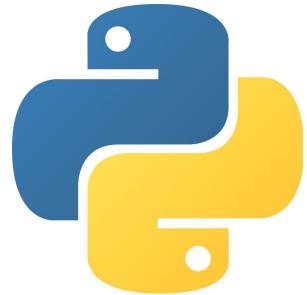


**COMPOSITION BOOK**

*Python*

# A linguagem Python é:

- Uma linguagem com propósito geral
- Fácil e Intuitiva
- Multiplataforma
- Batteries included
- Livre
- Organizada
- Orientada a objetos
- Muitas bibliotecas



# Principais Áreas:

- Inteligência Artificial
- Biotecnologia
- Computação 3D



# Zen of Python



- Bonito é melhor que feio.
- Explícito é melhor que implícito
- Simples é melhor que complexo
- Complexo é melhor que complicado
- Linear é melhor que aninhado
- Esparsos é melhor que densos
- Legibilidade conta
- Casos especiais não são especiais o bastante para quebrar as regras
- Ainda que praticidade vença a pureza
- Erros nunca devem passar silenciosamente
- A menos que sejam explicitamente silenciados
- Diante da ambiguidade, recuse a tentação de adivinhar
- Deveria de haver um - e preferencialmente só um - modo óbvio para fazer algo
- Embora esse modo possa não ser óbvio a princípio a menos que você seja holandês
- Agora é melhor que nunca
- Embora nunca frequentemente seja melhor que *\*ja\**
- Se a implementação é difícil de explicar, é uma má ideia
- Se a implementação é fácil de explicar, pode ser uma boa ideia
- Namespaces são uma grande ideia - vamos ter mais dessas!

# Operações básicas em Python

## - Comandos -

Print ('Mensagem!') - escreve na tela o que está entre parenteses

Print (2+4) - quando se trata de números para a realização de algum cálculo matemático por exemplo, não precisa colocar os parenteses, pois o programa vai realizar o cálculo e mostrar o resultado.

Print ('7' + '4') - o programa vai mostrar - '74', isso porque o 7 e 4 estão entre parenteses (ou seja, agora são mensagens) e o sinal de + junta as mensagens, gerando o resultado 74

Print ('Olá', 5) - junta a mensagem com um número, se tentarmos utilizar o + ( print ('ola' + 5)) vai dar erro

Print (variável , variável , variável) - vai mostrar todas as variáveis em uma só linha, exemplo: « Print (Nome, idade) - Silvia, 26

# Variaveis

- Em Python todas as variaveis sao objetos;
- Colocar o nome das variaveis apenas em minusculas;
- Toda variavel pode receber valores, com o sinal = 'recebe'

## Exemplos:

nome = 'Silvia'  
Lê-se - A variavel 'nome' recebe 'Silvia'

idade = 26

peso = 63

Print (nome, idade, peso) - Resultado - Silvia 26 63

Quando queremos que seja o usuario a digitar o que vai estar dentro da variavel usamos o INPUT

## Exemplo:

nome = input ('Qual é o seu nome? :')

Resposta do programa - Qual é o seu nome? : (usuario responde)

idade = input ('Qual a sua idade? :')

Resposta do programa - Qual a sua idade? : (usuario responde)

Print (nome , idade) - Resposta - Jean 30

# Tipos Primitivos

Quando utilizamos INPUT, mesmo que dentro esteja escrito um numero, o programa sempre vai ler como uma string.

Para tornar uma string em numeros inteiros para serem executado em funções podemos usar a função INT

Exemplo:

```
nl = int( input ('Digite um numero: ' ) )
```

**Int** - numeros inteiros — 7, -4, 0, 5467

**Float** - numeros reais, ou numeros de ponto flutuante — 4.5, 0.674, -12.76

**Bool** - valores logicos, ou booleanos — True, False

**Str** - caracteres, ou strings — 'Ola', '7.5', ''

```
Print ('A soma vale', s)
```

Colocar chaves {} para ficar mais fácil de ler

```
Print ('A soma vale {}' .format(s))
```

Usar .format(variaveis na ordem que queremos usar em{})

# Operadores Aritméticos



+ Adição

\*\* Potencia

- Subtração

// Divisão inteira

\* Multiplicação

/ Divisão

% Resto da Divisão

## Exemplos:

$5 + 2 == 7$

Para se ler e representar o igual em Python, utilizaremos dois sinais de igual ==, porque um só significa recebe

$5 ** 2 == 25$

$5 - 2 == 3$

$5 // 2 == 2$

$5 * 2 == 10$

$5 \% 2 == 1$

$5 / 2 == 2.5$

## Ordem de procedencia - Operações aritméticas

1 - ()

2 - \*\*

3 - \* / // %

4 - + -

Exemplos:

$$3 + 5^2 = 11$$

$$3^2 \cdot 5 + 4^2 \cdot 2 = 31$$

$$3^3 \cdot (5 + 4)^2 \cdot 2 = 243$$



# Utilizando modulos

Biblioteca (ex: Math)

Para importar a biblioteca inteira:

Modulos:

ceil  
floor  
trunc  
pow  
sqrt  
factorial

Import math

Para importar um modulo específico da biblioteca:

From math import sqrt

Sqrt - Calcula a raiz quadrada

Ceil - arredonda para cima

Floor - arredonda para baixo



## Importante:

Ctrl + espaço mostra a lista de bibliotecas e modulos

Para instalar uma biblioteca externa usar:

pip install nome-da-biblioteca

# Manipulando Cadeias de Caracteres

Cadeia de caracteres são frases, 'strings', dentro de uma variável, e cada caractere é representado por um número:



## Fatiamento:

Frase[9] — 'V' → Pega apenas o caractere escolhido

Frase[9:13] — 'Vide' → Pega do primeiro caractere escolhido até o anterior ao caractere escolhido

Frase[9:21:2] — 'VdoPto' → Vai do 9 até 21(20) pulando de 2 em 2

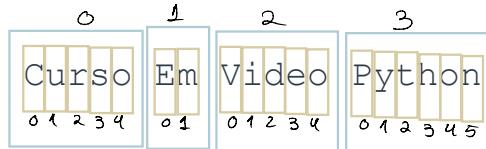
Frase[:5] — 'Curso' → Vai começar no inicio da string até o final pedido 5(4, o número final nunca aparece)

Frase[:15] — 'Python' → Começa no 15 como indicado e vai até o final da string

Frase[9::3] — 'VePh' → Vai começar no 9 e como não colocamos nada entre : ele vai até o final da string, o 3 no final significa que ele vai pular de 3 em 3

## Divisão:

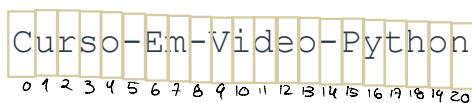
Frase.split() —



Ele divide cada elemento em uma lista, faz o corte do elemento a cada espaço

## Junção:

— 'join(frase)' —



Ele junta todas as listas e une com o símbolo que está entre "", dá para colocar espaço em branco entre "" e assim a frase aparece com espaços

Frase =   
 Análise:

`Len(frase) — 21` → O len nos diz o tamanho da string, neste caso ela tem 21 caracteres

`Frage.count('o') — 3` → Mostra quantas vezes aparece a letra escolhida entre "", neste caso "o" aparece 3 vezes

`Frage.count('o'), 0, 13 — 1` → Mostra quantas vezes a letra "o" aparece entre os espaços 0 a 13, lembrando que o último número nunca é considerado então é de 0 a 12

`Frage.find('deo') — 11` → Ele diz nos onde encontrou a palavra ou letra entre "", neste caso ele encontrou "deo" na linha 11, mesmo tendo 3 caracteres ele só vai mostrar onde começa

`Frage.find('Android') — -1` → Quando retorna o valor -1, significa que ele não achou a palavra ou letra entre "".

`'Curso' in frase — True` → O find mostra onde achou, mas o in nos dá valor de verdadeiro ou falso, neste caso é verdadeiro

## Transformação:

`Frage.replace('Pytgon', 'Android')` — 'Curso Em Video Android'

→ Substituiria a palavra "Python", por "Android", mesmo que Android seja maior, o Python adiciona mais espaço

`Frage.upper()` — 'CURSO EM VIDEO PYTHON' → Coloca tudo em maiúsculo

`Frage.lower()` — 'curso em video python' → Coloca tudo em minúsculo

`Frage.capitalize()` — 'Curso em video python' → Deixa a primeira em maiúsculas e todo resto em minúsculas

`Frage.title()` — 'Curso Em Video Python'

→ Ele faz uma quebra nas palavras quando tem espaços e coloca a primeira letra de todas as palavras em maiúsculo

Frase = 

`Frage.strip()` — 'Aprenda Python' → Vai remover espaços iniciais do começo e do final

`Frage.rstrip()` — ' Aprenda Python' → O "r" é de right, ou seja vai remover todos os espaços à direita

`Frage.lstrip()` — 'Aprenda Python ' → O "l" é de left, ou seja vai remover todos os espaços à esquerda

# Condições - if..else

Se carro.esquerda()  
bloco\_V\_

Senão  
bloco\_F\_

## Em Python

```
if carro.esquerda():  
    bloco True  
else:  
    bloco False
```

### Exemplo:

```
tempo = int(input('Quantos anos tem seu carro?'))  
if tempo <=3:  
    print('Seu carro é novo')  
else:  
    print('Seu carro é velho')  
print('—Fim—')
```

### Condição simplificada

```
tempo = int(input('Quantos anos tem seu carro?'))  
print('Carro novo') if tempo <=3 else 'Carro velho'  
print('—Fim—')
```

# Cores no Terminal

ANSI — escape sequence

Sempre que quiser aplicar uma cor utilizar o código:

\033[  
  style  
  0;33;44m  
  text  
  back

## Style

0 — none (nenhum)

1 — bold (negrito)

4 — underline (sublinhado)

7 — Negative (inverter)

Text                          Back

37      branco                  40

31 — vermelho — 41

32 — verde — 42

33 — amarelo — 43

34 — azul — 44

35 — roxo — 45

36 — azul ciano — 46

30 — cinza — 47