



# Introdução ao Python: fundamentos e estruturas lógicas

Poliana N Ferreira



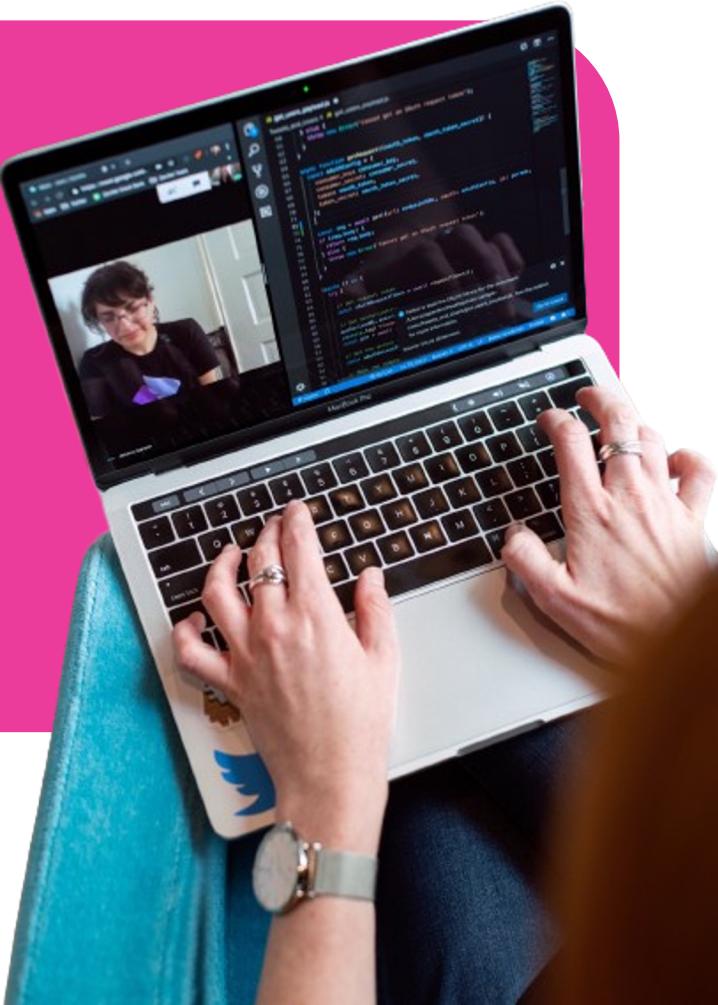
Quem sou eu

### Poliana Ferreira

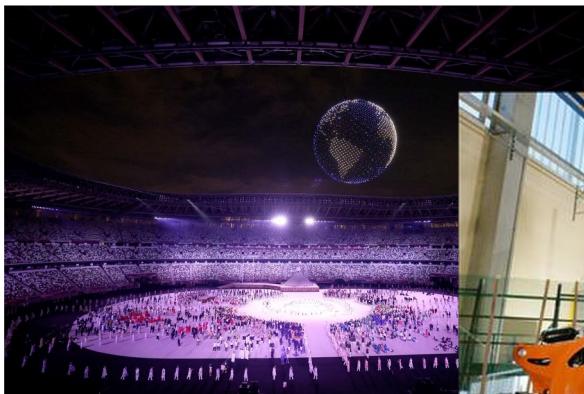
- Desenvolvendo projetos em dados há 7 anos
- Mestranda em Física na área de Computação e ML Quântico
- Mestre em Ciência da Computação na área de Wearables, Machine Learning e Educação (UFABC)
- Graduada em Análise e Desenvolvimento de Sistemas (FIAP)

Vamos lá?

# Introdução a Programação



# Programação e Computadores no dia a dia



A screenshot of a website page. The header features a purple banner with the text "INFORMAÇÕES OFICIAIS SOBRE O CORONAVIRUS" in white. Below the banner, there's a logo of the city of Vitória da Conquista and a section titled "Educação". The main content area contains text about the municipal education department, followed by a "Apresentação" section with contact information for the Santander branch.

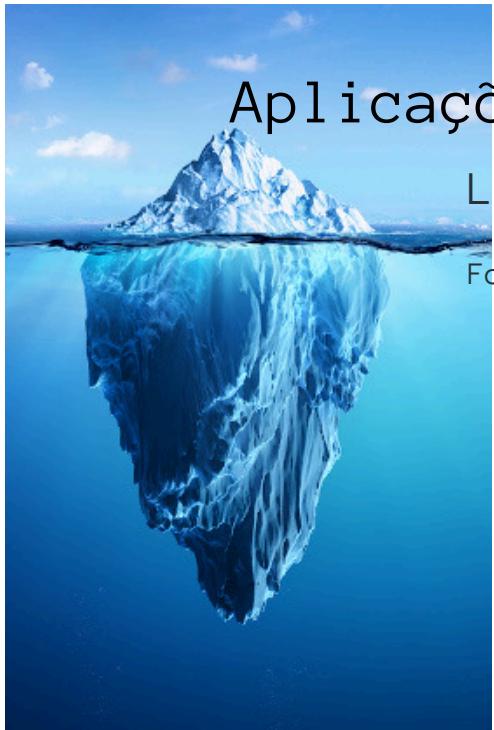


KORU  
SCHOOL

# O Iceberg da Programação

## Algoritmos

Podemos explicar a lógica da programação, a **base de tudo**, usando o “Pensamento Computacional”.



Aplicações do dia a dia

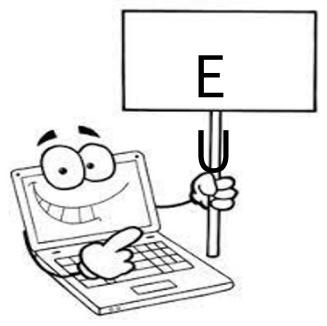
Linguagens de programação

Forma de comunicação com o computador;

# Algoritmos

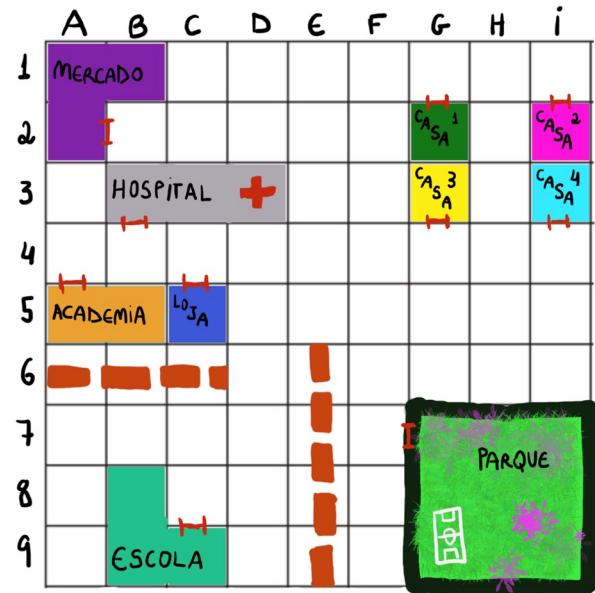
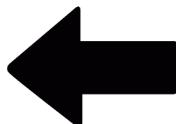
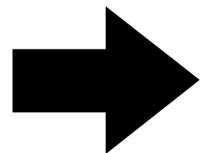
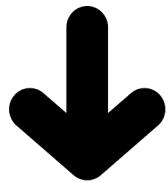
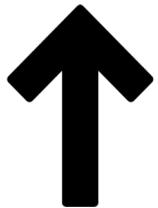
## O que é?

- Sequência bem definida de passos que têm por objetivo resolver um problema;
- Instruções sequenciais, específicas e explícitas.
- Forma lógica de organizar o pensamento

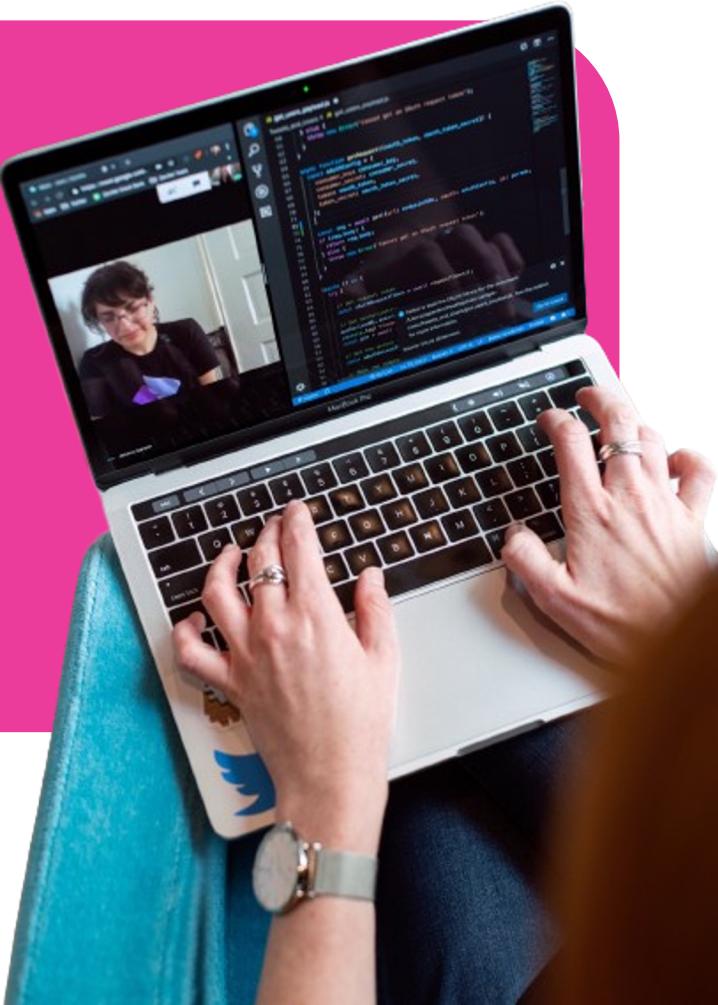


# Vamos “programar” um carro autônomo?

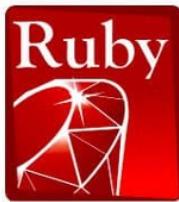
- Como vamos fazer isso?
- Com setas de comando
- E um mapa de uma cidade!



# Linguagens de Programação



# Linguagens de Programação Não Visuais



C#



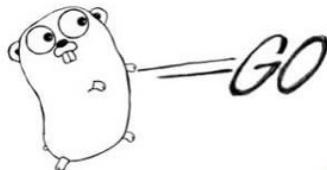
C++



Objective-C



Perl



# Python

<https://www.python.org/>

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

Linguagem de Programação

- Comandos escritos estruturados
- Similar ao inglês
- 1991 – até hoje
  - Grande comunidade de desenvolvedores



# Python

## Como programar?

- Para programar, usamos uma “IDE” – Ambiente de desenvolvimento integrado.
- Vamos baixar o Anaconda e VSCode
- Possibilidade: Google Colab

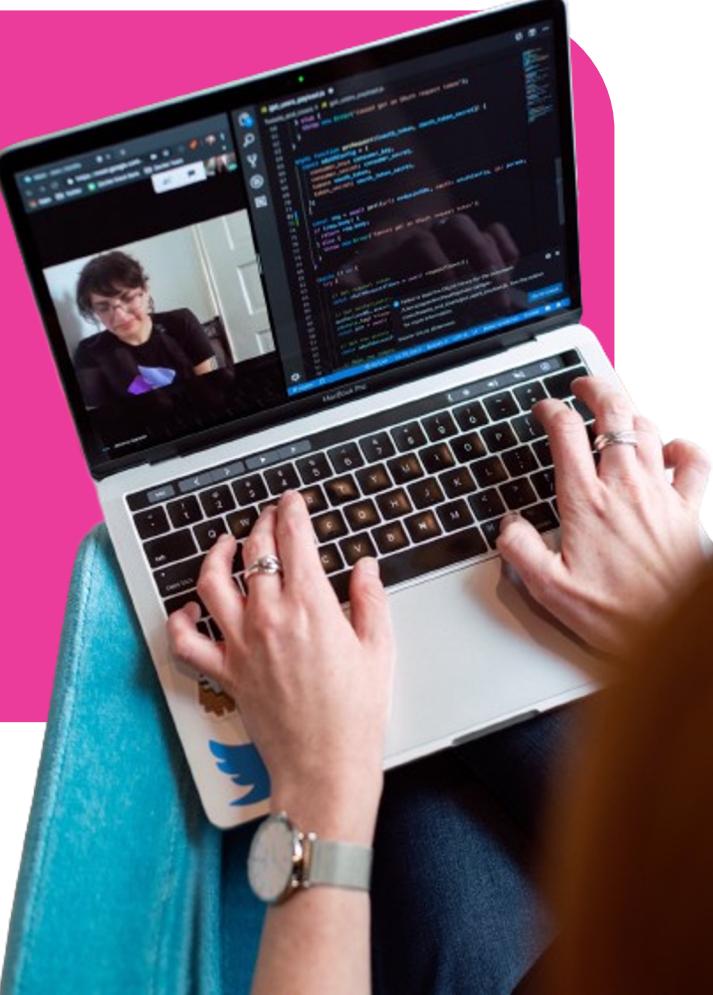


<https://colab.google/>

<https://www.anaconda.com/download>

<https://code.visualstudio.com/>

# Python



# Python

## Primeiro Programa

- Podemos escrever um texto na tela (sempre entre aspas)

```
[1] print("Olá, mundo!")
```

```
Olá, mundo!
```

# Python

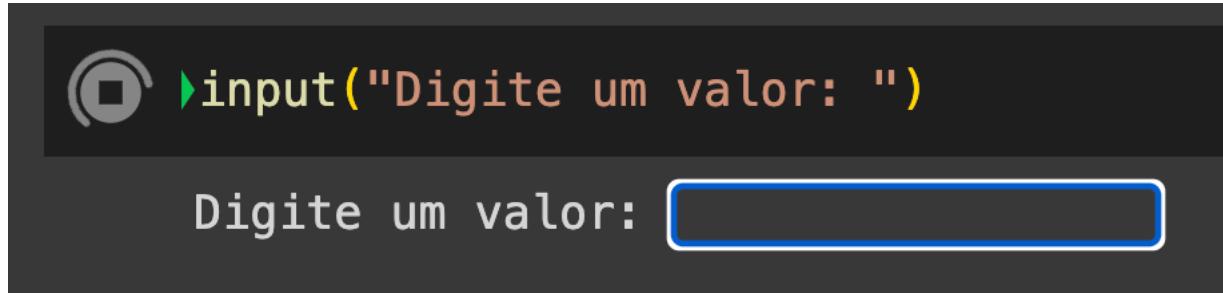
- Podemos fazer o programa imprimir números ou o resultado de operações aritméticas:

```
[4] print(2+2) #soma  
      print(10-8) #subtr  
      print(2*4) #multiplicação  
      print(4/3) #divisão  
      print(4//3) #div inteira  
      print(10%4) #resto
```

```
4  
2  
8  
1.333333333333333  
1  
2
```

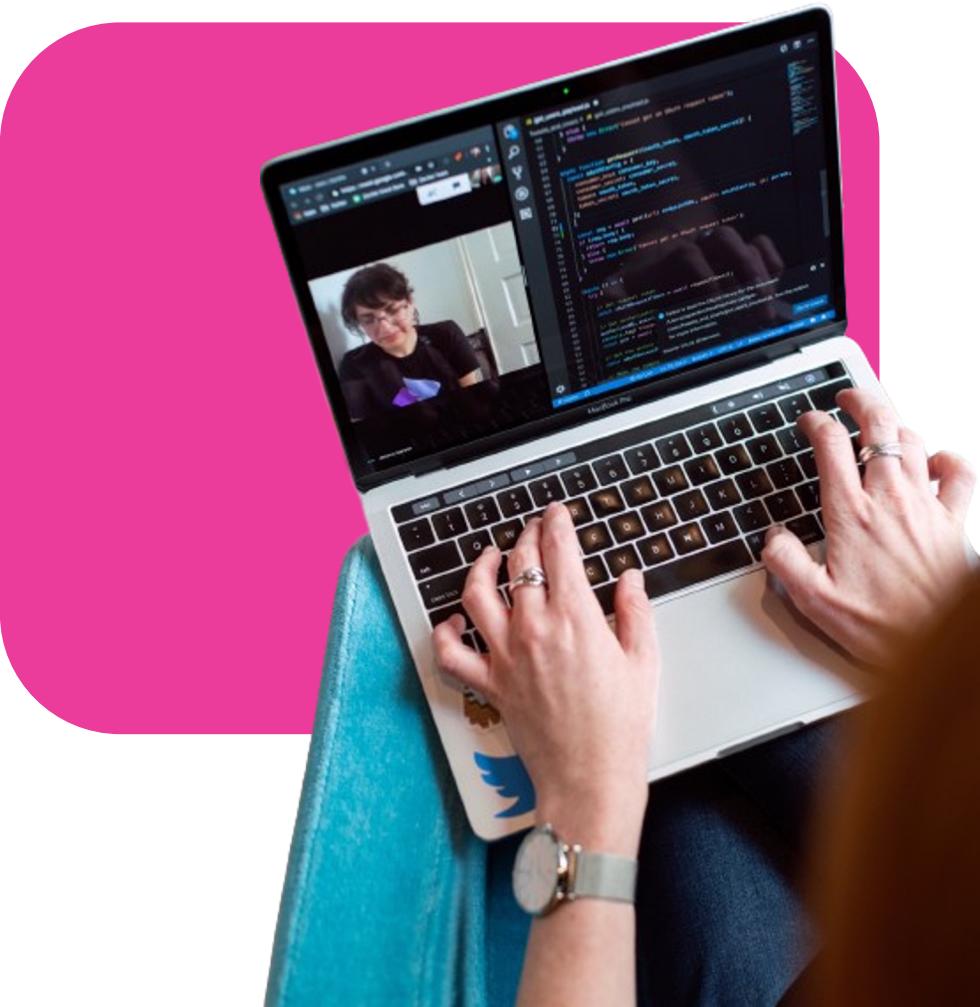
# Entrada de Dados - Python

- Podemos também ler um valor que o usuário do programa escrever.
- Mas e como armazenar esse valor para realizar ações com ele?



A screenshot of a terminal window. On the left, there is a small icon of a terminal window. To its right, the Python code `input("Digite um valor: ")` is displayed in green. Below the code, the text "Digite um valor:" is shown in white, followed by a blue-outlined rectangular input field where the user can type a value.

# Variáveis em Python



# Variáveis

Nome que definimos para armazenar  
um valor na programação



# Variáveis

- As variáveis são um nome (ex.: “x”, “turma”, “pontos”) que representa um local na memória do computador que guarda a informação.
- Para **guardar** informação, você deve escrever o nome, símbolo = e o valor
- Para **buscar a informação, basta você escrever o nome definido e imprimir, por exemplo**

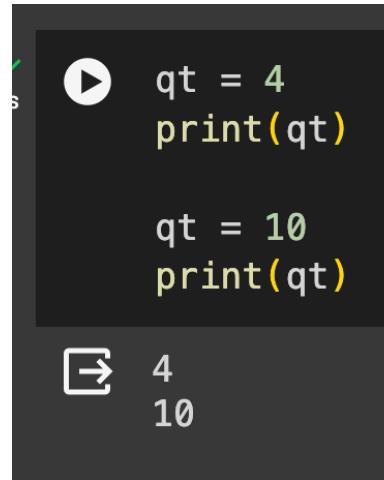
```
[5] nome = "Poliana"  
     idade = 24  
     divisao = 20/3  
  
     print(nome, idade, divisao)
```

```
Poliana 24 6.666666666666667
```

# Variáveis

- Podemos fazer operações entre as variáveis também.
- Ou reescrever o seu valor

```
[7] print(divisao + idade)  
30.666666666666668
```



The screenshot shows a Jupyter Notebook cell with the following code:

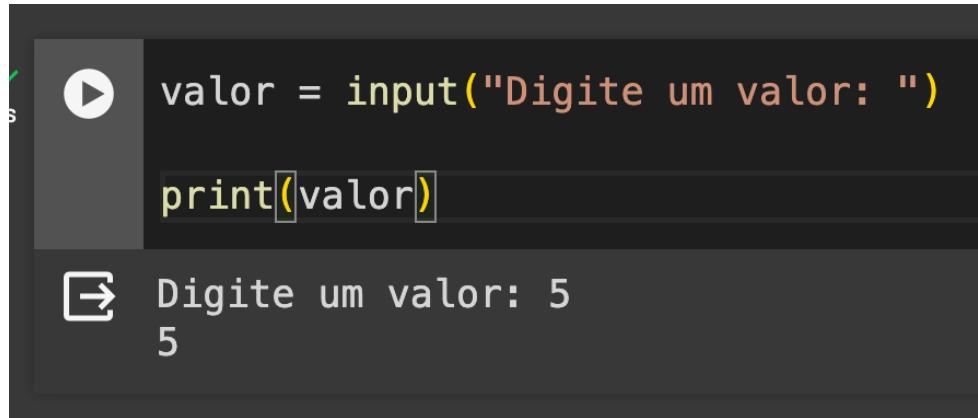
```
qt = 4  
print(qt)  
  
qt = 10  
print(qt)
```

When run, it produces the following output:

```
4  
10
```

# Entrada de Dados - Python

- Podemos também ler um valor que o usuário do programa escrever:
- E armazená-lo em uma variável.



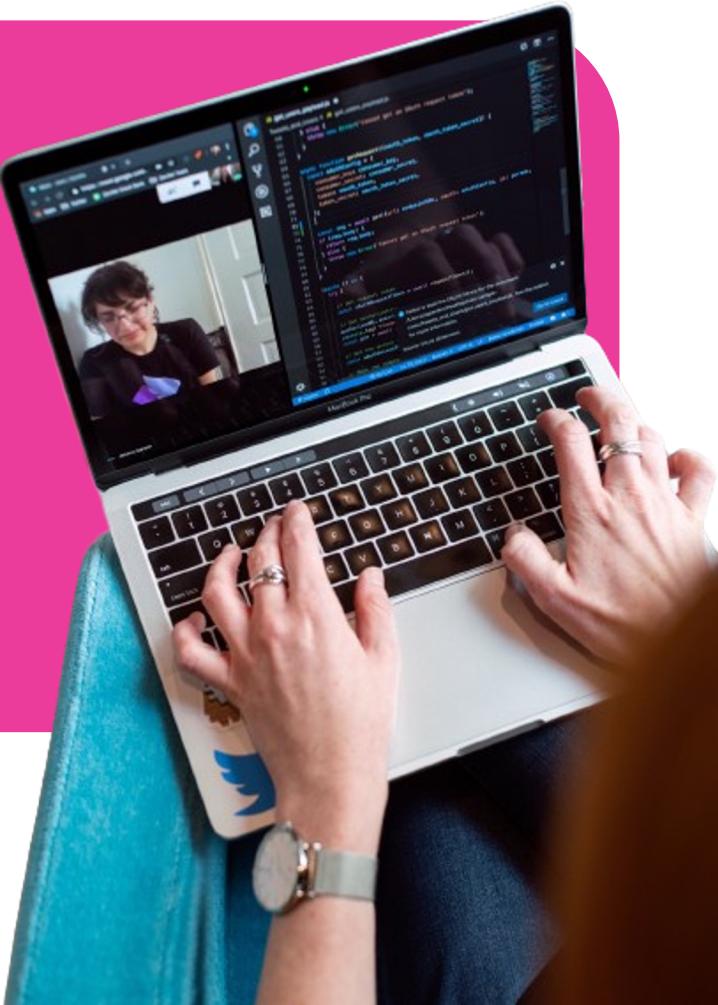
```
valor = input("Digite um valor: ")  
print(valor)
```

Execution output:

```
→ Digite um valor: 5  
5
```

The screenshot shows a dark-themed code editor window. On the left, there's a vertical toolbar with a play button icon. The main area contains two lines of Python code: `valor = input("Digite um valor: ")` and `print(valor)`. Below the code, the terminal output is shown, starting with a right-pointing arrow and the text "Digite um valor: 5", followed by the number "5".

# Tipos de Dados em Python



# Entrada de Dados - Python

Cada variável tem um tipo diferente:

- int – números inteiros;
- float – números racionais;
- str – texto escrito: “texto”;
- bool – valores de verdadeiro (True) ou falso (False).

```
▶ nome = "Poliana"
idade = 24
divisao = 20/3
v_ou_f = True

print(type(nome))
print(type(idade))
print(type(divisao))
print(type(v_ou_f))

➡ <class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

# Métodos para manipulação de strings

LIST OF PYTHON STRING METHODS				
isalnum()	isupper()	capitalize()	lower()	rsplit()
isalpha()	istitle()	casefold()	lstrip()	rstrip()
isascii()	isspace()	center()	maketrans()	split()
isdecimal()	index()	count()	partition()	splitlines()
isdigit()	format_map()	encode()	replace()	startswith()
isidentifier()	format()	endswith()	rfind()	strip()
islower()	find()	expandtabs()	rindex()	swapcase()
isnumeric()	upper()	join()	rjust()	title()
isprintable()	zfill()	ljust()	rpartition()	translate()

```
▶ texto = " Olá, Mundo! Este é um exemplo de uso de métodos de strings em Python. "
  texto_upper = texto.upper()
  texto_strip = texto.strip() #remove espaços em branco do inicio e fim
  posicao_exemplo = texto.find('exemplo')
  lista_palavras = texto.split() # divide a string por espaços
  contagem_o = texto.count('o')
  texto_isupper = texto_upper.isupper()

  (texto_upper, texto_strip, posicao_exemplo, lista_palavras, contagem_o, texto_isupper)

⇒ (' OLÁ, MUNDO! ESTE É UM EXEMPLO DE USO DE MÉTODOS DE STRINGS EM PYTHON. ',
   'Olá, Mundo! Este é um exemplo de uso de métodos de strings em Python.',
   24,
   ['Olá',
    'Mundo!',
    'Este',
    'é',
    'um',
    'exemplo',
    'de',
    'uso',
    'de',
    'métodos',
    'de',
    'strings',
    'em',
    'Python.'],
   6,
   True)
```

# Conversão de tipos

Podemos alterar o tipo de uma variável, por exemplo:

- Número em um texto;
- Texto (formato) em número;
- Valor inteiro para um float;
- Número para booleano;

```
▶ cast_idade = float(idade)
  cast_divisao = int(divisao)
  cast_idade2 = str(idade)

  print(type(cast_idade), cast_idade)
  print(type(cast_divisao), cast_divisao)
  print(type(cast_idade2), cast_idade2)

⇒ <class 'float'> 24.0
  <class 'int'> 6
  <class 'str'> 24
```

```
[20] cast_bool = bool(0)
      cast_bool1 = bool(1)
      cast_bool2 = bool(2)

      print(cast_bool, cast_bool1, cast_bool2)

False True True
```

```
[24] int("12p")
-----
ValueError                                Traceback (most recent call last)
<ipython-input-24-e8f8517f6b14> in <cell line: 1>()
----> 1 int("12p")

ValueError: invalid literal for int() with base 10: '12p'

EXPLICAR O ERRO
```

## Tipo list

- Listas – armazena mais de um valor
- Similar a um vetor em outras linguagens
- Podemos acessar os valores através do índice



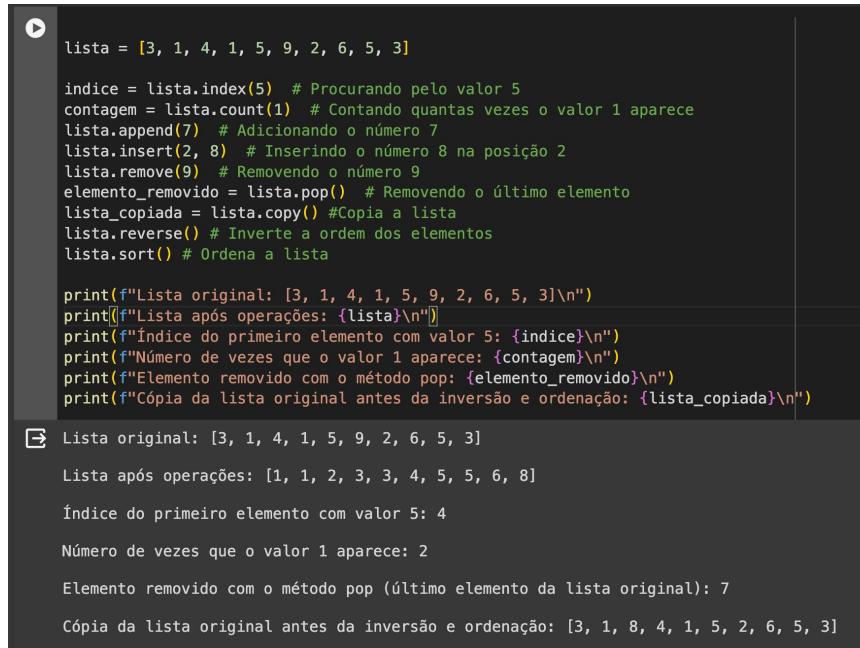
```
lista = ["Poliana", "Ana", "Alberto"]  
lista[0]
```



```
'Poliana'
```

# Métodos para manipulação de listas

- index
- count
- append
- insert
- remove
- pop
- copy
- reverse
- sort



```
lista = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3]

indice = lista.index(5) # Procurando pelo valor 5
contagem = lista.count(1) # Contando quantas vezes o valor 1 aparece
lista.append(7) # Adicionando o número 7
lista.insert(2, 8) # Inserindo o número 8 na posição 2
lista.remove(9) # Removendo o número 9
elemento_removido = lista.pop() # Removendo o último elemento
lista_copiada = lista.copy() #Copia a lista
lista.reverse() # Inverte a ordem dos elementos
lista.sort() # Ordena a lista

print(f"Lista original: {lista}\n")
print(f"Lista após operações: {lista}\n")
print(f"Índice do primeiro elemento com valor 5: {indice}\n")
print(f"Número de vezes que o valor 1 aparece: {contagem}\n")
print(f"Elemento removido com o método pop: {elemento_removido}\n")
print(f"Cópia da lista original antes da inversão e ordenação: {lista_copiada}\n")
```

Lista original: [3, 1, 4, 1, 5, 9, 2, 6, 5, 3]

Lista após operações: [1, 1, 2, 3, 3, 4, 5, 5, 6, 8]

Índice do primeiro elemento com valor 5: 4

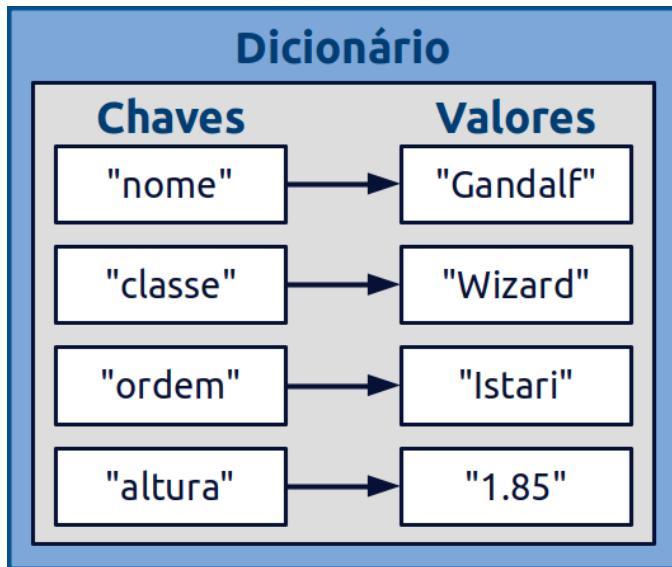
Número de vezes que o valor 1 aparece: 2

Elemento removido com o método pop (último elemento da lista original): 7

Cópia da lista original antes da inversão e ordenação: [3, 1, 8, 4, 1, 5, 2, 6, 5, 3]

# Tipo dict

- Dicionários – conjunto de pares chave/valor
- Acessamos os valores pela chave



```
[30] dicionario = {"nome":"Poliana", "idade":24, "profissao":"Professora"}  
print(dicionario['nome'])  
  
Poliana
```

# Métodos para manipulação de dicionários

- keys
- values
- items
- pop
- clear
- copy

```
[33] dicionario = {'nome': 'Alice', 'idade': 30, 'cidade': 'Curitiba'}

chaves = dicionario.keys() # Lista das chaves
valores = dicionario.values() # Lista dos valores
itens = dicionario.items() # Lista dos pares chave-valor
valor_removido = dicionario.pop('idade') # Removendo 'idade'
dicionario_copiado = dicionario.copy()

# Resultados
print(f"Chaves do dicionário: {chaves}\n")
print(f"Valores do dicionário: {valores}\n")
print(f"Itens do dicionário: {itens}\n")
print(f"Valor removido com chave 'idade': {valor_removido}\n")
print(f"Cópia do dicionário antes do método clear: {dicionario_copiado}\n")

dicionario.clear() # Remove tudo
print(f"Dicionário após o método clear: {dicionario}\n")


Chaves do dicionário: dict_keys(['nome', 'cidade'])

Valores do dicionário: dict_values(['Alice', 'Curitiba'])

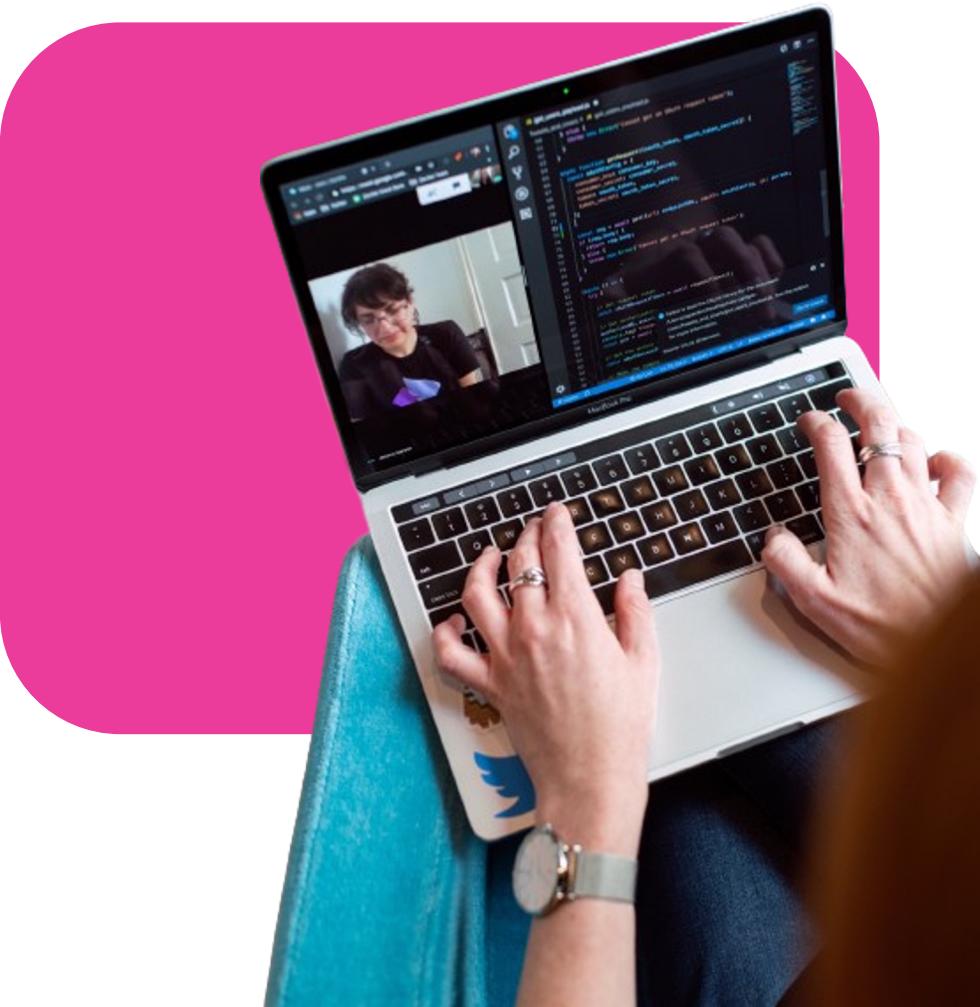
Itens do dicionário: dict_items([('nome', 'Alice'), ('cidade', 'Curitiba')])

Valor removido com chave 'idade': 30

Cópia do dicionário antes do método clear: {'nome': 'Alice', 'cidade': 'Curitiba'}

Dicionário após o método clear: {}
```

# Operações em Python



# Operações aritméticas

## Adição

```
print(2 + 2)
```

4

## Subtração

```
print(2 - 2)
```

0

## Multiplicação

```
print(2 * 5)
```

10

## Divisão

```
num1 = 2  
num2 = 5  
print(num1 / num2)  
# e divisão por zero?
```

0.4

## Potência

```
print( 2 ** 5)
```

32

## Raiz

```
print(9 ** (1/2))
```

3.0

## Mod - o resto da divisão

```
print(7 % 3)
```

```
print(4 % 2)
```

1

0

# Operações relacionais

Maior que: >

Maior ou igual: >=

Menor que: <

Menor ou igual: <=

Igual: ==

Diferente: !=



True  
False

# Operações relacionais

```
aula5.py
1 # Escreva o seu código aqui :-(

2 print(3>2)
3 print(3<2)
```

Em execução: aula5.py

```
True
False
>>> |
```

aula5.py



```
1 # Escreva o seu código aqui :-(

2 print(3>2)
3 print(2>2)
4 print(3>=2)
5 print(2>=2)
6
```

Em execução: aula5.py

```
True
False
True
True
>>>
```

aula5.py



```
1 # Escreva o seu código aqui :-(

2 print(1<2)
3 print(2<2)
4 print(1<=2)
5 print(2<=2)
6
```

Em execução: aula5.py

```
True
False
True
True
>>> |
```

# Operações lógicas

e (**and**) – conjunção entre dois valores

ou (**or**) – disjunção entre dois valores

não (**not**) – inverte o valor

## and

A	B	Resultado
True	True	True
True	False	False
False	True	False
False	False	False

# Operações lógicas

e (**and**) – conjunção entre dois valores

ou (**or**) – disjunção entre dois valores

não (**not**) – inverte o valor

**or**

A	B	Resultado
True	True	True
True	False	True
False	True	True
False	False	False

# Operações lógicas

e (**and**) – conjunção entre dois valores

ou (**or**) – disjunção entre dois valores

não (**not**) – inverte o valor

**not**

A	Resultado
True	False
False	True

# Operações lógicas

```
aula5.py X
1 # Escreva o seu código aqui :-)
2 print(True and True)
3 print(True and False)
4 print(False and True)
5 print(False and False)
```

Em execução: aula5.py

```
True
False
False
False
>>> |
```

```
aula5.py X
1 # Escreva o seu código aqui :-)
2 print(True or True)
3 print(True or False)
4 print(False or True)
5 print(False or False)
6 |
```

Em execução: aula5.py

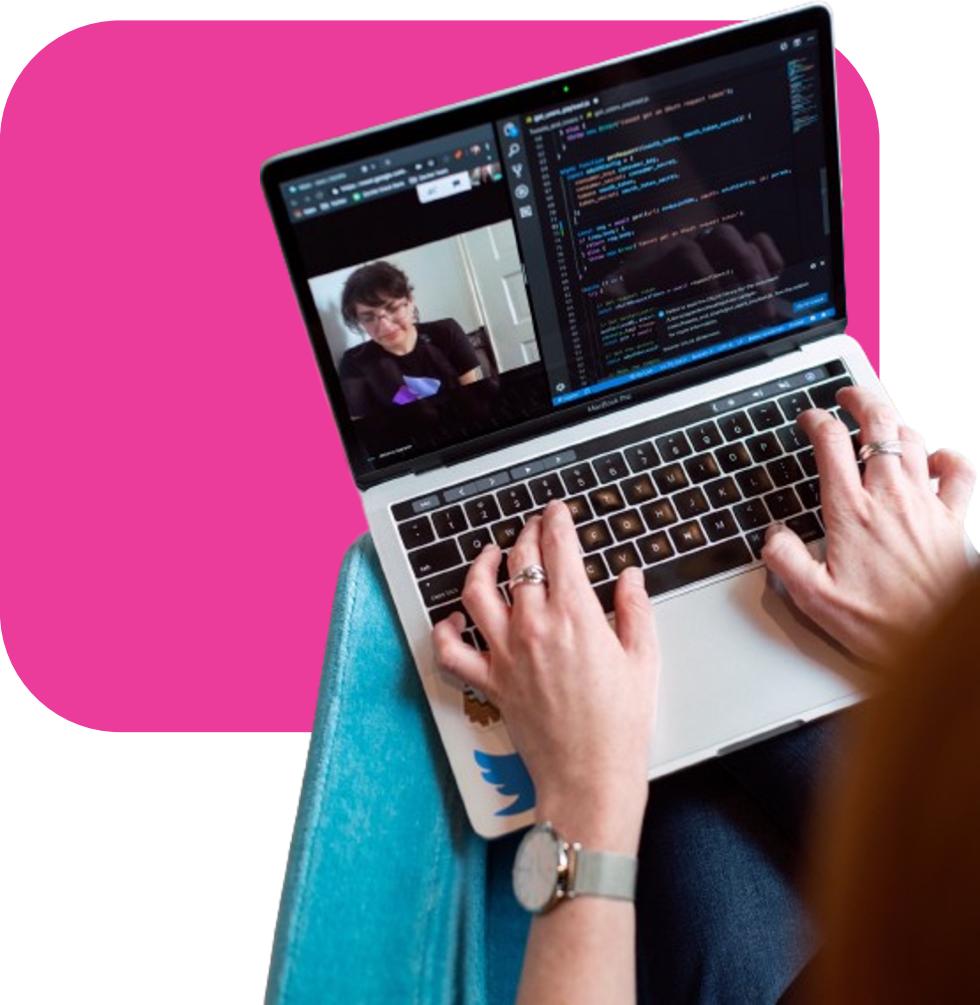
```
True
True
True
False
>>>
```

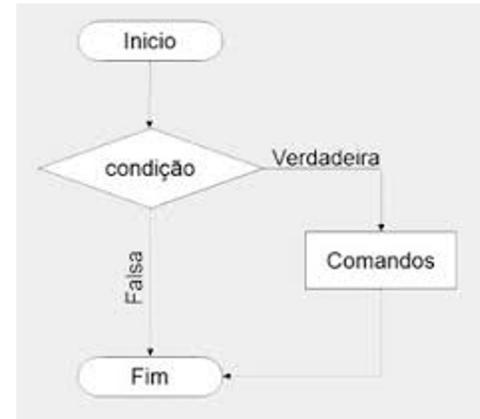
```
aula5.py X
1 # Escreva o seu código aqui :-)
2 print(not True)
3 print(not False)
4 |
```

Em execução: aula5.py

```
False
True
>>>
```

# Estruturas condicionais em Python



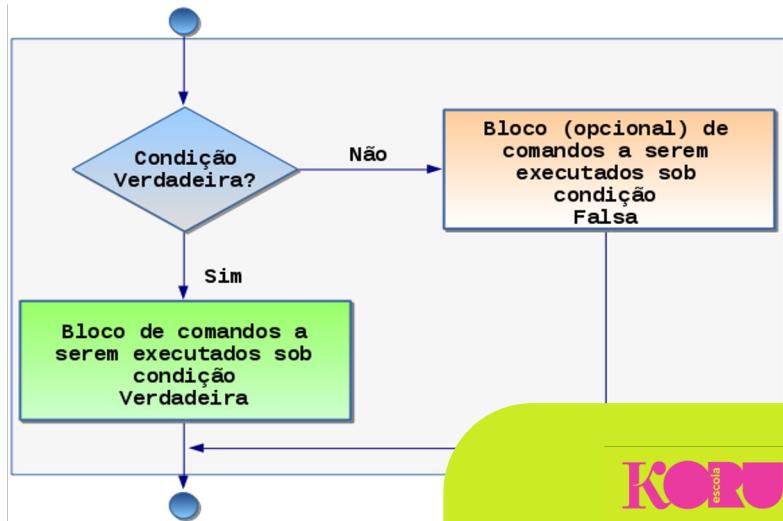


## Estruturas Condicionais

Permite escolher entre duas ações diferentes a depender do resultado de uma condição

# Condicional

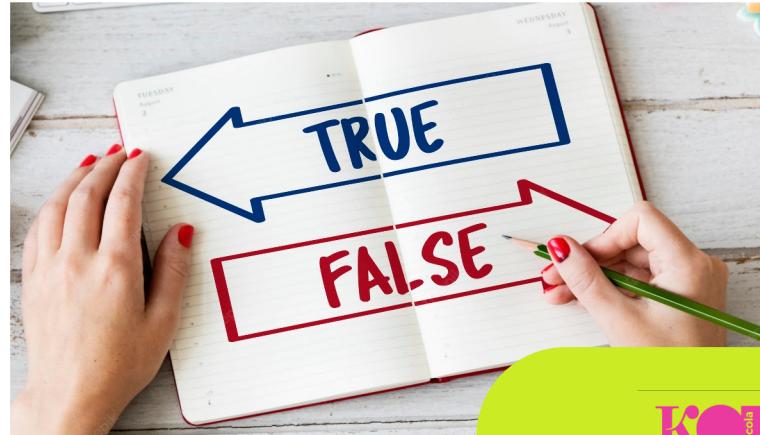
- ◎ **SE** <condição é verdadeira>:
  - Faça a atividade 1
- ◎ **SENÃO**:
  - Faça a atividade 2



# Condicional

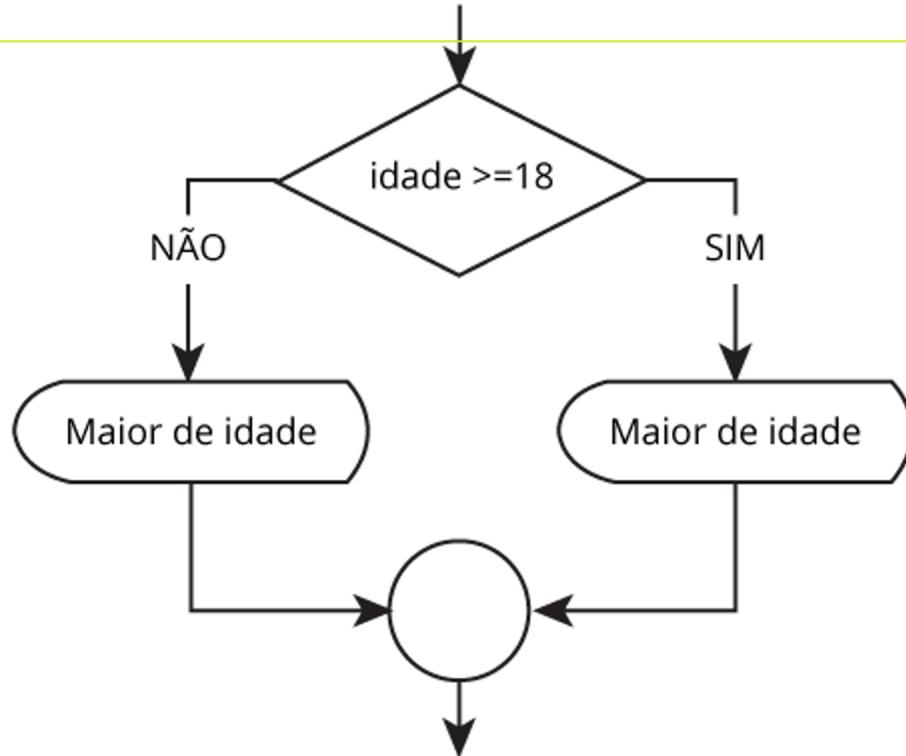
True

- ◎ SE ~~«condição é verdadeira»~~:
  - Faça a atividade 1
- ◎ SENÃO:
  - Faça a atividade 2



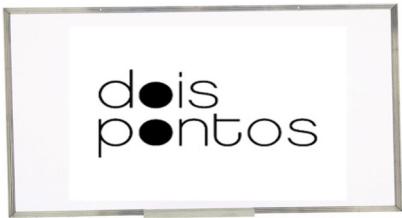
# Condicional

- ◎ SE idade  $\geq 18$ :
  - Pode beber
- ◎ SENÃO:
  - Não beba

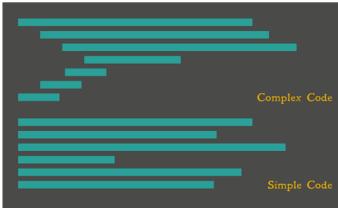


# Condicional em Python

se = if  
senão = else



## Indentação



```
[35] idade = int(input("Quanto anos vocês tem? "))
if idade >= 18:
    print("Pode dirigir")
else:
    print("Não pode")
```

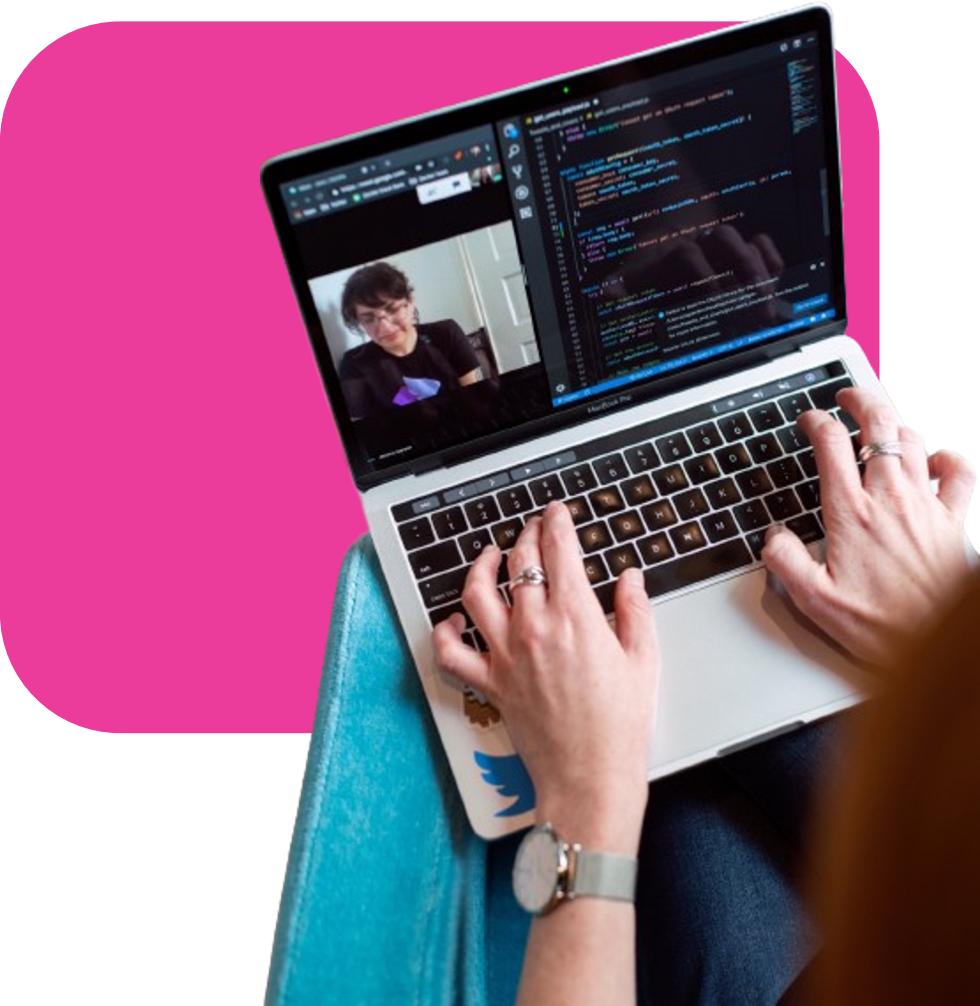
```
Quanto anos vocês tem? 24
Pode dirigir
```

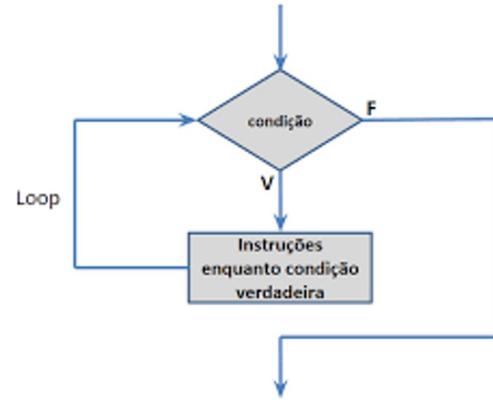
```
idade = int(input("Quanto anos vocês tem? "))
carteira = input("Tem carteira? (s/n) ")

if idade >= 18 and carteira == 's':
    print("Pode dirigir")
elif idade >= 18:
    print("Pode entrar na autoescola")
else:
    print("Não pode")
```

```
Quanto anos vocês tem? 24
Tem carteira? (s/n) n
Pode entrar na autoescola
```

# Estruturas de repetição em Python



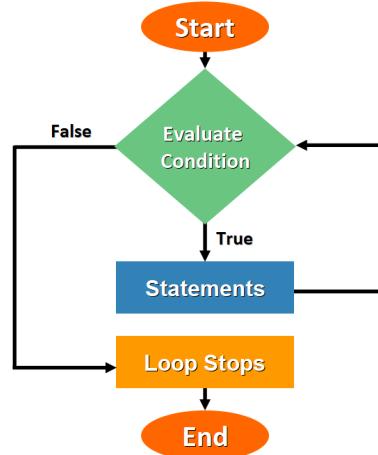


# Estruturas de Repetição

Permite repetir um bloco de programa  
diversas vezes a depender do resultado  
de uma condição

# Comando while

- While = Enquanto
- Enquanto True:
  - Repita esse pedaço de programa!



# Comando while em Python

- Usamos também dois pontos e identação
- Podemos colocar alguma condição de saída específica, por exemplo o usuário digitar sair
- Ou utilizar a ideia de contador, com uma variável numérica que é acrescida de um a cada iteração do laço

```
[41] entrada = input("sair ou ficar? ")
      while entrada == 'ficar':
          entrada = input("sair ou ficar? ")

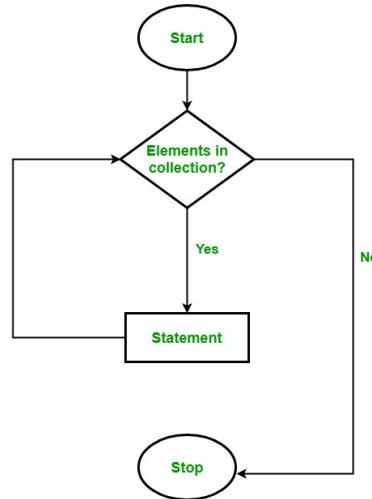
sair ou ficar? ficar
sair ou ficar? ficar
sair ou ficar? sair
```

```
[39] cont = 0
      while cont < 5:
          print(cont)
          cont = cont+1

0
1
2
3
4
```

# Comando for

- For = Para todo
- Realiza as repetições (iterações) por um conjunto de dados
- Para todo *ELEMENTO* em *CONJUNTO*:
  - Repita esse pedaço de programa!



# Comando for em Python

- Usamos também dois pontos e identação
- Objetos iteráveis (range)
- Exemplo: percorrer uma lista

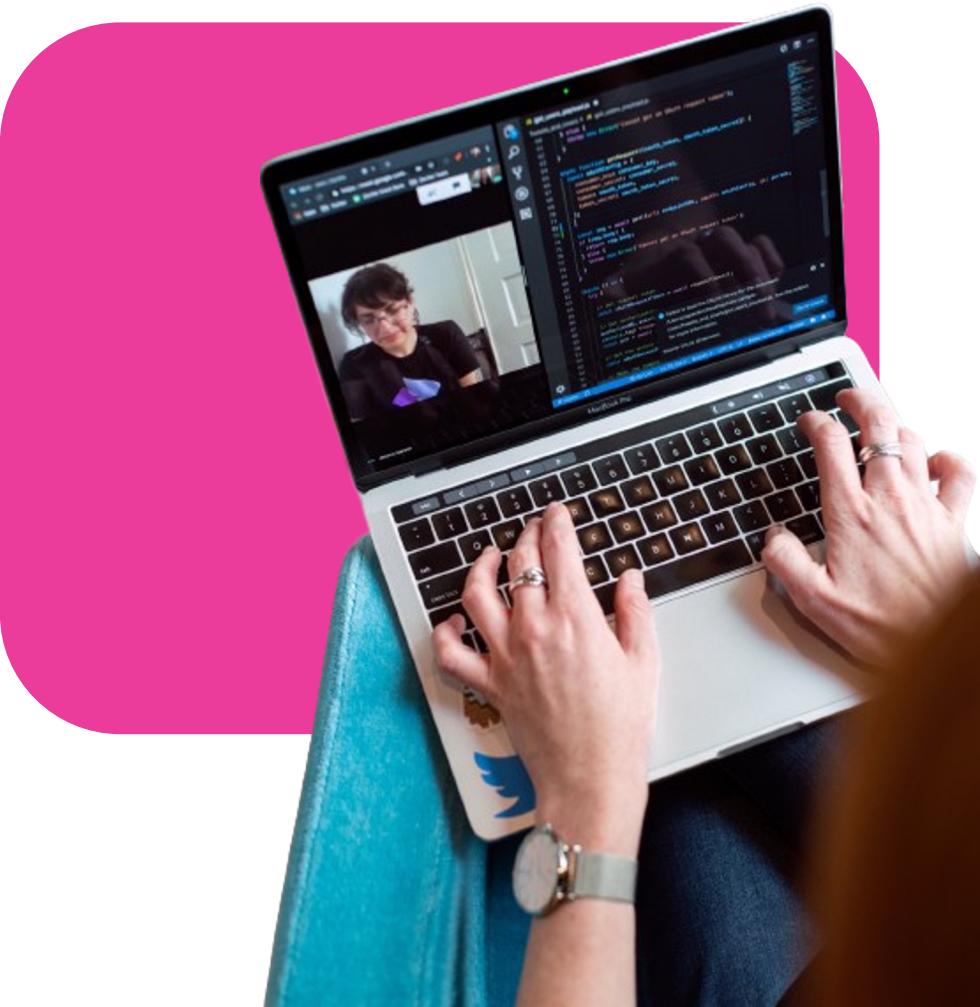
```
[43] for i in range(5):
      print(i)

0
1
2
3
4
```

```
frutas = ["banana", "maçã", "pera", "uva", "melancia"]
for f in frutas:
    print(f)

banana
maçã
pera
uva
melancia
```

# Funções em Python



# Funções

Bloco de Código que só é executado  
quando explicitamente chamado



# Funções

- Você pode passar dados, conhecidos como parâmetros ou argumentos, para uma função.
- Uma função pode retornar dados como resultado.



# Funções em Python

- Usamos a sintaxe def para definir a função
- Colocamos o nome dos argumentos a serem recebidos entre parênteses
- E a saída é colocada no return

```
[44] def multiplica_e_soma(a, b, c):
        resultado = a * b
        resultado += c
        return resultado

multiplica_e_soma(2, 2, 2)
```

6

# Funções Python

- Outros elementos de funções em Python são:
- Argumentos arbitrários
- Parâmetros com valor padrão
- Recursão

```
[48] def my_function(x=10):
      return x
```

```
print(my_function())
print(my_function(5))
```

```
10
5
```

```
[46] # Parâmetros arbitrários
def my_function(*kids):
    print("The youngest child is " + kids[2])
```

```
# Parâmetros arbitrários nomeados
def my_function2(**kid):
    print("His last name is " + kid["lname"])

my_function2(fname = "Tobias", lname = "Refsnes")
```

```
The youngest child is Linus
His last name is Refsnes
```

```
[51] def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 1) + fib(n - 2)
```

```
fib(7)
```

```
13
```

# Agora vamos avaliar a aula?



Obrigada !