TÍTULO: Estruturas e tipos de dados em Python

CENÁRIO DE APRENDIZAGEM		
Escola:	***Duração (minutos):***	**45**
Professor(a):	***Idade dos alunos:***	**13**

Ideia Chave:	**Vamos conhecer estruturas e tipos de dados em Python**

Tópicos:

- Os alunos aprofundam a sua compreensão do uso de vários softwares e políticas.

Objetivos:

- Os alunos serão capazes de projetar e criar programas que utilizam sub-rotinas, estruturas e tipos de dados apropriados, expressões, variáveis e comandos interativos e condicionais.
- Linguagens de programação gerais são usadas para criar programas.
- Os alunos compreendem as diferentes maneiras de usar simulações e algoritmos de organização passo a passo para resolver problemas.

Resultados:

- Os alunos criam um jogo, aplicativo ou aplicativo móvel mais complexo que resolve um problema particular de um assunto ou tópico específico.
- Os alunos aprendem a delinear a operação de um programa mais complexo em vários padrões e generalizações.

Formas de trabalho:

- trabalho individual
- trabalho em pares
- trabalho de grupo

Métodos:

- apresentação
- discussão

- exercício interativo

ARTICULAÇÃO
Linha de atuação (duração, minutos)
INTRODUÇÃO O professor explica e inicia a discussão com os alunos sobre as variáveis. Enquanto resolvemos problemas no nosso computador, criamos programas que usam diferentes variáveis de entrada. Entende-se por variável uma área de memória onde armazenamos um valor qualquer. Portanto, enquanto se projeta uma solução, um dos primeiros passos é rever essas mesmas variáveis. Vamos conhecer diferentes tipos de dados usados no Python. Nos nossos primeiros programas de computador no Python, usamos dados numéricos e de texto. Os dados de texto foram escritos usando apóstrofos (‘), que delimitam os “strings” (palavras ou cadeias de caracteres). Os tipos de dados usados nas soluções do computador são importantes na aplicação de comandos adequados de entrada e saída de dados, bem como nas diferentes funções que adicionalmente ajustam os dados. No Python há três tipos diferentes de dados simples: **- Numéricos (números).** **- Caracteres (strings).** **- Lógicos (booleanos).**
PARTE PRINCIPAL **Tipo de dados numéricos** Ao começar a aprender a codificar e a arranjar novas soluções no computador com a linguagem Python, ficamos a conhecer diferentes tipos de dados como um número inteiro, caracteres e valores lógicos. Também ficamos a conhecer comandos para a entrada e saída desses valores usando um programa de computador. No entanto, os números que utilizamos no dia a dia não têm de ser inteiros – por exemplo podem ser decimais. Vamos então ver como podemos usar números decimais no Python. Os números decimais são escritos como em matemática, como dois números inteiros separados por um ponto, por exemplo 4.56.

```
>>> 10
10
>>> - 10
-10
>>>
```

```
>>> 4.56
4.56
>>> .25
0.25
>>>
```

Que operações podem ser usadas com números decimais?
A capacidade de separar decimal e parte inteira do número decimal é muito interessante na programação e, por causa disso, muitas linguagens de programação têm uma função incorporada que permite exatamente isso.
Como converter um decimal em um número inteiro?
Vejamos alguns exemplos.

```
>>> int(4.2)
4
>>> int(4.8)
4
>>>
```

```
>>> float(5)
5.0
>>>
```

A função **int ()** converte um decimal em um número inteiro, desconsiderando a parte decimal do número e deixando apenas a parte inteira.
A função **float ()** converte um inteiro em um número decimal adicionando um zero após o ponto.

Tipo de dados lógicos

O tipo de dados lógico é comumente usado em comandos em que observamos um valor de algumas condições lógicas. Uma condição lógica é uma construção lógica em que estamos verificando o valor de algo. Geralmente, ele pode ter apenas dois valores: True (verdadeiro) ou False (falso). Em comandos de desvio (if, if-else, elif-else) e na repetição condicional (while), aprendemos que, dependendo da condição lógica, o programa está tomando a decisão de continuar funcionando. Se quisermos armazenar um valor lógico em uma variável, podemos-lhe atribuir um valor lógico.

```
>>> a=4<5
>>> print(a)
True
>>>
```

```
>>> b=5<2
>>> print(b)
False
>>>
```

```
>>> b=False
>>> print(b)
False
>>>
```

Strings

Os dados strings são um tipo básico de dados para armazenar valores textuais. Eles representam palavras ou frases dentro de aspas (“) ou semi-aspas (‘).

```
>>> a='python'
>>> print(a)
python
>>>
```

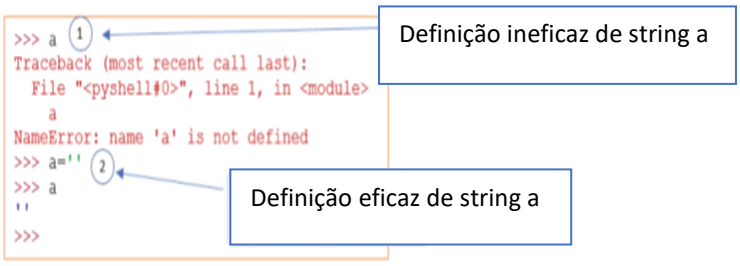
```
>>> b="dobar dan"
>>> print(b)
dobar dan
>>>
```

Também aprendemos como aplicar operadores matemáticos em uma *string*, por exemplo, adição ou multiplicação.

```
>>> x='abc'
>>> y='def'
>>> x + y
'abcdef'
>>>
```

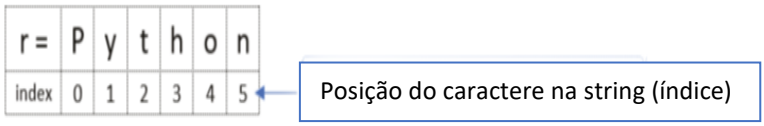
```
>>> x * 4
'abcabcabcabc'
>>>
```

Adicionando dois caracteres, a string cria uma nova string que contém os caracteres originais.
Multiplicando uma string com um número vai multiplicar a string. Uma string não pode ser multiplicada por outra string – temos de usar um número inteiro.
Uma string pode ainda ser definida como vazia, quando não se escrevem os caracteres entre as semi-aspas.



Indexando uma string(“linha”)

Uma *string* pode ser vista como vários caracteres classificados em uma determinada ordem. A atribuição de números às posições dos caracteres é chamada de indexação. Esse é o procedimento em que cada caractere é atribuído com um número de índice ou, em outras palavras, um número que descreve onde o caractere está em uma *string*.
Em Python, o primeiro caractere em uma *string* é sempre atribuído ao número zero (0) e, em seguida, a cada caractere seguinte são atribuídos os números 1, 2, 3, 4 ...



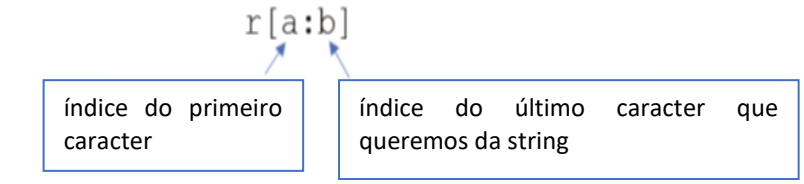
EXCERCÍCIO 1

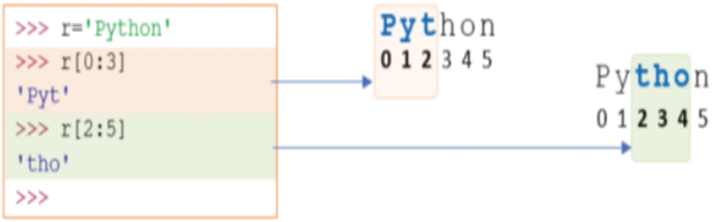
Usando a interface de usuário interativa, pegue o primeiro e o último caractere na *string* "Dubrovnik".

```
>>> r='Dubrovnik'
>>> r[0], r[8]
('D','k')
```

Agarrando vários caracteres ao mesmo tempo - uma parte da *string*

Se quisermos pegar várias partes de uma *string*, entre os colchetes de uma *string* podemos escrever os índices da primeira e da última parte da *string* que queremos agarrar e separá-la com dois pontos (:).

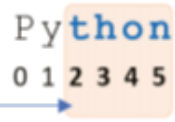




Existem várias maneiras de capturar partes específicas de *strings*, dependendo se queremos que essas partes incluam o primeiro ou o último caractere na *string*. Uma parte de uma *string* que inclui o primeiro caractere pode ser capturada sem os índices 0, por exemplo, em vez de r [0: 3], podemos escrever r [: 3]. o mesmo vale para o último caractere da *string*, por exemplo, podemos escrever r [2:].

```
>>> r='Python'
>>> r[:3]
'Pyt'
>>>
```


```
>>> r='Python'
>>> r[2:]
'thon'
>>>
```



Se não definirmos os pontos de partida e de chegada de uma string, Python apenas levará a string inteira.

```
>>> r[:]
'Python'
>>>
```

EXCERCÍCIO 2

De acordo com o exemplo anterior, os alunos podem projetar, criar e testar seus próprios exemplos.

CONCLUSÃO

Alunos e professor discutem e avaliam as soluções apresentadas.

Métodos		***Formas de trabalho***
apresentação	entrevista	Trabalho individual
discussão	demonstração	Trabalho em pares
trabalhar no texto	representação	trabalho em equipa/grupo
trabalho gráfico		trabalho frontal
exercício interativo / simulação no computador		

Material:

-

Bibliografia

-

OBSERVAÇÕES PESSOAIS, COMENTÁRIOS E NOTAS