Laboratório de Programação 1

Algoritmos e a Lógica de Programação

Mário Hozano
professor@hozano.com

Ciência da Computação
UFAL - Arapiraca

Relembrando a aula anterior...

- O que é um programa?
- O que é uma Linguagem de Programação?
- O que é Linguagem de Máquina?
- Cite linguagens de baixo e alto níveis de abstração?
- O que é compilador e interpretador?
- O que é Python?
- Como utilizar o interpretador Python?

Roteiro da aula

- Recomendações Iniciais
- Erros de Sintaxe, *Runtime Errors* e Erros de Semântica
- Depuração de Código
- Valores, Tipos e Variáveis
- Entrada Manual
- Operações Matemáticas e com Strings
- Comentários

Recomendações

- Antes de sair programando é importante saber algumas coisas

- Muuuuuitas vezes seus programas vão apresentar erros, também chamados de *bugs*

- Estes erros podem ser de 3 tipos:
  1. Erros de sintaxe
  2. Erros em tempo de execução (*runtime errors)*
  3. Erros de semântica

Erros de Sintaxe

- O interpretador Python só pode executar um programa se sua sintaxe estiver correta
- Sintaxe se refere à estrutura de um programa e às regras sobre esta estrutura.
- Ex.: Um parêntese aberto deve ser fechado no mesmo programa

```
>>> (1 + 2)
3
>>> 1 + 2)
  File "<stdin>", line 1
    1 + 2)
         ^
SyntaxError: invalid syntax
```

Erros de Sintaxe

- Outro erro comum: espaços excessivos/inadequados no início das instruções (**erro de identação**)

```
>>> 1 + 2
3
>>>   1 + 2
  File "<stdin>", line 1
    1 + 2
    ^
IndentationError: unexpected indent
>>>
```

- Ao encontrar erros de sintaxe, o interpretador exibe uma mensagem de erro e não inicia o programa
- No início do aprendizado erros de sintaxe são comuns

Erros em Tempo de Execução (*Runtime Errors*)

- Erros de execução são percebidos pelo interpretador apenas quando o programa está sendo executado
- Estes erros são conhecidos como **Exceções,** raras em programas muito simples
- Ao encontrar um erro de *runtime*, o sistema pára e apresenta uma mensagem com o erro

```
>>> a = 1
>>> b = "2"
>>> a + b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>>
```

Erros de Semântica

- Também conhecido como **erro de lógica**
- Não param a execução do programa nem avisa que existe erro
- Entretanto, o programa não vai fazer a coisa certa porque foi escrito de forma errada

```
>>> a = "1"
>>> b = "2"
>>> a + b
'12'
>>>
```

Depuração de Código

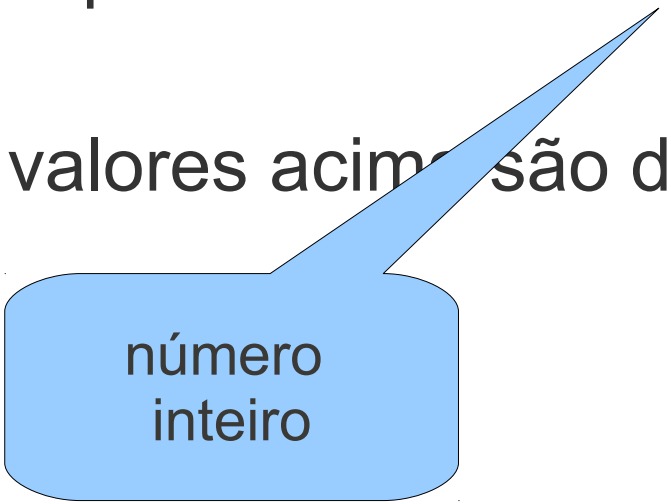
- Muitas vezes você fará programas que terão erros
- Encontrar estes erros pode ser uma tarefa difícil
- O processo de investigação e correção de erros em um programa é conhecido como **depuração** (*debugging*)
- “Debugar” é uma das mais importantes habilidades de um grande programador
- A depuração de código é uma área fértil em pesquisas na Ciência da Computação

Valores e Tipos

- Valores são registros fundamentais que são manipulados pelo programa
- Valores podem ser coisas básicas como letras, números, datas
- Exemplos de valores: 13, “Maria”, 3.14
- Os valores acima são de diferentes tipos

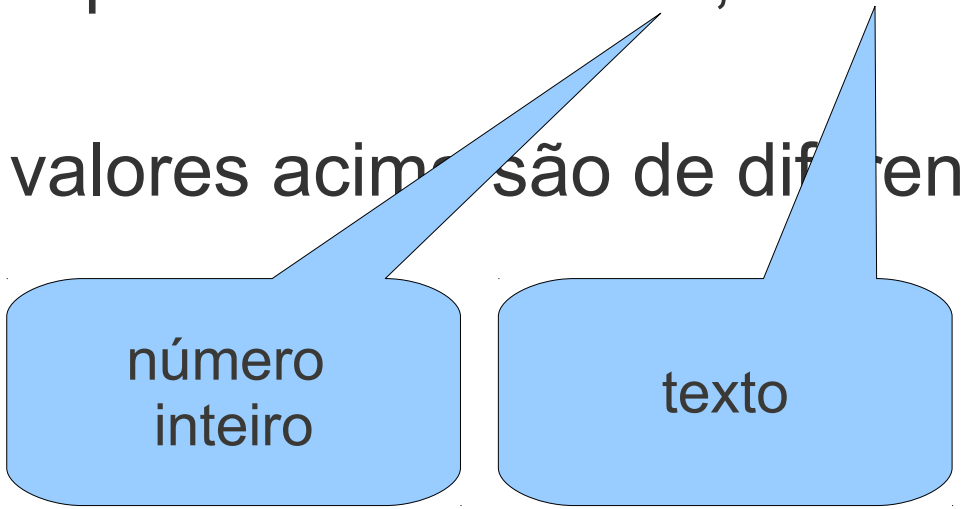
Valores e Tipos

- Valores são registros fundamentais que são manipulados pelo programa
- Valores podem ser coisas básicas como letras, números, datas
- Exemplos de valores: 13, “Maria”, 3.14
- Os valores acima são de diferentes tipos



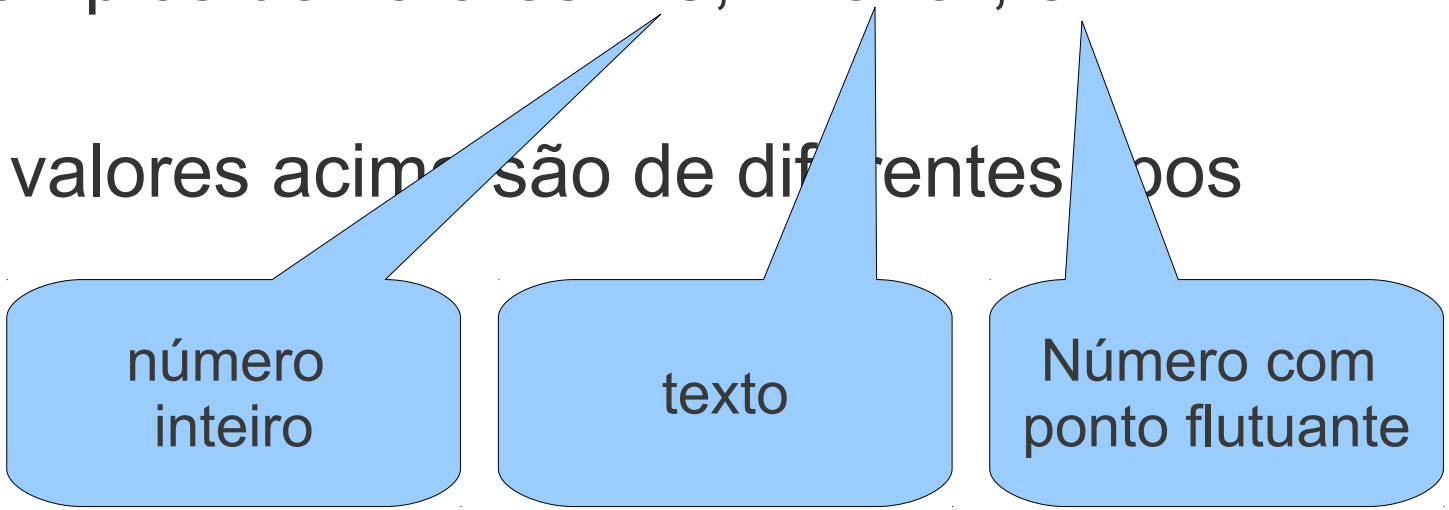
Valores e Tipos

- Valores são registros fundamentais que são manipulados pelo programa
- Valores podem ser coisas básicas como letras, números, datas
- Exemplos de valores: 13, “Maria”, 3.14
- Os valores acima são de diferentes tipos



Valores e Tipos

- Valores são registros fundamentais que são manipulados pelo programa
- Valores podem ser coisas básicas como letras, números, datas
- Exemplos de valores: 13, “Maria”, 3.14
- Os valores acima são de diferentes tipos



Valores e Tipos

- Em Python, números e texto são definidos com os tipos a seguir:
  - *int* = Número inteiro
  - *string* (ou *str*) = Texto
  - *float* = Número com ponto flutuante

- Para o interpretador python, qualquer valor que esteja entre aspas é considerado do tipo *string*

```
>>> type("Maria")
<type 'str'>
>>> type("   ")
<type 'str'>
>>> type("325")
<type 'str'>
```

Use a função *type()* para identificar o tipo dos valores

Valores e Tipos

- Números são apresentados sem aspas (Ex.: *19*)
- Números com ponto flutuante utilizam um ponto (.) como separador das casas decimais (Ex.: *3.14*)

```
>>> type(325)
<type 'int'>
>>> type(3.14)
<type 'float'>
>>> type(3)
<type 'int'>
>>> type(3.0)
<type 'float'>
```

Variáveis

- Uma **variável** é um nome que se refere a um valor
- Em python uma variável é criada (ou atualizada) na hora em que recebe um valor
- Não existe variável sem valor
- Os comandos abaixo representam a criação de 3 variáveis com valores distintos

```
>>> a = 17
>>> b = “Ana Maria”
>>> pi = 3.1415926535897931
```

Variáveis

- Para exibir o valor de uma variável você pode utilizar a função *print()*

```
>>> nome = “Ana Maria”
>>> print(nome)
Ana Maria
```

- Para exibir o tipo de uma variável você pode usar também a função *type()*

```
>>> nome = “Ana Maria”
>>> type(nome)
<type 'str'>
```

Nomes de Variáveis

- Variáveis devem possuir nomes que indiquem o que elas representam

- Os nomes das variáveis devem ser, na maioria das vezes, auto-descritivos

```
>>> a = “Ana Maria da Silva”
>>> nome = “Ana Maria da Silva”
>>> nome_completo = “Ana Maria da Silva”
>>> telefone = “8888-9999”
```

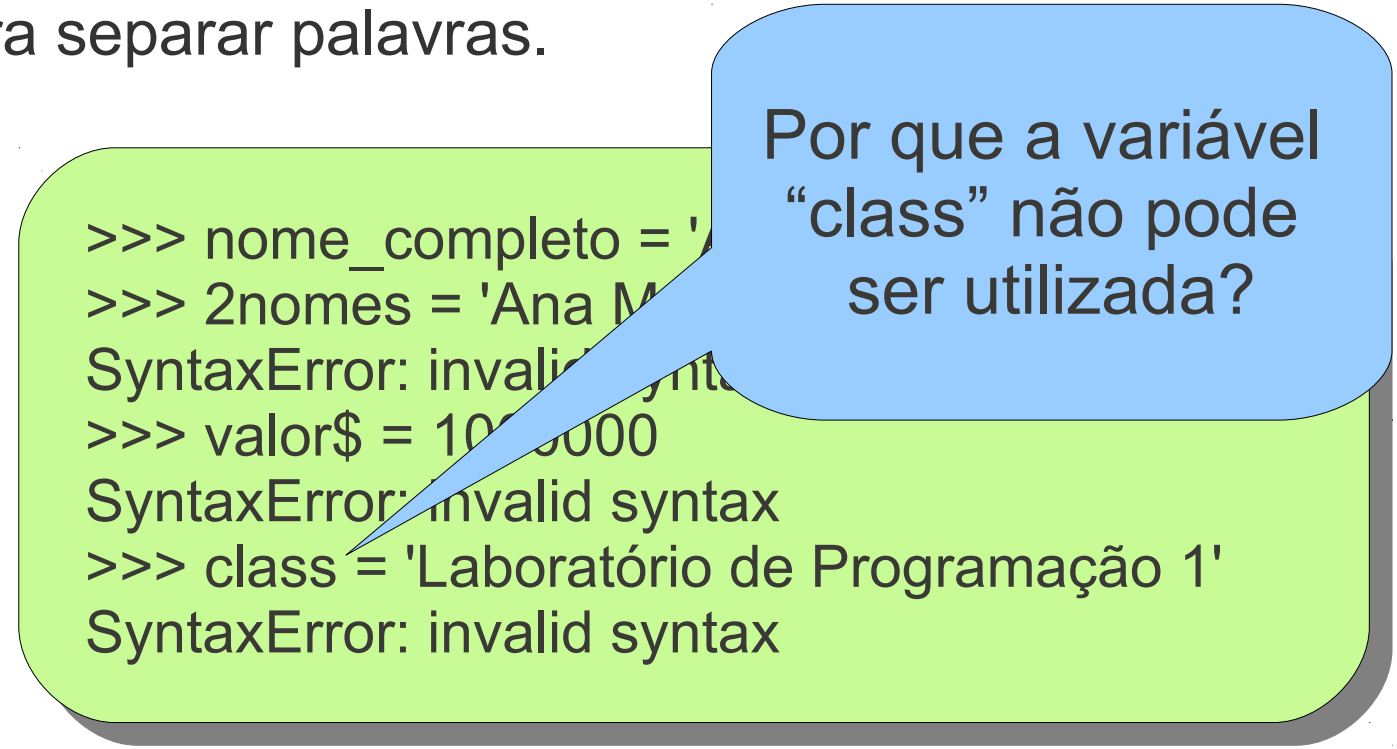
Nomes de Variáveis

- Nomes de variáveis ...
  - podem ser longos;
  - podem conter letras e números, mas devem, obrigatoriamente, começar com uma letra;
  - podem conter o caractere (_). Normalmente ele é utilizado para separar palavras.

```
>>> nome_completo = 'Ana Maria da Silva'
>>> 2nomes = 'Ana Maria'
SyntaxError: invalid syntax
>>> valor$ = 1000000
SyntaxError: invalid syntax
>>> class = 'Laboratório de Programação 1'
SyntaxError: invalid syntax
```

Nomes de Variáveis

- Nomes de variáveis ...
  - podem ser longos;
  - podem conter letras e números, mas devem, obrigatoriamente, começar com uma letra;
  - podem conter o caractere (_). Normalmente ele é utilizado para separar palavras.



Nomes de Variáveis

- Nomes de variáveis não podem ser iguais às palavras reservadas de Python
- Python possui 32 palavras reservadas

and	del	for	lambda	return
as	elif	from	None	try
assert	else	global	not	while
break	except	if	or	with
class	exec	import	pass	yield
continue	exec	in	print	
def	finally	is	raise	

Entrada Manual

- Muitas vezes, os programas precisam que o usuário informe valores
- Isto pode ser feito através da função *input().* Esta fução pára a execução do programa até que o usuário use o teclado para digitar o valor e pressionar ENTER.

```
>>> num1 = input()
```

- A função *input* permite que o programador indique um texto explicativo para o usuário entrar com o valor desejado

```
>>> num1 = input(“Digite um número inteiro:”)
```

Entrada Manual – Usando a função *input()*

- Com a função *input()*, o usuário indica valores a partir do primeiro caractere após a mensagem explicativa

- Para melhorar a apresentação do programa, costuma-se deixar um espaço no final da mensagem explicativa

```
>>> num1 = input(“Digite um número inteiro: ”)
```

- … ou então, pode-se utilizar o marcador (\n) representando uma quebra de linha. Assim, o usuário indicará valores apenas na linha seguinte.

```
>>> num1 = input(“Digite um número inteiro: \n”)
```

Operadores Matemáticos

- Em python, o desenvolvedor pode realizar operações matemáticas utilizando os seguintes operadores:

Operador	Operação
+	Adição
-	Subtração
/	Divisão
*	Multiplicação
**	Exponenciação

```
>>> 10 + 5
15
>>> 6 * 9
54
>>> 3**3
27
```

Operadores Matemáticos

- O resulado de uma operação matemática pode ser diferente a depender dos seus operandos
- Caso os osperandos sejam do tipo *int* a operação resultará em um número inteiro
- Caso um dos operandos seja do tipo *float* a operação resultará em um número de ponto flutuante

```
>>> type(3 + 2)
<type 'int'>
>>> type(3 - 2.0)
<type 'float'>
>>> type(3.0 * 2)
<type 'float'>
```

Operadores Matemáticos - Divisão

- O resultado de uma divisão pode ser diferente dependendo dos seus operandos
- Caso ambos os osperandos sejam do tipo *int* será realizada uma divisão inteira
- Se, pelo menos, um dos operandos seja do tipo float a operação realizará uma divisão comum

```
>>> 3/2
1
>>> 3/2.0
1.5
>>> 3.0/2
1.5
```

Operadores Matemáticos – Ordem dos Operadores

- Quando mais de um operador é utilizado em um comando, o cálculo da expressão é realizado seguindo uma ordem de precedência dos operadores

- Python segue a mesma regra de precedência da matemática, com precedência para:
  1. Parênteses
  2. Exponenciação
  3. Multiplicação e Divisão
  4. Adição e Subtração

```
>>> 3 + 7 * 5**(4 - 2) / 2
90
```

Operações com Strings

- Em geral não se utiliza operadores matemáticos com operandos do tipo *string*
- Entretanto, Python permite utilizar os operadores (+) e (*) com *strings*

```
>>> "lab" + "prog"
'labprog'
>>> "lab" * 3
'lablablab'
```

Operações com Strings

- Em geral não se utiliza operadores matemáticos com operandos do tipo *string*
- Entretanto, Python permite utilizar os operadores (+) e (*) com *strings*

```
>>> "lab" + "prog"
'labprog'
>>> "lab" * 3
'lablablab'
```

Em computação, o processo de junção de duas *strings* é chamado de **concatenação**.

Comentários

- À medida que seu programa cresce se torna mais difícil compreendê-lo
- Em programas com centenas ou milhares de linhas de código é praticamente impossível entender o que faz cada trecho de código
- Para facilitar, as linguagens de programação possui facilidades para a escrita de textos explicativos nos códigos, conhecidos como **comentários**
- Os comentários normalmente apresentam textos em linguagem natural que ajudam os desenvolvedores a entenderem do que se trata uma parte do código

Comentários

- Em Python, comentários são utilizados utilizando o caractere (#) antes do texto explicativo
- Os comentários são ignorados pelo interpretador

```
# converte uma temperatura de celsius para farenheit
temp_celsius = (temp_f - 32) * ( 5 / 9 )

velocidade = 100      # velocidade em m/s
```

- A utilização de nomes adequados para variáveis e o uso de instruções bem definidas podem minimizar a necessidade de comentários

Exercícios

- Crie programas em Python para:

1. Realizar soma, subtração, multiplicação e divisão a partir de dois números indicados pelo usuário.

2. Calcular o Índice de Massa Corpórea de uma pessoa a partir de peso e altura dados.

3. Calcular a área de um triângulo a partir da indicação de base e altura.

4. Calcular a área e circunferência de um círculo a partir da indicação do raio.