



Fundamentos de Programação

Operações em STRINGS

Aula 08

Post It da Aula Passada

Expressões Relacionais

- Operandos
 - Numéricos
 - Outros
- Operadores
 - `==; !=; >; <; >=; <=`
- Resultado
 - Lógico

Expressões Lógicas

- Operandos
 - Lógicos
- Operadores
 - `not; and; or`
- Resultado
 - Lógico

Fundamentação em *Strings*

Anteriormente, estudamos as *strings* quanto a um tipo de dado que se dedica a trabalhar com cadeias de caracteres

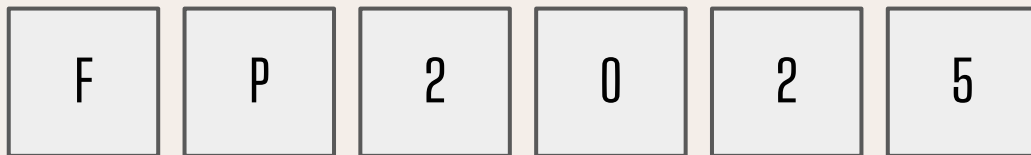
“Eu sou uma string”

Porém, esse tipo de dado, em Python, apresenta diversas características e operações exclusivas que podem ser exploradas pelo programador

Fundamentação em *Strings*

em Python, podemos compreender uma *string* como uma **sequência de nichos em uma parede**, formando uma estante horizontal

Cada um dos nichos suporta um símbolo que, em ordem, formam o dado



turma = “FP2025”

Fundamentação em *Strings*



Fazer a atribuição da *string* na variável *turma* é como rotular o conjunto de nichos, permitindo nos referimos diretamente a ele

```
print(turma)  
type(turma)
```

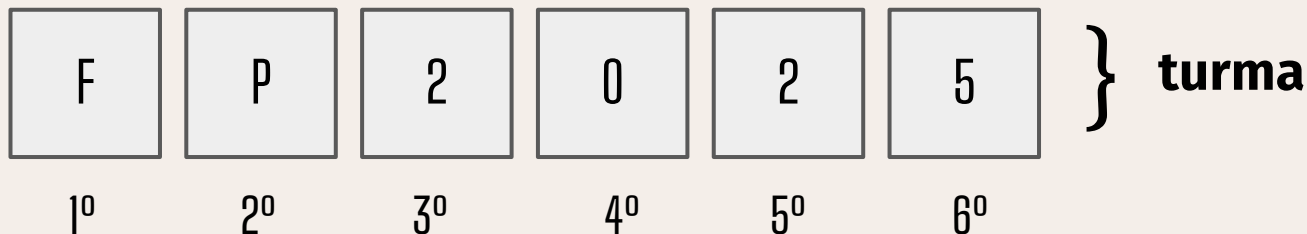
Fundamentação em *Strings*



Porém, a variável `turma` se refere ao conjunto de nichos por inteiro! **E se for necessário consultar apenas um caractere da *string*?**

No exemplo dos nichos, isso seria equivalente a observar apenas um nicho em particular, ignorando os demais

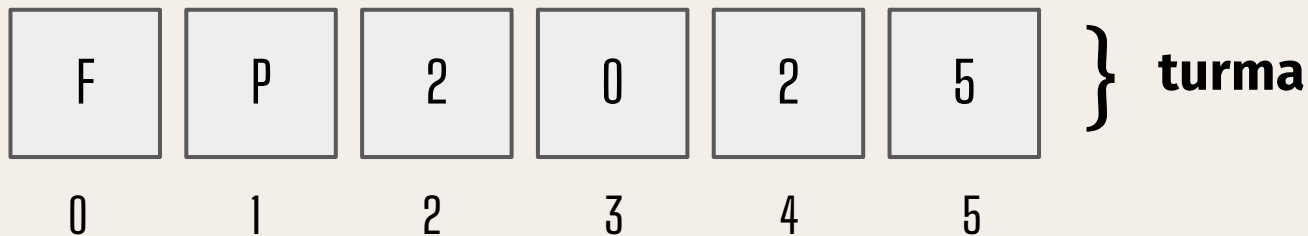
Indexação de *Strings*



Para organizarmos os nossos nichos, podemos definir índices para os mesmos conforme as suas posições:

- O primeiro (1º) nicho contém o caractere "F"
- O terceiro (3º) nicho contém o caractere "2"
- O quarto (4º) nicho contém o caractere "0"

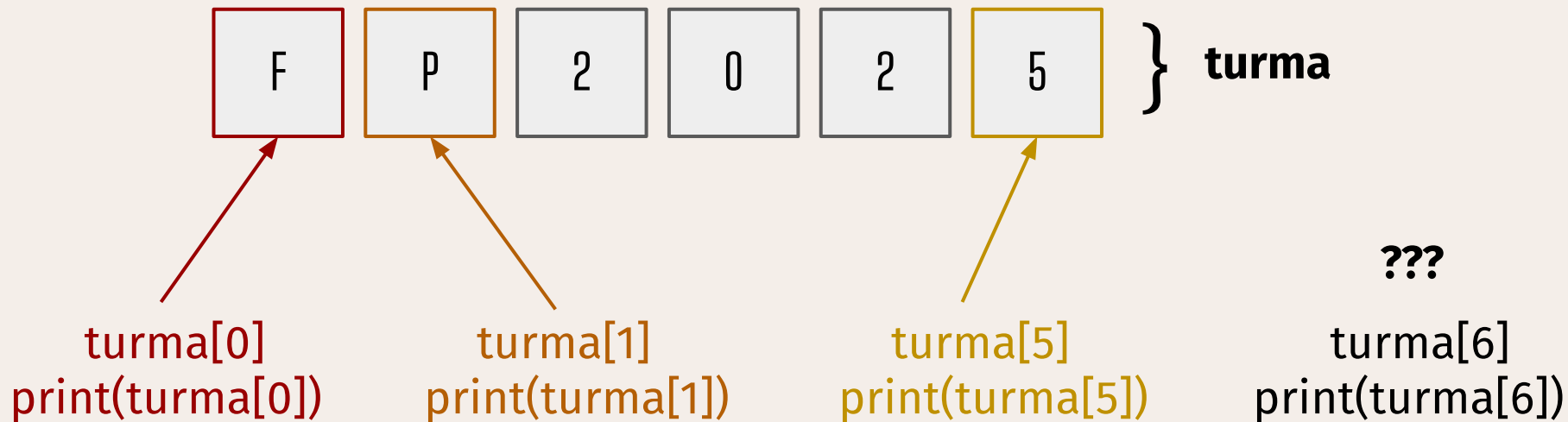
Indexação de *Strings*



Em Python, o mesmo acontece! A exceção é pelos índices das posições, sendo sempre **numéricos e iniciados em 0 ao invés de 1** (o primeiro índice é 0)

Indicamos o acesso a um índice em específico em Python inserindo o valor do mesmo entre colchetes, ao lado do dado do tipo *string*: turma**[0]**

Indexação de *Strings*



Tamanho de *Strings*

Sabendo que existe um limite de posições/índices acessíveis em uma *string*, a pergunta que surge é: **como saber quantas posições/caracteres contém uma *string* qualquer?**

A função nativa chamada *len* pode nos trazer tal informação. Tal função apresenta o seguinte protótipo:

```
len("Uma string qualquer")
```

O argumento provido pode ser, também, uma variável contendo um dado do tipo *string*

Atribuições em *Strings*

O tipo de dados *string* **não permite a atribuição de caracteres** em posições específicas:

turma[5] = "4" (**NÃO PODE SER FEITO**)

Entretanto, outros caracteres e *strings* podem ser adicionados ao início e ao final de uma *string* específica:

CONCATENAÇÃO

Concatenação de *Strings*

A concatenação de *strings* acontece através da utilização do **operador de adição entre duas *strings***:

`"abc" + "def"`

O resultado da concatenação será **uma nova *string*** contendo os caracteres, em ordem, das *strings* concatenadas:

`"abcdef"`

Note que o operador de adição para concatenação só funciona entre *strings*!

Concatenação de *Strings*

Podemos manipular *strings* **via caracteres específicos de seu início ou final**: a chamada manipulação *head/tail*

“Isso é uma string”[n:m]

Para fazer isso, **utilizamos a mesma notação []** de acesso a uma posição específica da *string*

Porém, agora indicamos um **intervalo de caracteres que será adicionado (m) e removido (n) ao início** da *string*

Concatenação de *Strings*

`"Isso é uma string"[n:m]`

`"Isso é uma string"[:4]` -> `"Isso"` (Adicione apenas os 4 primeiros caracteres)

`"Isso é uma string"[11:]` -> `"string"` (Remova os primeiros 11 caracteres)

`"Isso é uma string"[7:10]` -> `"uma"` (Adicione os primeiros 10 caracteres e remova os primeiros 7 ou adicione os caracteres com índices 7, 8 e 9)

O **retorno da manipulação *head/tail* é uma nova *string*** de forma que a *string* original se mantém intacta

Tabulação e Quebra de Linha

O tipo de dado *string* assume um caractere de escape para a inserção de caracteres especiais em uma cadeia: \

Existem vários **caracteres especiais** que podem ser incluídos através do símbolo de escape, porém os mais comuns são:

- \t: tabulação, equivalente a inserir um “*tab*” nos editores de texto
- \n: quebra de linha, equivalente a um “*enter*” nos editores de texto
- \\: Contra barra, utilizado para inserir o caractere de contra barra

Tabulação e Quebra de Linha

Vejamos alguns exemplos:

1. `print("\tEssa string apresenta tabulação!")`
2. `print("Essa string é formada\npor duas linhas!")`
3. `print("Essa string mostra uma contra barra: \\")`

Resultados:

1. Essa string apresenta tabulação
2. Essa string é formada
por duas linhas!
3. Essa string mostra uma contra barra: \

Funções de *Strings*

O tipo de dado *string* dispõe de algumas funções nativas para a manipulação do mesmo

Essas funções, na verdade, são métodos da sua respectiva classe... porém, como não estamos estudando orientação a objetos, vamos considerar tais métodos como funções pré-implementadas para todas as *strings*

Funções de *Strings*

A notação para utilizar essas funções é chamada de **notação ponto**. O objeto da função (elemento onde ela será executada) aparece antes de um ponto, e a função a ser executada depois do ponto:

```
“abcdef”.função()
```

```
var_string = “abcdef”  
var_string.função()
```

Funções de *Strings*

A função **upper** retorna uma nova string que apresenta todas as letras da *string* original em letras maiúsculas:

```
"abc".upper()
```

Já a função **lower** retorna uma nova string contendo todas as letras da *string* original em letras minúsculas:

```
"ABC".lower()
```

Se não houver letras na *string*, ou se elas já estiverem no formato esperado, uma nova *string* idêntica a original é retornada

Funções de *Strings*

A função ***startswith*** verifica se uma dada *string* inicia com os mesmos caracteres de uma segunda *string*, a última provida como argumento da função:

```
"abcdef".startswith("abc")
```

De forma similar, a função ***endswith*** verifica se uma *string* termina com os mesmos caracteres de uma segunda *string* provida como argumento:

```
"abcdef".endswith("def")
```

O retorno dessas funções é um dado do tipo lógico (*bool*)

Funções de *Strings*

A função ***split*** separa uma dada *string* em uma ou mais *strings* originalmente separadas por um marcador, o último provido como argumento da função:

```
"ab/cd/ef".split("/")
```

O retorno da função *split* será um dado do tipo lista (*list*); iremos estudar tal tipo de dados nas próximas aulas da disciplina:

```
"ab/cd/ef".split("/") -> ["ab", "cd", "ef"]
```

Funções de *Strings*

A função ***format*** objetiva incluir *strings* em locais demarcados dentro de outras *strings*, essa demarcação é feita com chaves {}:

```
nome = input("Qual é o seu nome?:")  
sobrenome = input("Qual é o seu sobrenome?:")  
print("O nome do usuário é: {} {}".format(nome, sobrenome))
```

Assim, o resultado da função *format* que será exibido na tela (no meu caso) é:

```
O nome do usuário é: Vinícius Garcia
```

Funções de *Strings*

A função ***replace*** é empregada na criação de novas *strings* que baseadas em uma *string* original com caracteres específicos substituídos:

```
str_base = “Uma string qualquer”  
str_nova = str_base.replace(“a”, “e”)
```

Qual é o resultado?

A função *replace* pode também ser usada para substituir *substrings*:

```
str_nova = str_base.replace(“qualquer”, “específica”)
```

Funções de *Strings*

Outras funções de *string* úteis e suas respectivas maneiras de uso podem ser encontradas em:

https://www.w3schools.com/python/python_ref_string.asp

- index
- find
- isdecimal
- isalpha
- isalnum

- strip
- lstrip
- restrip
- capitalize
- count

Exercício #08

Faça um programa que leia uma sequência de caracteres sem acentos gráficos, remova os espaços da mesma e padronize-a com letras maiúsculas ou minúsculas. Finalmente, **verifique se a mesma é um palíndromo ou não** (para isso exiba o resultado de uma expressão relacional definida com o operador ==)

Dicas: você precisará das funções de *string* chamadas *upper* ou *lower* para padronizar as letras como maiúsculas ou minúsculas; *replace* para remover os espaços da *string*; e do *token* *[::-1]* colocado ao lado da *string* para executar a inversão da mesma



Fundamentos de Programação

Aula 08

Obrigado e
ATÉ A
PRÓXIMA!