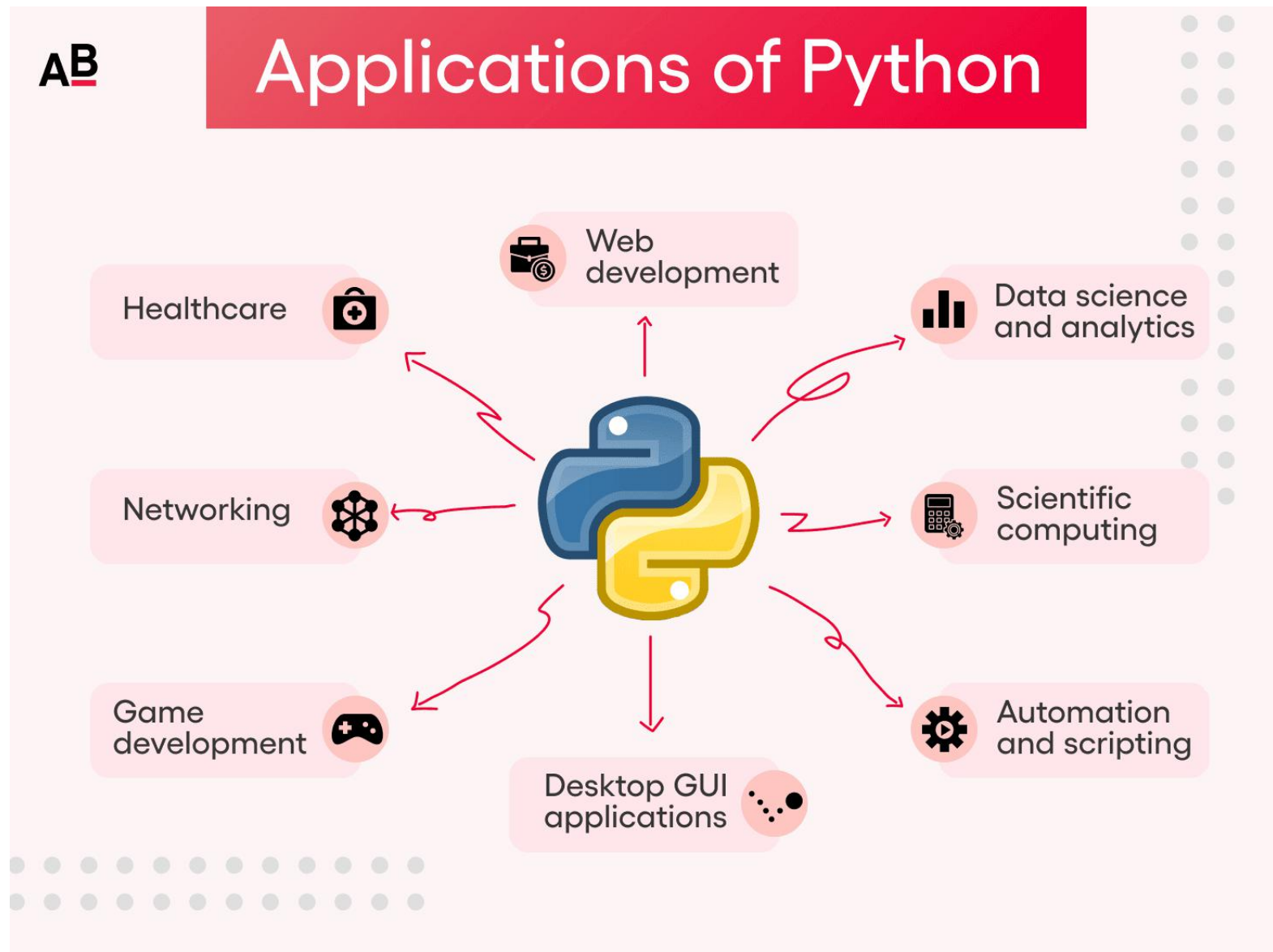
- Para a dinâmica de aprendizagem da disciplina funcionar, é importante realizar as práticas e exercícios passados em sala (não deixem acumular atividades).
- Quem deixa acumular as atividades, tende a ter um desempenho inferior.
- Não deixem o projeto para última hora.
- Dividam a carga entre todos os membros da equipe.
- Organizem seu tempo entre as disciplinas.

Um pouco de Python

- Python foi lançada por Guido Van Rossum em 1991.
- Python é uma linguagem de programação de alto nível e interpretada.
- Características principais:
 - Indentação: a indentação obrigatória.
 - Interpretada: O código gerado pelo desenvolvedor é interpretado para linguagem de máquina somente no momento em que ele é atingido em tempo de execução.
 - Legibilidade: Facilita o aprendizado e o desenvolvimento dos programas. Aproxima a codificação da linguagem natural.
 - Multiparadigma: Maiori liberdade ao desenvolvedor. Existe o suporte a Programação Orientada a Objetos (POO), imperativa e funcional.



Um pouco de Python



<https://www.almbetter.com/bytes/tutorials/python/python-features-and-applications>

Um pouco de Python

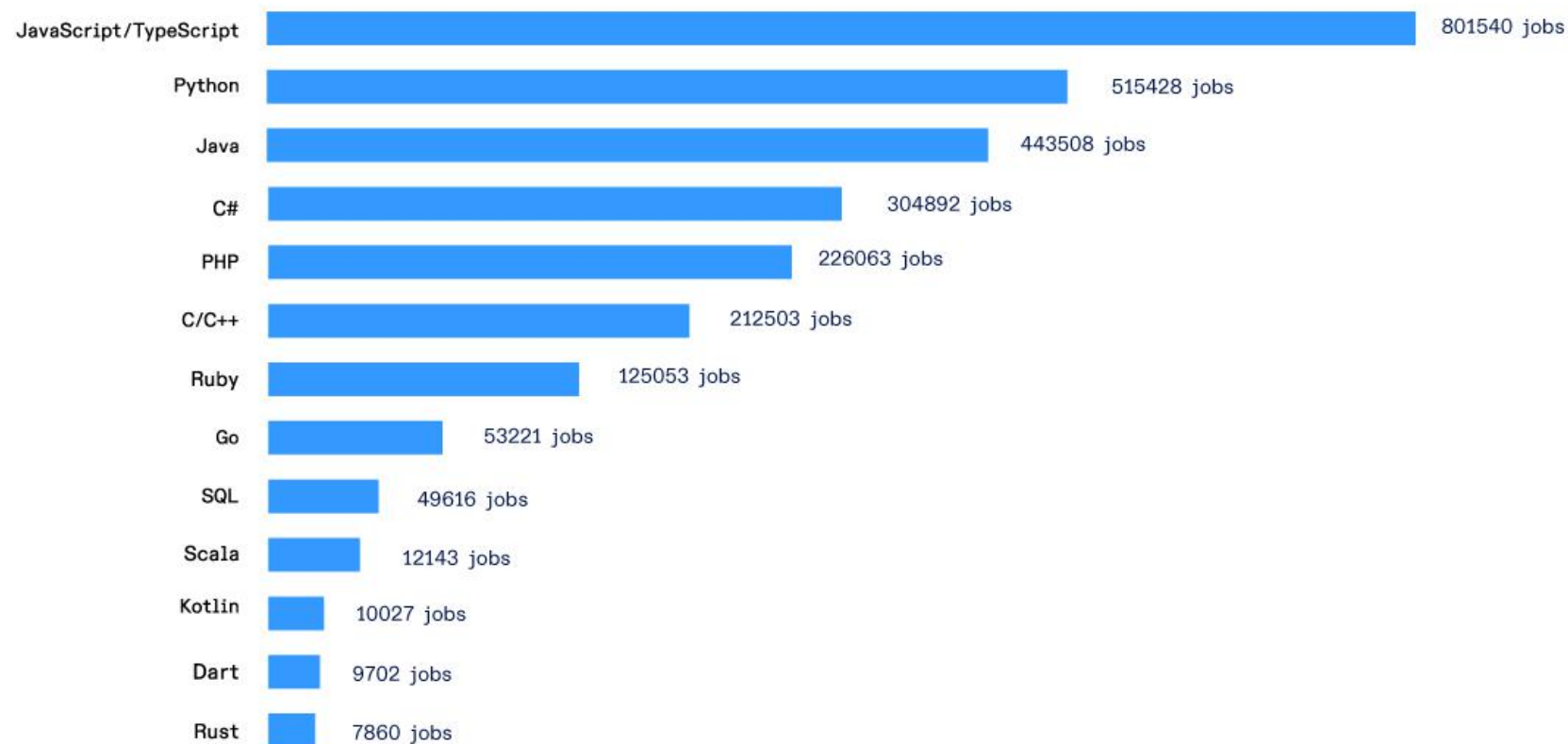
Top Companies Using Python



<https://medium.com/nerd-for-tech/top-10-python-development-company-to-hire-d83507a33755>

Um pouco de Python

Most Demanded Programming Languages by Number of Jobs

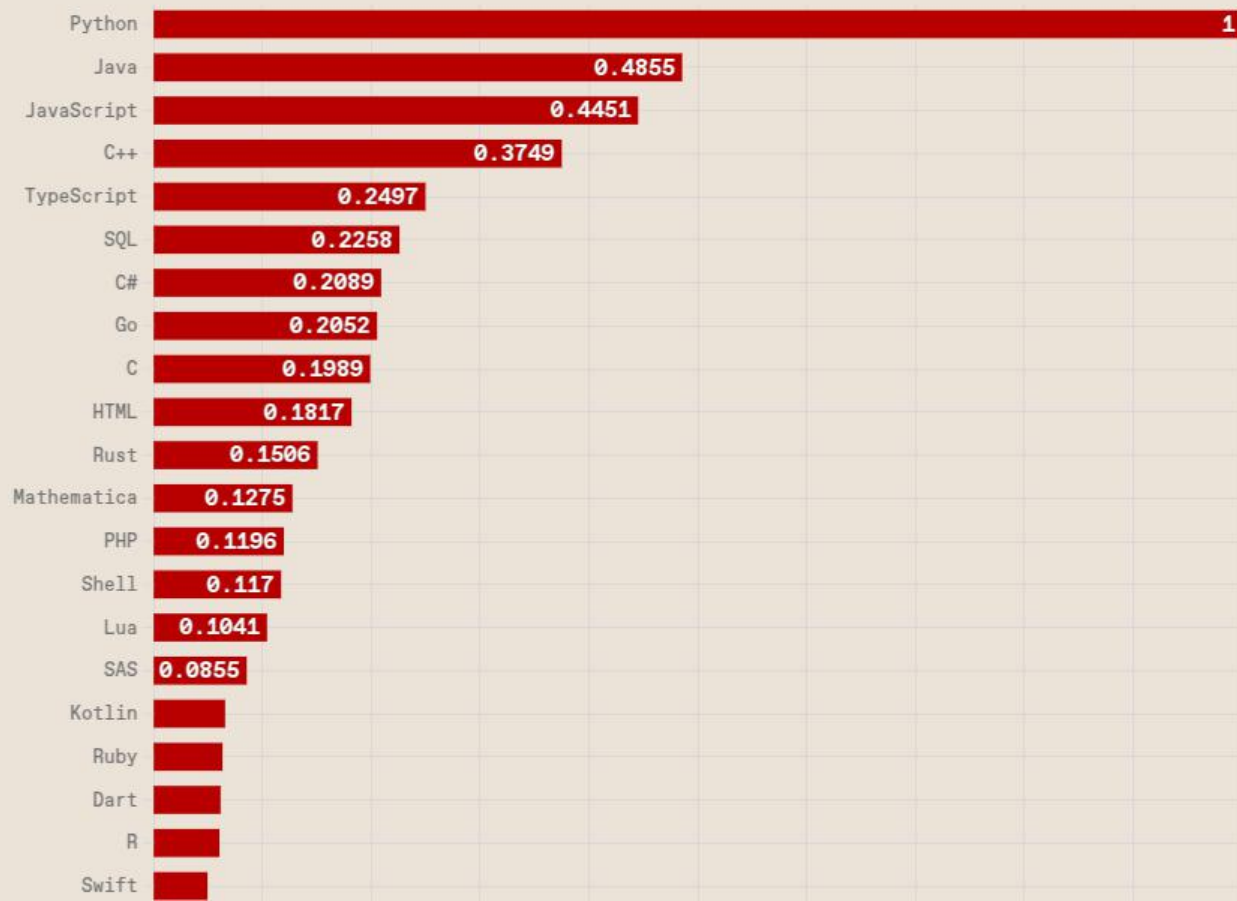


<https://www.index.dev/blog/most-in-demand-programming-languages>

Um pouco de Python

Top Programming Languages 2024

Click a button to see a differently weighted ranking



<https://spectrum.ieee.org/top-programming-languages-2024>

Revisão de conceitos básicos em Python

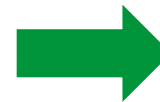
- Um nome de variável pode ser definido em Python por uma sequência de letras ($a \rightarrow z$, $A \rightarrow Z$) e números ($0 \rightarrow 9$), que devem sempre começar com uma letra.
- Apenas letras comuns são permitidas. Letras acentuadas, cedilhas, espaços, caracteres especiais como \$, #, @, etc. são proibidos, exceto para o caractere `_` (sublinhado/underline).

Por convenção, as variáveis devem começar com letras minúsculas.

Revisão de conceitos básicos em Python

- As variáveis em Python tem um tipo, que é definido dinimicamente de acordo com o valor passado no atribuição.
- Cada tipo define os valores que a variável pode armazenar.
- Cada tipo ocupa uma certa quantidade de memória.

```
main.py
1  print(type(1))
2  print(type(1.1))
3  print(type(1 + 1.1j))
4  print(type('Python'))
5  print(type([1, 1.1, 1 + 1.1j, 'Python']))
6
```



```
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'str'>
<class 'list'>
```


Revisão de conceitos básicos em Python

- A regra de nomeação das constantes no Python segue um padrão parecido com as de variáveis, com a diferença de que todas as letras são maiúsculas e separadas por underline “_”.

```
main.py > ...  
1  MINHA_CONSTANTE = 10  
2  print(MINHA_CONSTANTE) # 10  
3  
4  MINHA_CONSTANTE = 15  
5  print(MINHA_CONSTANTE) # 15  
6
```

CUIDADO!!! Os valores atribuídos à constantes em Python podem ser alterados.

Revisão de conceitos básicos em Python

Descrição	É válido?	Exemplo
Iniciar por letra		
Iniciar por número		
Conter letras e números		
Conter underline		
Conter espaço		
Conter caracter especial		
Conter acentuação		

Revisão de conceitos básicos em Python

Descrição	É válido?	Exemplo
Iniciar por letra	Sim	variavel_teste
Iniciar por número	Não	1variavel
Conter letras e números	Sim (desde que não inicie por número)	variavel1
Conter underline	Sim	_variavel_
Conter espaço	Não	variavel teste
Conter caracter especial	Não	variavel*teste
Conter acentuação	Sim (v3) / Não (v2)	variável

Operadores aritméticos

Operação	Operador
Adição	+
Subtração	-
Multiplicação	*
Divisão	/

Operadores aritméticos

Operação	Operador
Exponenciação	**
Parte inteira do resultado da divisão	//
Módulo	%

Ordem de precedência dos operadores aritméticos

Operador	Ordem de resolução na expressão
()	1º
**	2º
*, /, //, %	3º
+, -	4º

Vamos exercitar?

a) $5+3*2$

b) $5/2$

c) $5//3$

d) $4*(5+2)**3$

e) $420**339$

f) $5\%2$

g) $9**(1/2)$

h) $8**(1/3)$

i) $\text{pow}(2,5)$

Revisão de conceitos básicos em Python

Operador	Exemplo	Operação
=	x = 5	x recebe 5
+=	x += 5	Adiciona 5 a x
-=	x -= 3	Decrementa 3 de x
*=	x *= 2	Multiplica x por 2
/=	x /= 4	Divide x por 4
**=	x **= 2	x elevado a 2
%=	x %= 2	Resto de x por 2
//=	x //= 3	Quociente inteiro de x por 3

Revisão de conceitos básicos em Python

- O operador + executa uma concatenação de strings, ou seja, une as strings pelas extremidades.
- O operador * também funciona em strings; ele executa a repetição da string.

```
instituicao = "IF Baiano"  
campus = "Senhor do Bonfim"  
curso = "Técnico Integrado em Informática"  
resultado = instituicao + "/" + campus + "/" + curso  
print(5*"=" + resultado + 5*"=")
```



```
=====IF Baiano/Senhor do Bonfim/Técnico Integrado em Informática=====
```

Revisão de conceitos básicos em Python

Operador	Referente a:
==	Igual a
!=	Diferente
>=	Maior ou igual
>	Maior que
<	Menor que
<=	Menor ou igual

Revisão de conceitos básicos em Python

Operador	Referente a:
and	e
or	ou
not	não

Revisão de conceitos básicos em Python

- Considere os valores de $a=5$, $b=10$ e $c=20$. Qual das expressões abaixo teria resultado False?

- a) `(a > b) or (c >= b)`
- b) `(b > a) and (c >= a)`
- c) `(not(a > b)) and (c >= b)`
- d) `((a != b) or (a == c)) and (b != c)`
- e) `(c != a) and (b == c)`

Revisão de conceitos básicos em Python

- Comandos de entrada e saída:

```
1 nome = input('Digite o seu nome: ')\n2 idade = int(input('Digite a sua idade: '))\n3 print('Olá, {}! Você tem {} anos.'.format(nome, idade))\n4 print(f'Olá, {nome}! Você tem {idade} anos.')\n5 nova_idade = idade + 25\n6 print('Daqui a 25 anos, sua idade será {} anos.'.format(nova_idade))\n7 print(f'Daqui a 25 anos, sua idade será {nova_idade} anos')
```



Os dados digitados pelos usuários sempre serão do tipo <str>

Revisão de conceitos básicos em Python

- Comandos de entrada e saída:

```
1 nome = input('Digite o seu nome: ')\n2 idade = int(input('Digite a sua idade: '))\n3 print('Olá, {}! Você tem {} anos.'.format(nome, idade))\n4 print(f'Olá, {nome}! Você tem {idade} anos.')\n5 nova_idade = idade + 25\n6 print('Daqui a 25 anos, sua idade será {} anos.'.format(nova_idade))\n7 print(f'Daqui a 25 anos, sua idade será {nova_idade} anos')
```



```
Olá, Lucas! Você tem 31 anos.\nOlá, Lucas! Você tem 31 anos.\nDaqui a 25 anos, sua idade será 56 anos.\nDaqui a 25 anos, sua idade será 56 anos.
```

Revisão de conceitos básicos em Python

```
1 nome = input("Digite o seu nome: ")
2 idade = int(input("Digite a sua idade: "))
3 altura = float(input("Digite a sua altura "))
4
5 print(nome, type(nome))
6 print(idade, type(idade))
7 print(altura, type(altura))
```



```
Digite o seu nome: Lucas
Digite a sua idade: 18
Digite a sua altura 1.72
Lucas <class 'str'>
18 <class 'int'>
1.72 <class 'float'>
```

Revisão de conceitos básicos em Python

- Pode-se usar aspas triplas (""" ou """) para criar uma string em várias linhas e, em seguida, imprimi-la. Isso pode ser útil quando você tem um bloco de texto grande para imprimir:

```
1 mensagem = """
2 Esta é a primeira linha.
3 Esta é a segunda linha.
4 E aqui está a terceira linha.
5 """
6
7 print(mensagem)
```



```
● lucas@lucas-Inspiron-5548:~/Dr
  /Logica/codes/teste.py

Esta é a primeira linha.
Esta é a segunda linha.
E aqui está a terceira linha.
```


Revisão de conceitos básicos em Python

- Formatar número de casas de ponto flutuante:

```
PI = 3.141592653589793238462643383279

print("PI com 2 casas decimais {:.2f}".format(PI))
print("PI com 3 casas decimais {:.3f}".format(PI))
print("PI com 10 casas decimais {:.10f}".format(PI))

print(f"PI com 2 casas decimais {PI:.2f}")
print(f"PI com 3 casas decimais {PI:.3f}")
print(f"PI com 10 casas decimais {PI:.10f}")
```



```
PI com 2 casas decimais 3.14
PI com 3 casas decimais 3.142
PI com 10 casas decimais 3.1415926536
PI com 2 casas decimais 3.14
PI com 3 casas decimais 3.142
PI com 10 casas decimais 3.1415926536
```

Revisão de conceitos básicos em Python

- Estrutura condicional (if):

if <condição>:
 bloco verdadeiro

```
numero = int(input("Digite um número: "))

if numero%2 == 0:
    print(f"{numero} é par.")

if numero%2 == 1:
    print(f"{numero} é ímpar.")
```

Revisão de conceitos básicos em Python

- Estrutura condicional (if-else):

```
if <condição>:  
    bloco verdadeiro  
else:  
    bloco falso
```

```
numero = int(input("Digite um número: "))  
  
if numero%2 == 0:  
    print(f"{numero} é par.")  
else:  
    print(f"{numero} é ímpar.")
```

Revisão de conceitos básicos em Python

- Estrutura condicional (if-elif-else):
 - Substitui o else seguido de if, com um único comando.

```
main.py > ...  
1 a = 5  
2 if a > 0:  
3     print('valor positivo')  
4 else:  
5     if a < 0:  
6         print('valor negativo')  
7     else:  
8         print('valor nulo')
```

=

```
main.py > ...  
1 a = 5  
2 if a > 0:  
3     print('valor positivo')  
4 elif a < 0:  
5     print('valor negativo')  
6 else:  
7     print('valor nulo')
```


Revisão de conceitos básicos em Python

- Estrutura condicional (if-elif-else):
 - Substitui o else seguido de if, com um único comando.

```
num1 = int(input("Digite o primeiro número: "))
num2 = int(input("Digite o segundo número: "))

if num1 == num2:
    print("Os números digitados são iguais.")
elif num1 > num2:
    print("O primeiro número é maior que o segundo.")
else:
    print("O segundo número é maior que o primeiro.")
```

Revisão de conceitos básicos em Python

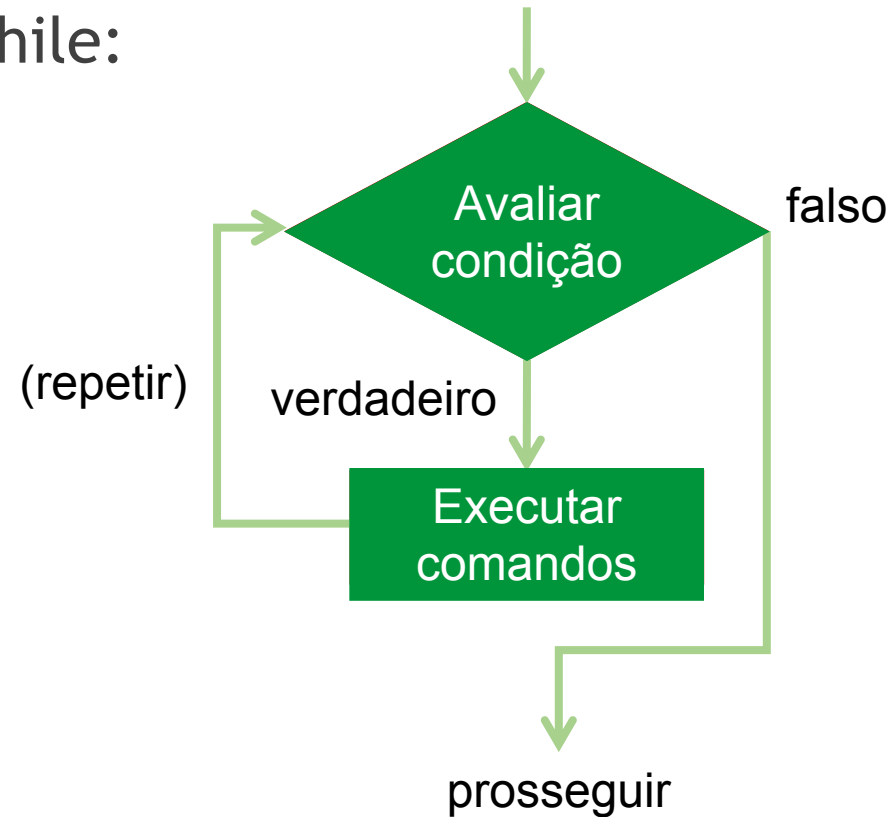
```
temp = 2
if temp < 10:
    if temp < 5:
        print('muito frio')
    else:
        print('frio')
else:
    if temp > 30:
        print('muito quente')
    else:
        if temp < 20:
            print('agradável')
        else:
            print('quente')
```

=

```
temp = 10
if temp < 5:
    print('muito frio')
elif temp < 10:
    print('frio')
elif temp < 20:
    print('agradável')
elif temp <= 30:
    print('quente')
else:
    print('muito quente')
```

Revisão de conceitos básicos em Python

- Fluxograma do while:



- Observe que há a possibilidade de nunca se executar os comandos caso a primeira avaliação da condição já resulte em falso.

Revisão de conceitos básicos em Python

- O que será impresso?

```
main.py > ...  
1     i = 0  
2     while(i < 10):  
3         print(i)  
4         i = i+1
```


Revisão de conceitos básicos em Python

- O que será impresso?

```
main.py > ...  
1   i = 0  
2   while(i < 10):  
3       print(i)  
4       i = i+1
```

```
main.py > ...  
1   i = 10  
2   while(i >= 1):  
3       print(i)  
4       i = i-1
```

Revisão de conceitos básicos em Python

- O que será impresso?

```
main.py > ...  
1   i = 0  
2   while(i < 10):  
3       print(i)  
4       i = i+1
```

```
main.py > ...  
1   a = 0  
2   b = 2  
3   while a <= b:  
4       print( a, ' <= ', b, ' ' )  
5       a += 1
```

```
main.py > ...  
1   i = 10  
2   while(i >= 1):  
3       print(i)  
4       i = i-1
```

Revisão de conceitos básicos em Python

- O que o código abaixo faz?

```
main.py > ...  
1  while(True):  
2      nota1 = float(input('Digite a primeira nota do aluno: '))  
3      nota2 = float(input('Digite a segunda nota do aluno: '))  
4      nota3 = float(input('Digite a terceira nota do aluno: '))  
5  
6      media = (nota1+nota2+nota3)/3  
7  
8      print('A média do aluno é: ', media)  
9  
10     digito = input('Deseja continuar? (s/n):')  
11     if digito == 'n':  
12         break
```

Revisão de conceitos básicos em Python

- O que o código abaixo faz?

```
main.py > ...  
1   i = 0  
2   while i < 100:  
3       i = i + 1  
4       if i % 5 == 0:  
5           continue  
6       print(i)
```

Revisão de conceitos básicos em Python

- A sintaxe da utilização do for com a função range():

for i in range <n>:
comandos

- Exemplo:

```
main.py  
1  for i in range(5):  
2      print(i)  
3
```

Neste caso, o laço for imprime na tela o respectivo valor de i para cada iteração.
O primeiro valor de i é 0 e segue sendo incrementado de um em um até o número 4.


Revisão de conceitos básicos em Python

- A sintaxe da utilização do for com a função range():

```
for i in range (<inicio>,<fim>):  
    comandos
```

- Exemplo:

```
main.py > ...  
1   for i in range(2,6):  
2       print(i)  
3
```



Neste caso, o laço for imprime na tela o respectivo valor de i para cada iteração.

O primeiro valor de i é 2 e segue sendo incrementado de um em um até o número 5.

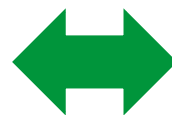
Revisão de conceitos básicos em Python

- A sintaxe da utilização do for com a função range():

for i in range (<inicio>, <fim>, <incremento>):
comandos

- Exemplo:

```
main.py > ...  
1 m=1  
2 n=100  
3 p=2  
4 for i in range (m,n,p):  
5     print(i)
```



```
main.py > ...  
1 for i in range(1,100,2):  
2     print(i)  
3
```

O que seria impresso?

Revisão de conceitos básicos em Python

```
for x in range(0, 5, 1):  
    print(x)
```

início (opcional) –
quando omitido,
início = 0

fim (obrigatório)

incremento (opcional) –
quando omitido,
incremento = 1

Revisão de conceitos básicos em Python

- Percorrendo uma string:

```
main.py > ...  
1  x = 'Lógica de programação'  
2  for i in x:  
3      print (i)
```

Revisão de conceitos básicos em Python

- Percorrendo uma lista:

```
main.py > ...  
1  ✓ for i in [0, 1, 2, 3]:  
2      |     print(i)  
3
```

Dúvidas



PROGRAMAÇÃO WEB II

Curso Técnico Integrado em Informática
Lucas Sampaio Leite

