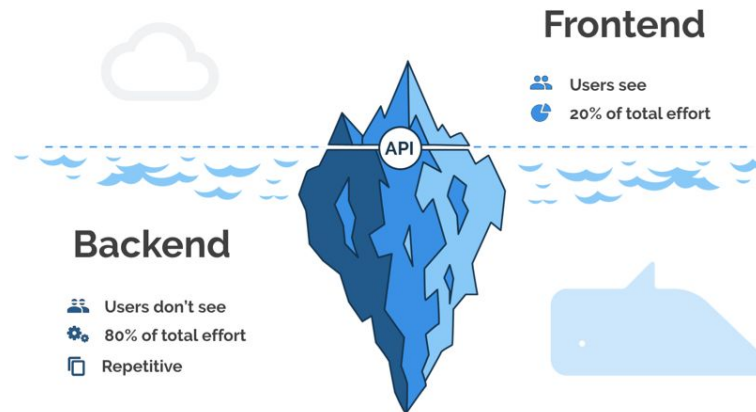


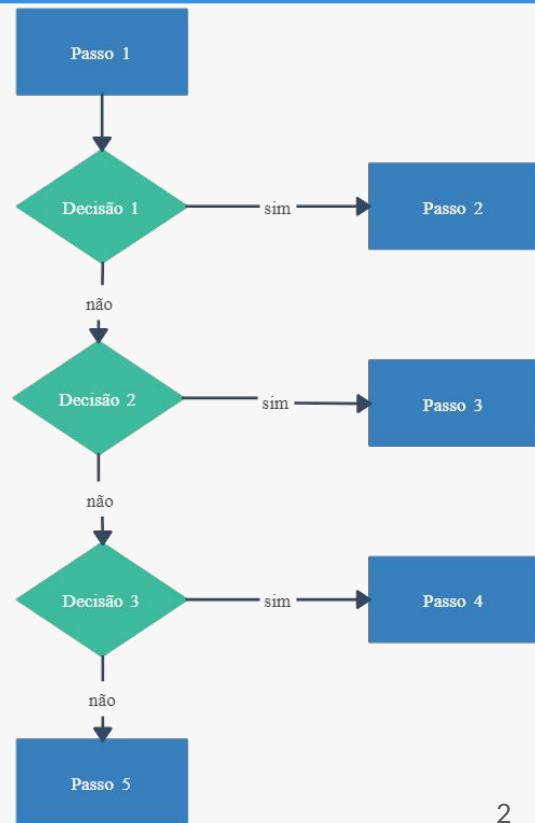
Objetivos da Aula:

- Conhecer métodos sobre strings
- Conhecer métodos sobre listas e dicionários.
- Ensinar os fundamentos sobre loops.




Estruturas Condicionais

```
idade = 18
if idade >= 18:
    print("Maior de idade")
elif idade == 17:
    print("Quase lá")
else:
    print("Menor de idade")
```




Estruturas de Repetição com for



```
for i in range(5):  
    print(i)
```

Nos tipos de laço for, você geralmente define logo no início: a inicialização e a **condição de parada**.

```
frutas = ["maçã",  
          "banana", "uva"]
```



```
for fruta in frutas:  
    print(fruta)
```

Estruturas de Repetição com while

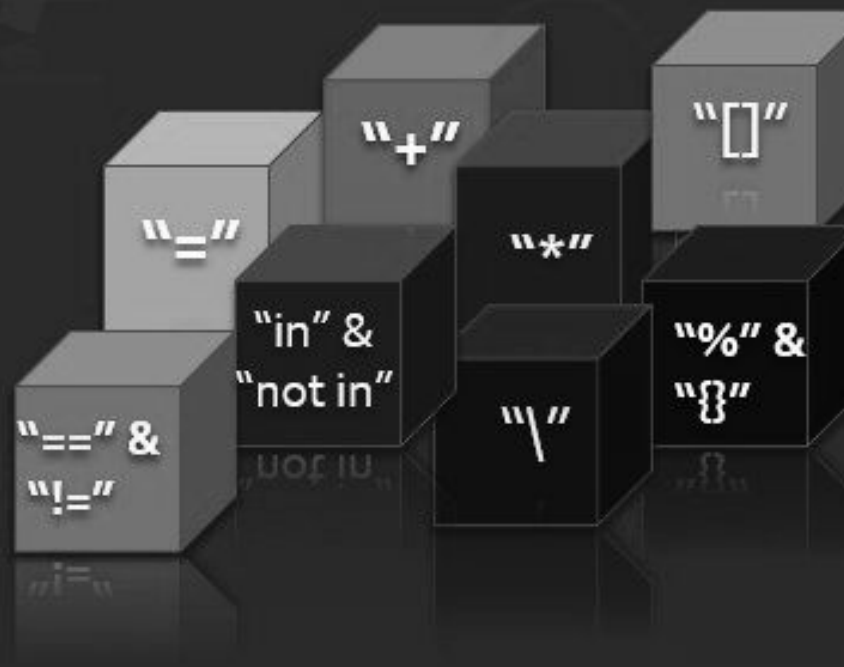
```
contador = 0
while contador < 5:
    print(contador)
    contador += 1
```

O laço while normalmente é mais simples, pois você define somente qual é a **condição de parada**. Isso significa que a inicialização e atualização deve ser feita em algum outro lugar, só use com cuidado para não ter um loop infinito.

Estruturas de Repetição com break e continue

```
for i in range(10):  
    if i == 5:  
        break # Interrompe o loop  
    if i % 2 == 0:  
        continue # Pula a iteração  
    print(i)
```

String Operators in Python



— converter todas as letras em **minúsculas**

```
"HEllo".lower() → hello
```

converter todas as letras em **Maiúscula**

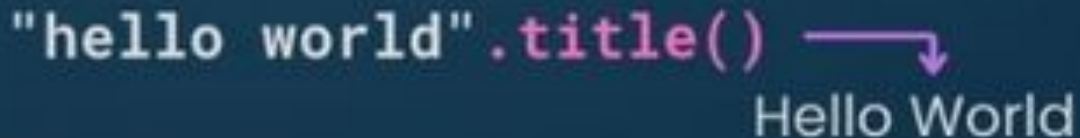
```
"hello".upper() → HELLO
```

converter a primeira letra em: **Maiúscula**




```
"hello world".capitalize() → Hello world
```

converter as primeiras letras de cada palavra em: **Maiúscula**



```
"hello world".title() → Hello World
```

```
" hello ".strip() → "hello"
```

remove os espaços da string
pode ser especificado
com direita **rstrip()**
ou esquerda **lstrip()**

verifica se a string iniciar com

```
"Hello".startswith("He") → True
```

verifica se a string termina com

```
"Hello".endswith("lo") → True
```


Métodos para manipular strings

```
" hello ".strip() → "hello"
```

remove os espaços da string
pode ser especificado
com direita **rstrip()**
ou esquerda **lstrip()**

Métodos para manipular strings

```
"one, three".replace(",", " | ")  
one| three
```

A purple bracket on the right side of the code block connects the comma in the first argument of the replace method to the resulting vertical bar in the output string.

altera todas as entradas do primeiro
argumento no segundo, no caso vírgula em
barra vertical

Métodos para manipular strings

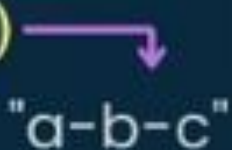
```
"one,three".split(", ") →  
['one', 'three']
```

converte a string em uma lista, onde a vírgula é o separador dos itens, mas pode ser um separador padronizado

Métodos para manipular strings

X =

```
"-".join(["a", "b", "c"])
```



"a-b-c"

converte uma lista em uma string,
especificando um separador ou nenhum


Procura a posição do item, se a posição não existe retorna -1

```
"hello".find("e") → 1
```

Procura a posição do item, se a posição não existe emite um erro de exceção

```
"hello".index("e") → 1
```

conta quantas vezes o item existe na string



```
"hello world".count("o") → 2
```

verifica se a string é numérica

```
"12345".isnumeric() → True
```


Python List Methods

list

- append()
- clear()
- copy()
- count()
- extend()
- index()
- insert()
- pop()
- remove()
- reverse()
- sort()



Listas

Listas são coleções heterogêneas de objetos, que podem ser de qualquer tipo, inclusive outras listas.

Você pode colocar qualquer informação que quiser em uma lista.

lista = ['a', 'b', 23, 'z']

colchetes

e separação por vírgulas



Listas

```
lista = ['a', 'b', 23, 'z']
```

`print(lista)` mostra a lista inteira

```
print(lista[0])
```



acessa a posição específica

Listas índices começa em 0

```
lista = ['a', 'b', 23, 'z']
```

```
print(lista )
```

mostra a lista inteira

```
print(lista[0] )
```



acessa a posição específica

```
print(lista[-1] )
```



último item

Listas exibindo uma mensagem

```
lista = ['a', 'b', 23, 'z']
```

```
mensagem = f 'ele tem { lista[2] } anos'
```

formatando a string



chaves para variáveis

```
print(mensagem)
```

Listas adicionando itens

```
lista = ['a', 'b', 23, 'z', 'x']
```

```
lista.append('x')
```

adicionando ao final da lista

```
lista = ['a', 'b', 'y', 23, 'z']
```

```
lista.insert(2, 'y')
```

adicionando em uma posição

Listas removendo itens

`lista = ['a', 'b', 23, 'z', 'x']`

`del lista[4]`



remove da posição

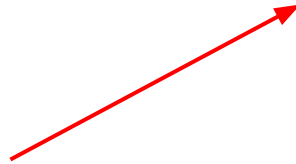
`lista = ['a', 'b', 23, 'z', 'x']`

`lista.remove(23)`

remove pelo valor

`lista = ['a', 'b', 'y', 23, 'z']`

`lista.pop()`



remove o último item

Listas ordenando itens

```
lista = ['c', 'b', 'a', 'z', 'x']
```

```
lista.sort()
```

ordena a lista em ordem

```
lista = ['c', 'b', 'a', 'z', 'x']
```

```
lista.sort(reverse=True)
```

ordena a lista em ordem

inversa

Listas mostrar a quantidade de itens

```
lista = ['c', 'b', 'a', 'z', 'x' ]
```

```
len(lista)
```

constar uma lista

Listas percorrer a lista

```
lista = ['c', 'b', 'a', 'z', 'x' ]
```

```
for x in lista:
```

```
    print(x)
```

exibindo itens individuais

Python Dictionary Methods



Em Python, dicionários, também conhecidos como "dicts", assumem o papel de estruturas de dados robustas e versáteis. Sua função primordial consiste em armazenar coleções de dados na forma de pares ordenados, compostos por chave e valor.

```
carro = {
```

```
    "marca" : "Ford",  
    "modelo" : "ka",  
    "ano" : 2012
```

```
}
```

Separação entre chave e valor

Key
chave

values
valores

```
Q = { 'NAME': 'JOHN', 'AGE': 43 }
```

GET

Resgata o valor pela chave

```
q.get('NAME')  
'JOHN'
```

```
Q = { 'NAME': 'JOHN', 'AGE': 43 }
```

KEYS

Resgata somente as chaves

```
q.keys()  
[ 'NAME', 'AGE' ]
```

```
Q = { 'NAME': 'JOHN', 'AGE': 43 }
```

VALUES

Resgata somente os valores

```
q.values()
```

```
['JOHN', 43]
```

```
Q = { 'NAME' : 'JOHN', 'AGE' : 43 }
```

Update

Adiciona uma nova chave e um novo valor no final

```
q.update({ 'STATE' : 'CA' })
```

```
{ 'NAME' : 'JOHN', 'AGE' : 43,  
  'STATE' : 'CA' }
```

`q["cor"] = "azul"`


```
Q = { 'NAME': 'JOHN', 'AGE': 43 }
```

Items

Converte chave e valor em
tupla, dentro de uma lista
para ser iterável

```
q.items()
```

```
[('NAME', 'JOHN'), ('AGE', 43),  
( 'STATE', 'CA' )]
```

```
Q = { 'NAME' : 'JOHN', 'AGE' : 43 }
```

SET DEFAULT

Retorna o valor da chave da primeira posição se ele não existir cria a chave e o valor

```
q.setdefault('PIN', 58796)
```

```
{ 'NAME' : 'JOHN', 'AGE' : 43,  
  'STATE' : 'CA' 'PIN' : 58796 }
```

```
Q = { 'NAME': 'JOHN', 'AGE': 43 }
```

Pop

remove a chave e o valor, se
passado a chave como
argumento

```
{ 'NAME': 'JOHN', 'AGE': 43,  
  'STATE': 'CA' }
```

```
q.pop('STATE')
```

```
{ 'NAME': 'JOHN', 'AGE': 43 }
```

`del q["AGE"]`

```
Q = { 'NAME' : 'JOHN', 'AGE' : 43 }
```

Pop Item

remove o último item “chave e valor”

```
q.popitem()
```

```
{ 'NAME' : 'JOHN' }
```

```
Q = { 'NAME': 'JOHN', 'AGE': 43 }
```

Clear

remove todos os itens do
dicionário

```
q.clear()  
  
{ }
```

Listas exercícios

crie um programa para exibir uma **lista** de **roupas e seus valores**
(lista fixa) * para facilitar pode ser **duas listas**

depois exiba uma **opção de escolher** o produto pelo **número** e ao
produto escolhido salvar em **outra lista** (carrinho)

e **perguntar** se quer comprar mais, **por fim** exibir os produtos
comprados

Listas exercícios vale = 0.5 na p1

crie um programa que tenha **um valor fixo** que é seu dinheiro. ex. 1000, depois o programa irá pergunta, **o nome do produto, o preço do produto, a quantidade do produto**, e perguntar **se quer adicionar mais itens**.

no final deve exibir o **novo saldo da carteira** e o **carrinho de compras**
dica: 4 variáveis e lista = carrinho, **o valor pago**

Referências

BEAZLEY D.; JONES, B.K. Python Cookbook: Receitas para dominar Python. 3 ed. São Paulo: Novatec, 2019.

CRUZ, FJ. Python: escreva seus primeiro programas. 1.ed. São Paulo: Casa do Código, 2021.

PEREIRA, E, DOUGLAS MICHAEL. Trilhas Python: Programação multiparadigma e desenvolvimento web com Flask. 1.ed. São Paulo: Casa do Código, 2020.

