Variáveis, Expressões e Instruções (String)

Disciplina: Programação para Engenharia

Professores: Cristiane Pavei Martinello Fernandes
Douglas de Medeiros Deolindo
Giovani Martins Cascaes
Joel Barbosa Panchyniak
Marcelo Marcos Amoroso
Marcos Antônio Jeremias Coelho
Ramon de Souza Coan

Praticando com o Python ...

1) Escreva um programa que receba 2 valores de x e y. Em seguida calcule e imprima o valor de z:

$$z = \frac{(x^2+y^2)}{x-y}$$

2) Construa um programa que receba do usuário a variação do deslocamento de um objeto (em metros) e a variação do tempo percorrido (em segundo). Ao fim, o programa deve calcular a velocidade média, em m/s, do objeto. Mostrar os dados fornecidos e o valor calculado.

Praticando com o Python ...

3) Escreva um programa que solicita o raio de um círculo e em seguida exiba o perímetro e área do círculo. Para saber o valor do pi, faça:

```
import math
print(math.pi)
```

4) Você está no Brasil, e para temperatura usamos o grau Celsius. Porém, quando você for contrato para trabalhar como programador Python no exterior, deverá usar graus Fahrenheit.

Ou seja, você fornece a temperatura em graus Celsius e seu script faz a conversão para graus Fahrenheit.

Praticando com o Python ...

5) Agora faça o contrário. Você fornece a temperatura em graus Fahrenheit, seu programa converte para Celsius e exibe na tela.

6) Um novo modelo de carro, super econômico foi lançado. Ele faz 20 km com 1 litro de combustível. Cada litro de combustível custa R$ 4,95.

Faça um programa que pergunte ao usuário quanto de dinheiro ele pretende usar e em seguida o programa informa quantos litros de combustível ele pode comprar e quantos quilômetros o carro consegue rodar com esta quantidade de combustível.

Seu script será usado no computador de bordo do carro.

Praticando com o Python ...

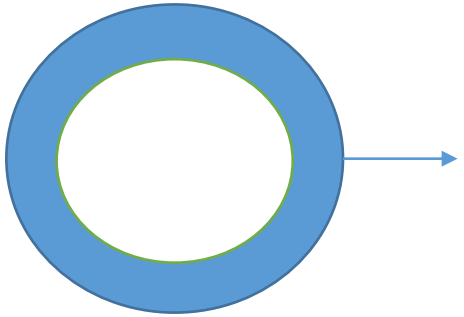
7) Sua tarefa é criar um programa em Python que pede o preço original de um produto e dá 20% de desconto. Você deve mostrar:

Preço original do produto
Valor do desconto em R$ (tipo 'Você ganhou R$ xx,xx de desconto’)
Valor do produto com o desconto

8) A loja percebeu que não quer dar 20% em tudo. Quer dar 20% em algumas coisas, 10% em outras, nada em outros produtos e até 30% em alguns outros produtos. Crie um script em Python que pergunte o preço original e o desconto que deve ser concedido. Ele deve mostrar a saída igual ao exercício anterior.

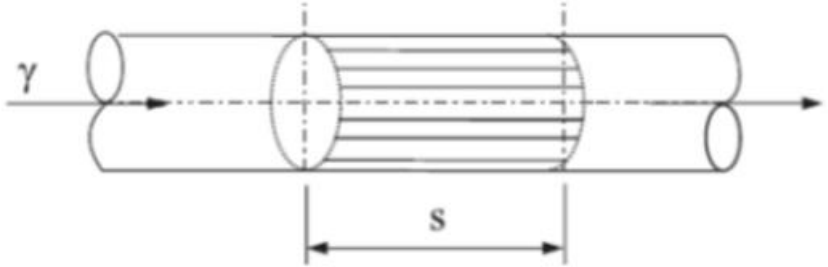
Praticando com o Python ...

9) Você foi contratado para desenvolver um programa que calcule área de uma coroa circular com base em duas medidas de raio fornecido pelo usuário.



área em azul

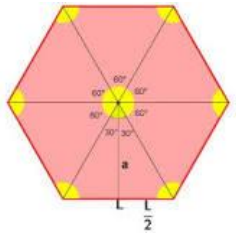
10) Você está desenvolvendo um programa para calcular a vazão de um fluido em um tubo com base no diâmetro interno do tubo e na velocidade do fluxo. A fórmula para calcular a vazão deve ser pesquisada. Os dados de entrada devem ser alimentados em metros e m/s.



Praticando com o Python ...

11) Você foi contratado para desenvolver um programa que calcule o Índice de Massa Corporal (IMC) com base nos dados de altura e peso fornecidos pelo usuário. O IMC é uma medida que relaciona o peso e a altura de uma pessoa para avaliar se ela está abaixo do peso, com peso normal, com sobrepeso ou obesa.
A fórmula para calcular o IMC é: IMC = peso / (altura^2), onde o peso é em quilogramas e a altura é em metros.

12) Você está desenvolvendo um programa para calcular a área de um hexágono regular com base no raio fornecido pelo usuário. Um hexágono regular tem seis lados de igual comprimento e seis ângulos internos de 120 graus. Assim, para determinar a área desse hexágono, basta determinar a área de um dos triângulos e, em seguida, multiplicar o resultado por 6.



Variáveis do tipo **string**

- Representam informação textual
- Uma **string** é uma sequência de caracteres simples
- Na linguagem Python, as strings são utilizadas com aspas simples ('... ') ou aspas duplas ("...")

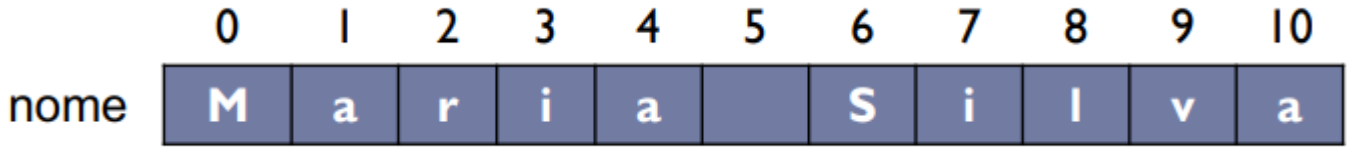
```
nome = "Maria Silva"
nacionalidade = "brasileira"
nome_mae = "Ana Santos Silva"
nome_pai = "Jonas Nunes Silva"
```

```
>>> print("Olá Mundo")
Olá Mundo
>>>
```

Acessando caracteres de uma **string**

- Caracteres podem ser acessados pela sua posição dentro da **string**
- Primeira posição é a posição ZERO

```
>>> nome = "Maria Silva"
>>> print(nome[0])
M
>>> print(nome[6])
S
```



Alteração de um caractere em **strings**

O conteúdo de uma determinada posição de uma **string** não pode ser alterado – são sequências imutáveis.

```
>>> nome = "Maria Silva"
>>> nome[3] = "t"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support
item assignment
```

Fatiamento das **strings**

- O fatiamento é uma ferramenta usada para extrair apenas uma parte dos elementos de uma **string**

string[Limite_Inferior : Limite_Superior]

- Retorna uma **string** com os elementos das posições do limite inferior até o limite superior -1

Fatiamento de **string**

```
s = "Python"
s[1:4]      → seleciona os elementos das posições 1,2,3
'yth'

s[2:]       → seleciona os elementos a partir da posição 2
'thon'

s[:4]       → seleciona os elementos até a posição 3
'Pyth'
```

Principais operadores para **string**

- Alguns operadores importantes que podem ser usados em **string**:
  - `in`
  - `len`
  - `+`
  - `*`

Operador in

- substring in string
  - ➢ retorna True ou False

```
>>> nome = "Maria Silva"
>>> "M" in nome
True
>>> "B" in nome
False
>>> "m" in nome
False
>>> "ria" in nome
True
```

Função **len()**

- `len(string)`

  - Retorna a quantidade de caracteres da string.

```
>>> nome = "Maria"
>>> len(nome)
5
>>> nome = "Maria Silva"
>>> len(nome)
11
```

Operador de concatenação – “+”

- string1 + string2
  - ➢ Concatena duas strings

```
>>> nome = "Maria" + "Silva"
>>> nome
MariaSilva

>>> nome = "Maria"
>>> sobrenome = "Silva"
>>> nome_completo = nome + sobrenome
>>> nome_completo
MariaSilva
```

Operador de repetição – “*”

- string * int
  - Repete a string int vezes

```
>>> nome = "Maria"
>>> nome_repetido = nome * 2
>>> nome_repetido
MariaMaria
```

Praticando com o Python...

1) Crie um programa que imprima o comprimento de uma string fornecida pelo usuário.

2) Elaborar um programa que solicite nome e sobrenome. Em seguida, concatene ambos em uma nova string, separando com espaço. Ao final, imprima o nome completo fornecido pelo usuário.

3) Construa um programa que apresente o seguinte menu:

****

CÁLCULO DE GRANDEZAS ELÉTRICAS

****

Outros métodos

- Outros métodos úteis para se utilizar com **string**:
  - `upper;`
  - `lower`
  - `split`
  - `partition`

Método **upper()**

- string.upper()
  - Retorna a string com letras minúsculas substituídas por maiúsculas

```
>>> texto = "Quem parte e reparte, fica com a maior
parte"
>>> texto_up = texto.upper()
>>> texto_up
"QUEM PARTE E REPARTE, FICA COM A MAIOR PARTE"
>>> texto
"Quem parte e reparte, fica com a maior parte"
```

Método **lower()**

- string.lower()
  - Retorna a string com letras minúsculas substituídas por maiúsculas

```
>>> texto = "Quem parte e reparte, fica com a maior
parte"
>>> texto.lower()
"quem parte e reparte, fica com a maior parte"
```

Método **split()**

- string.split(separador)
  - Separa a string em "pedaços" que aparecem antes e depois do separador.
  - Se o separador não for especificado, é usado espaços em branco, tabs e quebras de linha como separador.
  - Útil para ler várias entradas de uma única vez

Método **split()**

- string.split(separador) No interpretador (SHELL)

```
>>>dia, mes, ano = input("Digite uma data:
").split("/")
Digite uma data: 10/04/2018
>>>dia
"10"
>>>mes
"04"
>>>ano
"2018"
```

Método **partition()**

- string.partition(separador)
  - Separa a string em três pedaços: o que vem antes da primeira ocorrência do separador, o separador e o que vem depois do separador

Método **partition()**

- string.partition(separador)

```
>>>antes, sep, depois = input("Digite valores:
").partition("-")
Digite valores: 10-20-30-40
>>>antes
"10"
>>>sep
"-"
>>>depois
"20-30-40"
```

Método **partition()**

- string.partition(separador)

```
>>>antes, sep, depois = input("Digite valores:
").partition(" ")
Digite valores: 10 20 30 40
>>>antes
"10"
>>>sep

>>>depois
"20 30 40"
```

Formatação de **string** com **.format()**

Você pode usar **format**() para fazer uma formatação posicional simples.

```
nome = "Guido"
sobrenome = "Van Rossum"
```

```
print('Meu nome é {} e meu sobrenome é {}'.format(nome, sobrenome))
Meu nome é Guido e meu sobrenome é Van Rossum
```

Caso tenhamos valores numéricos:

```
print('Premio total: R$ {0:.2f}, para {1:%d} ganhadores, `
      'onde cada um ficou com R$ {2:.2f}'.format(premio, ganhadores,
premio/ganhadores))
```

Formatação de **string** com **%()**

String em Python têm uma operação integrada exclusiva que pode ser acessada com o operador **%**.

```
nome = "Guido"
sobrenome = "Van Rossum"
```

```
print('Meu nome é {} e meu sobrenome é {}'.format(nome, sobrenome))
Meu nome é Guido e meu sobrenome é Van Rossum
```

Caso tenhamos valores numéricos:

```
print('Premio total: R$ %.2f, para %d ganhadores, '
'onde cada um ficou com R$ %.2f.'%(premio, ganhadores, premio/ganhadores))
```

Formatação de **string** – *f-String*

Quando trabalhamos com **strings**, é bastante comum querermos formatá-las e, para isso, Python oferece diversas maneiras, tais como os marcadores de posição **%**, o método **format**(). No entanto, uma das maneiras mais simples de implementar a formatação de strings é utilizando a Literal *Strings Interpolation* ou, simplesmente, ***f-Strings***. Elas foram incluídas na versão Python 3.6.

A sintaxe das ***f-Strings*** é bastante simples e o seu uso garante a incorporação de expressões dentro do texto literal.

Formatação de **string** – *f-String*

Em um programa Python, as ***f-Strings*** são iniciadas com a letra **f** ou **F**, contendo expressões envolvidas por um par de chaves **{...}**, modificadas dentro da string a ser formatada. As ***f-Strings*** consideram tudo que está fora do par de chaves como sendo um texto literal e, portanto, na saída, o texto será replicado sem nenhuma alteração.

A maneira mais simples de formatar uma **string** é informando o(s) valor(es) que a comporá(ão), conforme exemplo a seguir:

Formatação de **string** – *f-String*

```
premio = float(input('Valor do PREMIO:'))
ganhadores = int(input('Total de GANHADORES:'))

print(f'O PREMIO DE R$ {premio:,.2f}, COM
    {ganhadores} GANHADORES, RECEBERAM R$
    {premio/ganhadores:,.2f}')
```

Formatação de **string**

```
premio = float(input('VALOR DO PREMIO:'))
ganhadores = int(input('TOTAL DE GANHADORES:'))

print('Cada um vai ficar com R$', premio/ganhadores)
print('Cada um vai ficar com R$' + str(premio/ganhadores))
print('Cada um vai ficar com R$ %.2f'%(premio/ganhadores))
print('Cada um vai ficar com R$ {0:.2f}'.format(premio/ganhadores))


print(f'O PREMIO DE R$ {premio:,.2f}, COM {ganhadores} GANHADORES,
RECEBERAM R$ {premio/ganhadores:,.2f}')
```

Outros métodos para **string**

Método	Descrição	Exemplo
len()	Retorna o tamanho da string.	teste = “Apostila de Python” len(teste) 18
capitalize()	Retorna a string com a primeira letra maiúscula	a = "python" a.capitalize() 'Python'
count()	Informa quantas vezes um caractere (ou uma sequência de caracteres) aparece na string.	b = "Linguagem Python" b.count("n") 2
startswith()	Verifica se uma string inicia com uma determinada sequência.	c = "Python" c.startswith("Py") True
endswith()	Verifica se uma string termina com uma determinada sequência.	d = "Python" d.endswith("Py") False

Outros métodos para **string**

Método	Descrição	Exemplo
isalnum()	Verifica se a string possui algum conteúdo alfanumérico (letra ou número).	e = "!@#$%" e.isalnum() False
isalpha()	Verifica se a string possui apenas conteúdo alfabético.	f = "Python" f.isalpha() True
islower()	Verifica se todas as letras de uma string são minúsculas.	g = "pytHon" g.islower() False
isupper()	Verifica se todas as letras de uma string são maiúsculas.	h = "# PYTHON 12" h.isupper() True
lower()	Retorna uma cópia da string trocando todas as letras para minúsculo.	i = "#PYTHON 3" i.lower() '#python 3'

Outros métodos para **string**

Método	Descrição	Exemplo
upper()	Retorna uma cópia da string trocando todas as letras para maiúsculo.	j = "Python" j.upper() 'PYTHON'
swapcase()	Inverte o conteúdo da string (Minúsculo / Maiúsculo).	k = "Python" k.swapcase() 'pYTHON'
title()	Converte para maiúsculo todas as primeiras letras de cada palavra da string.	l = "apostila de python" l.title() 'Apostila De Python'
split()	Transforma a string em uma lista, utilizando os espaços como referência.	m = "cana de açúcar" m.split() ['cana', 'de', 'açúcar']

Outros métodos para **string**

Método	Descrição	Exemplo
replace(S1, S2)	Substitui na string o trecho S1 pelo trecho S2.	n = "Apostila teste" n.replace("teste", "Python") 'Apostila Python'
find()	Retorna o índice da primeira ocorrência de um determinado caractere na string. Se o caractere não estiver na string retorna -1.	o = "Python" o.find("h") 3
ljust()	Ajusta a string para um tamanho mínimo, acrescentando espaços à direita se necessário.	p = " Python" p.ljust(15) ' Python '
rjust()	Ajusta a string para um tamanho mínimo, acrescentando espaços à esquerda se necessário.	q = "Python" q.rjust(15) ' Python'
center()	Ajusta a string para um tamanho mínimo, acrescentando espaços à esquerda e à direita, se necessário.	r = "Python" r.center(10) ' Python '
lstrip()	Remove todos os espaços em branco do lado esquerdo da string.	s = " Python " s.lstrip() 'Python '
rstrip()	Remove todos os espaços em branco do lado direito da string.	t = " Python " t.rstrip() ' Python'
strip()	Remove todos os espaços em branco da string.	u = " Python " u.strip() 'Python'

Praticando com o Python...

1) Crie um programa que imprima o comprimento de uma string fornecida pelo usuário.

2) Escreva um programa que leia uma frase e converta em uma outra variável a cadeia de caracteres de letras maiúsculas em letras minúsculas.

3) Elabore um programa que solicite uma frase ao usuário e escreva a frase toda em maiúscula. No mesmo programa exiba a frase sem espaços em branco. Dica use **replace.**

4) Desenvolva um programa que solicite uma frase ao usuário e escreva a frase invertida. Dica **[::-1]**

Praticando com o Python...

5) Crie um programa que leia o nome completo de uma pessoa todo em minúsculo e exiba este nome com as primeiras letras em maiúsculo. Dica: use **.title()**

6) Seguindo o exercício acima, exiba da mesma forma, no entanto a entrada será com todos os caracteres maiúsculos. Dica: use **.lower()** e .**title**()

7) Elabore um programa que leia o nome do usuário e mostre o nome de traz para frente, utilizando somente letras maiúsculas.

8) Desenvolva um programa que leia uma frase e um caractere. Em seguida, exiba ambos e o número de ocorrências do caractere na frase.

Praticando com o Python...

9) Elabore um programa que leia uma frase, uma palavra antiga e uma palavra nova. O programa deve exibir uma string contendo a frase original e outra com a ocorrência da palavra antiga substituída pela palavra nova.

Exemplo:

Frase: “Quem parte e reparte fica com a maior parte”
Palavra antiga: “parte”
Palavra nova: “parcela”
Resultado a ser impresso no programa : “Quem parcela e reparcela fica com a maior parcela”

Dica: use replace


Obrigado!