Computação I - Python
Aula 4 - Teórica: Variáveis e Atribuição, Strings

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ

Variáveis e Atribuição

Dados a hora, minuto e segundo em que um corredor de uma maratona partiu, e dados a hora, minuto e segundos em que este mesmo corredor cruzou a linha de chegada, faça a função *tempoMaratona* que calcula o tempo total de prova deste corredor em horas, minutos e segundos.

```
1 def tempoMaratona(hi,mi,si,hf,mf,sf):
2   """Funcao que calcula o tempo de uma atleta em uma prova de maratona
3   Parametros de Entrada: int,int,int,int,int,int
4   Valor de Retorno : int,int,int"""
5
6   return ((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))//3600,
7          (((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))%3600)//60,
8          (((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))%3600)%60
```

Variáveis e Atribuição

Dados a hora, minuto e segundo em que um corredor de uma maratona partiu, e dados a hora, minuto e segundos em que este mesmo corredor cruzou a linha de chegada, faça a função *tempoMaratona* que calcula o tempo total de prova deste corredor em horas, minutos e segundos.

```
1 def tempoMaratona(hi,mi,si,hf,mf,sf):
2   """Funcao que calcula o tempo de uma atleta em uma prova de maratona
3   Parametros de Entrada: int,int,int,int,int,int
4   Valor de Retorno : int,int,int"""
5
6   return ((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))//3600,
7          (((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))%3600)//60,
8          (((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))%3600)%60
```

Note que a mesma conta é feita várias vezes !

Variáveis e Atribuição

```
1  def tempoMaratona(hi,mi,si,hf,mf,sf):
2    """Funcao que calcula o tempo de uma atleta em uma prova de maratona
3    Parametros de Entrada: int,int,int,int,int,int
4    Valor de Retorno : int,int,int"""
5
6    return ((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))//3600,
7           (((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))%3600)//60,
8           (((sf + mf * 60 + hf * 3600) - (si + mi * 60 + hi * 3600))%3600)%60
```

```
1  def tempoMaratona(hi,mi,si,hf,mf,sf)
2    """Funcao que calcula o tempo de uma atleta em uma prova de maratona
3    Parametros de Entrada: int,int,int,int,int,int
4    Valor de Retorno : int,int,int"""
5
6    # converte horas e minutos para segundos
7    segundos_inicial = si + mi * 60 + hi * 3600
8    segundos_final = sf + mf * 60 + hf * 3600
9
10   # calcula a quantidade de segundos que o corredor levou para terminar a prova
11   diferenca = segundos_final - segundos_inicial
12
13   # converte a quantidade de segundos de volta para horas, minutos e segundos
14   horas = diferenca//3600
15   resto = diferenca%3600
16   minutos = resto // 60
17   segundos = resto % 60
18   return horas, minutos, segundos
```

Variáveis e Atribuição

- **Variável**: É uma maneira simbólica de fazer referência a dados armazenados na memória do computador.
- Toda variável engloba os seguintes aspectos, semelhantes aos parâmetros de uma função:
  - **Nome (identificador)**: é a representação simbólica da variável, que será utilizada pelo programador para fazer referência aos dados que ela armazena.

```
1 >>> x = 3
2 >>> x
3     3
```

  - **Valor**: o que de fato está armazenado.
  - **Tipo**: o tipo de dado que está armazenado.

Variáveis – Nomes de Variáveis

- Letras, números e underline (não começar por números)
  - minhaVariavel = 1
  - minha_variavel = 2
  - minhaVariavel2 = 3
  - minha_variavel_2 = 4
- **Dica**: em funções muito grandes e complexas, escolha (se possível) nomes que descrevam o significado da variável. Exceto em funções muito simples ou exemplos didáticos, evite nomes genéricos como “x”, “y”, “a”, etc.

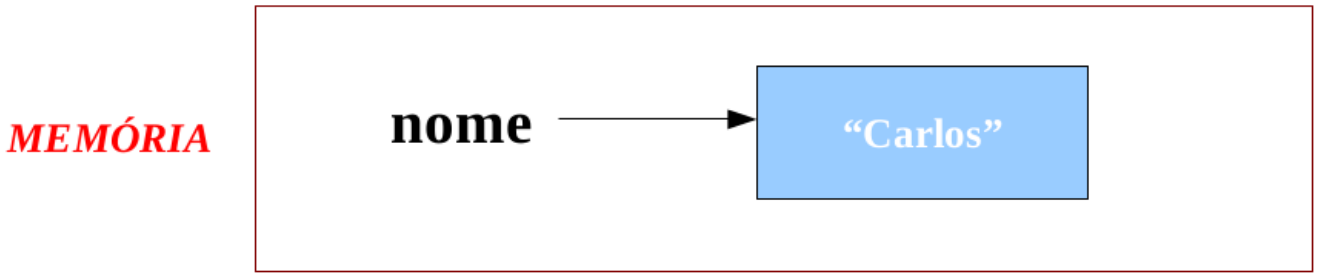
Variáveis e Atribuição

- **Atribuição**: O símbolo = é usado para atribuir um valor a uma variável.

$$var = valor$$

$$var1, var2, ..., varN = valor1, valor2, ..., valorN$$

```
...
nome = "Carlos"
return "Olá "+ nome
```



Atribuindo Valores a Variáveis

```
>>> a = 1 # atribuo o valor 1 a variavel a
>>> a     # da o valor armazenado em a
    1

>>> a,b,c = 1,2,3 # atribuicao multipla - variaveis a, b e c
>>> a
    1
>>> b
    2
>>> c
    3
>>> a = 2 + 5
>>> a
    7
>>> a = a + 4 # estamos fazendo uma auto-atribuicao a a
>>> a
    11
>>> a = 10 * d # d nao foi definido
Traceback (most recent call last)
  File ""<stdin>", line 1, in <module>
NameError: name "d" is not defined
```

O lado direito da atribuição é sempre avaliado antes que a atribuição seja feita (para que o valor seja calculado e depois armazenado na variável do lado esquerdo).

Atribuindo Valores a Variáveis

Uma variável é criada com um comando de atribuição: *variavel* = *valor*

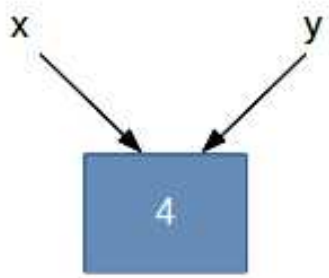
```
1 >>> x = 4
```

Um alias é um identificador que se refere a uma variável existente. É criado com uma atribuição *variavel* = *outra_variavel* já existente

```
1 >>> y = x
```

A variável *y* é um alias para a variável *x*. Portanto, *y* possui o mesmo valor e aponta para a mesma posição na memória que *x*.

```
1 >>> y
2     4
```



Atribuindo Valores a Variáveis

Para os tipos de dados imutáveis (inteiros, strings, booleanos e float), os alias são desfeitos quando uma nova atribuição é feita a qualquer uma das duas variáveis envolvidas no alias:

```
>>> x = 4
>>> y = x
>>> x = 5
>>> y
    4
>> estudante = True
>> meia_entrada = estudante
>> meia_entrada
   True
>> estudante = False
>> meia_entrada
   True
>> estudante
   False
```

Atribuindo Valores a Variáveis

ATENÇÃO

Qual a diferença entre as funções abaixo ?

```
1 def teste1():
2   a = 10
3   a, b = 3 * a, a
4   return a, b
```

```
1 def teste2():
2   a = 10
3   a = 3 * a
4   b = a
5   return a, b
```

Atribuindo Valores a Variáveis

ATENÇÃO

Qual a diferença entre as funções abaixo ?

```
1 def teste1():
2   a = 10
3   a, b = 3 * a, a
4   return a, b
```

```
1 def teste2():
2   a = 10
3   a = 3 * a
4   b = a
5   return a, b
```

O lado direito da atribuição é sempre avaliado antes que a atribuição seja feita.

Variáveis e Atribuição

Variáveis são usadas para guardar dados intermedidários nas funções.

Faça uma função que dado um número inteiro, retorna seu valor absoluto.

```
1 def absoluto(X):
2   """Funcao que retorna o valor absoluto de um numero
3   Parametros de Entrada: int
4   Valor de Retorno : int"""
5
6   if X < 0:
7     return -1*X
8   else:
9     return X
```

Variáveis e Atribuição

Variáveis são usadas para guardar dados intermedidários nas funções.

Faça uma função que dado um número inteiro, retorna seu valor absoluto.

```
1 def absoluto(X):
2   """Funcao que retorna o valor absoluto de um numero
3   Parametros de Entrada: int
4   Valor de Retorno : int"""
5
6   if X < 0:
7     return -1*X
8   else:
9     return X
```

```
1 def absoluto(X):
2   """Funcao que retorna o valor absoluto de um numero
3   Parametros de Entrada: int
4   Valor de Retorno : int"""
5
6   if X < 0:
7     X = -1*X # alterando o valor de X
8   return X
```

Variáveis e Atribuição

```
1  import datetime
2  def idade(dia,mes,ano):
3    """Funcao que calcula a idade de uma pessoa.
4    Parametros de Entrada: int, int, int
5    Valor de Retorno : str"""
6
7    if dia == datetime.datetime.now().day and mes == datetime.datetime.now().month:
8
9      return str(datetime.datetime.now().year - ano) + " anos. Parabens pelo aniversario"
10
11   elif ((mes < datetime.datetime.now().month) or
12 (mes == datetime.datetime.now().month and dia < datetime.datetime.now().day)):
13
14     return str(datetime.datetime.now().year - ano) + " anos."
15
16   else:
17
18     return str(datetime.datetime.now().year - ano - 1) + " anos."
```

Como podemos simplificar a função acima usando variáveis ?

Variáveis e Atribuição

Como podemos simplificar a função acima usando variáveis ?

```
1  import datetime
2  def idade(dia,mes,ano):
3    """Funcao que calcula a idade de uma pessoa.
4    Parametros de Entrada: int, int, int
5    Valor de Retorno : str"""
6
7    diaHoje = datetime.datetime.now().day # variavel para guardar o dia corrente
8    mesHoje = datetime.datetime.now().month # variavel para guardar o mes corrente
9    anoHoje = datetime.datetime.now().year # variavel para guardar o ano corrente
10
11   if dia == diaHoje and mes == mesHoje:
12     return str(anoHoje - ano) + " anos. Parabens pelo aniversario"
13   elif ((mes < mesHoje) or (mes == mesHoje and dia < diaHoje)):
14     return str(anoHoje - ano) + " anos."
15   else:
16     return str(anoHoje - ano - 1) + " anos."
```

Variáveis – Tipo

- Python é uma linguagem dinamicamente tipada ou fracamente tipada.
- O tipo é atribuído de acordo com o valor atribuído à variável. Não é necessário *declarar previamente* o tipo.

```
>>> x = 4
>>> type(x)
    <class 'int'>
```

- O tipo de uma variável pode mudar depois de alguma operação ou nova atribuição.

```
>>> x = complex(x)
>>> type(x)
    <class 'complex'>
```

Variáveis – Escopo

- **Escopo**: onde a variável existe e onde ela deixa de existir.
- As variáveis definidas dentro de uma função são ditas **variáveis locais**, porque não podem ser acessadas fora da função.

```
1 def produtoSomaDiferenca(a,b):
2   x = a + b
3   y = a - b
4   return x*y
```

- As variáveis x e y são locais, pois só existem dentro da função. Depois que a função é executada, elas são destruídas.
- Dizemos que a função é o escopo de x e y.
- Tentar chamá-las fora da função ocasionaria um erro.

Variáveis – Escopo

```
1 def produtoSomaDiferenca(a,b):
2   x = a + b
3   y = a - b
4   return x*y
```

```
1 def quociente(a,b):
2   return x/y
```

Ao executar esta função no shell:

```
1 >>> quociente(10,12)
2 Traceback (most recent call last):
3   File "<pyshell#0>", line 1, in <module>
4     quociente(10,12)
5   File "C:/Users/Desktop/teste.py", line 7, in quociente
6     return x/y
7 NameError: global name 'x' is not defined
```

Exercícios

1. Faça uma função que receba dois parâmetros, sendo eles a hora e o minuto corrente, e informe quanto tempo (em minutos) se passou desde o início do dia. Faça o chinês para os seguintes valores:

Entrada	**Valor de Retorno**
(3,25)	?
(?,?)	882

2. Faça uma função que receba o valor inicial e a razão e retorne os 4 primeiros elementos da progressão geométrica correspondente. Experimente resolver o problema com diferentes números de variáveis. Qual o mínimo?

Strings

- **Caracteres** são símbolos. Podem ser letras, números, caracteres especiais, e até o espaço em branco é um caractere.
  Exemplo: 'a', '9', '#', ' '.

- Uma *string* é uma sequência de caracteres.

```
>>> a = 'abcd'
>>> b = "1234"
>>> c = "#$5a"
>>> d = ''
>>> e = '  '
```

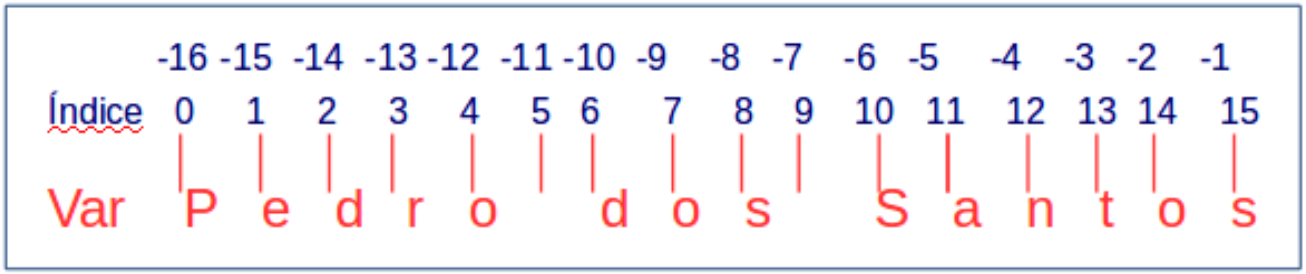
- Comprimento de uma string: número de caracteres que ela contém.

```
>>> s = '123456'
>>> len(s)
    6
```

Strings - Índices

- Todo caractere de uma string é **indexado**, começando do primeiro caractere (**índice 0**) à esquerda.
- **Notação**: string[indice]

Exemplo: var = "Pedro dos Santos"

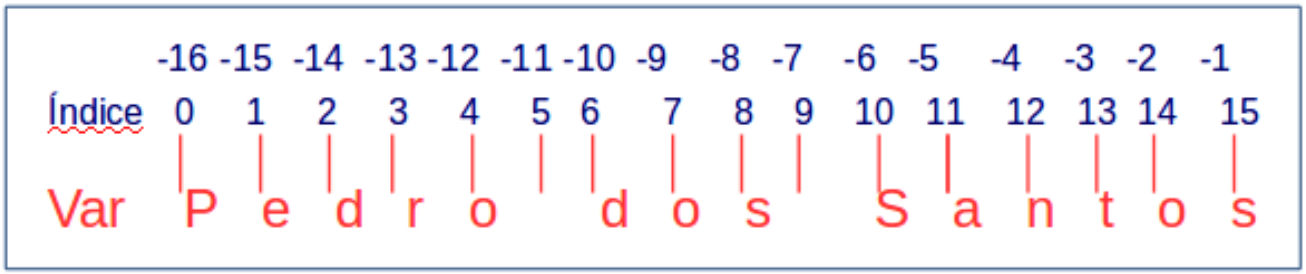



```
1 >>> var[2]
2     'd'
3 >>> var[9]
4     ' '
5 >>> var[15]
6     's'
```

Strings - Índices

- A string também pode ser indexada da direita para a esquerda, começando no **índice -1**.
- **Notação**: string[indice]

Exemplo: var = "Pedro dos Santos"




```
1 >>> var[-14]
2     'd'
3 >>> var[-7]
4     ' '
5 >>> var[-1]
6     's'
```

Strings - Fatiamento

- Separa trechos de uma string.
- **Notação**: string[índice1:índice2]
  - Retorna os caracteres desde o de índice1 até o imediatamente anterior ao índice2
  - Se índice1 é omitido, é assumido 0.
  - Se índice2 é omitido, é assumido o fim da string.

Strings - Fatiamento

Exemplo

```
>>> x = 'abcde'
>>> x[0:2]

>>> x  [2:]

>>> x[:]

>>> x[-1:]

>>> x[:-1]
```

Strings - Fatiamento

Exemplo

```
>>> x = 'abcde'
>>> x[0:2]
    'ab'
>>> x [2:]
    'cde'
>>> x[:]
    'abcde'
>>> x[-1:]
    'e'
>>> x[:-1]
    'abcd'
```

Strings - Fatiamento

Incremento: podemos usar incremento / decremento para selecionar os elementos de uma string.

[start:end:step]: vai do índice *start* até *end* (sem ultrapassá-lo, com passo *step*)

Exemplo

```
1 >>> x= "abcde"
2 >>> x[0:-1:2]
3
4 >>> x[3:0:-1]
```

Strings - Fatiamento

Incremento: podemos usar incremento / decremento para selecionar os elementos de uma string.

[start:end:step]: vai do índice *start* até *end* (sem ultrapassá-lo, com passo *step*)

Exemplo

```
>>> x= "abcde"
>>> x[0:-1:2]
    'ac'
>>> x[3:0:-1]
    'dcb'
```

Strings

- Elementos de uma string não aceitam o operador de atribuição.

```
>>> s = '123456'
>>> s[0] = '0'
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    s[0]='0'
TypeError: 'str' object does not support item assignment
```

- Strings são, portanto, imutáveis. Ou seja, os dados contidos em uma string não podem ser alterados.

Strings - Recapitulando

- **Representação**: s = “12346” ou s = ‘123456’
- **len(s)** : retorna o tamanho de uma string.
- **Operador +**: concatena strings. Ex: ‘ab’ + ‘cd’ = ‘abcd’
- **Operador ***: repete strings. Ex: ‘a’*5 = ‘aaaaa’
- **Fatias (Slices)**: [start:end:step]

Exercício

1. Faça uma função que dado o nome de uma pessoa, retorne o número de letras do nome e a primeira letra do nome.

2. Faça uma função que dada uma palavra, retorna a palavra invertida.

3. Faça uma função que dada uma palavra, retorna os caracteres nas posições ímpares.

4. Faça uma função que recebe duas strings e retorna a concatenação delas, com exceção do primeiro caractere de cada uma. Exemplo: dadas as entradas 'abcd' e 'efghi', o valor de retorno será 'bcdfghi'.

5. Escreva uma função que receba uma string e retorne a concatenação de três cópias dos dois últimos caracteres. Exemplo, se a entrada for 'abcd', a saída deve ser 'cdcdcd'.

6. Faça uma função que recebe duas strings e retorna a concatenação delas,em ordem alfabética, com espaço no meio. Exemplo: dadas as entradas 'xbcd' e 'efghi', o valor de retorno será 'efghi xbcd'.

Autores

- João C. P. da Silva ▸ Lattes
- Carla Delgado ▸ Lattes
- Ana Luisa Duboc ▸ Lattes

Colaboradores

- Anamaria Martins Moreira ▸ Lattes
- Fabio Mascarenhas ▸ Lattes
- Leonardo de Oliveira Carvalho ▸ Lattes
- Charles Figueiredo de Barros ▸ Lattes
- Fabrício Firmino de Faria ▸ Lattes

Computação I - Python
Aula 4 - Teórica: Variáveis e Atribuição, Strings

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ