Programação para Juristas

PROF. MATHEUS SILVA

2020-09-21

Utilizando o Python

- Tanto no Mac quanto no Windows: recomendável a utilização do editor de texto Atom. Sempre guardar com extensão .py.
  - Mac: terminal, ir para a pasta em que está o ficheiro .py, *python3 nome_ficheiro.py*
  - Windows: linha de comando, ir para a pasta em que está o ficheiro .py, *py nome_ficheiro.py*
- PythonAnywhere.com: fazer login em sua conta, no dashboard carregar em "browse files", em seguida digitar um novo nome com extensão .py e depois "new file".

Realizando *debug* em um programa

- Debug = *depuração*. Processo de encontrar a causa de um erro em um código.
- O processo de depuração pode envolver os seguintes passos:
  - Examinar o código por meio da leitura do mesmo, tentando descobrir se ele está fazendo o que você deseja;
  - Fazer pequenas mudanças no código, incluindo a função print() em partes distintas para que você veja o resultado do que o programa faz até ali;

Realizando *debug* em um programa

- Analisar o tipo de erro apresentado, buscando descobrir o que estas mensagens significam e vinculando as mensagens com alguma eventual causa;

- Desfazer as mudanças recentes, dando um passo atrás para poder depois dar dois passos à frente.

Possíveis tipos de erros no Python

- Erros de **sintaxe**: erros de “gramática”;
- Erros de **lógica**: quando a ordem das frases está errada;
- Erros de **semântica**: quando não há erros de sintaxe nem de lógica, mas mesmo assim o programa não faz aquilo que dele se espera.
- Geralmente o Python irá indicar a linha em que o erro ocorre e qual é o erro, mas isto nem sempre irá ocorrer exatamente como se espera.

Realizando *debug* em um programa

```
n = 5
while n > 0:
    print(n)
    n = n - 1
primt(“Fim”)
```

```
n = 5
while n > 0:
    n = n - 1
    print(n)
print(“Fim”)
```

```
n = 5
while n > 0:
    print(n)
n = n - 1
print(“Fim”)
```

Exercícios: primeiros programas “oficiais”

- Mostre a mensagem “Hello, World!” (sem aspas).

```
print(“Hello, World!”)
```

- Mostre os números de 5 a 1, um em cada linha, sem utilizar o comando while.

Operadores

- Operadores são utilizados para realizar processos computacionais.
- Alguns já foram apresentados: +, -, * e /.
- Há ainda os operadores **, % e //.

```
print(7**3)
print(7%3)
print(7//3)
```

```
print(7**3) – potência
print(7%3) – resto
print(7//3) – quociente
```

Atenção!

```
print(1//60)
print(1/60)
```

Operadores

- O operador + é também utilizado para concatenar strings.

```
a = “Olá,”

b = “mundo!”

print(a + b)
```

A) Olá,mundo!

B) Olá, mundo!

Operadores

- O operador * é também utilizado para “multiplicar” strings.

```
a = “Teste”

b = 3

print(a * b)
```

A) Traceback

B) TesteTesteTeste

Operadores

- Cuidado ao tentar concatenar **str** com **int**:

```
a = “Teste”                    b = 3

print(a + b)

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

Ordem dos operadores

- Qual o valor final de x?

```
x = 5

x = 3.9 * x * (1 - x)

print(x)
```

- Resultado: -78.0

Ordem dos operadores

- Ordem de precedência:
- **Parênteses**: 2 * (3 - 1) = 4; (1 + 1) ** (5 - 2) = 8.
- **Potência**: 2 ** 1 + 1 = 3, não 4; 3 * 1 ** 3 = 3, não 27.
- **Multiplicação**, **Divisão** e **Resto**: 2 * 3 - 1 = 5, não 4; 6 + 4 / 2 = 8, não 5.
- **Adição** e **Subtração**.

Ordem dos operadores

- Operadores com a mesma precedência são executados da esquerda para a direita: 5 - 3 - 1 = 1, não 3.
- Recomendação: sempre colocar parênteses para deixar clara para você mesmo a ordem de precedência.

Variáveis

- Variável: um nome que se refere a algum valor.
- Uma *declaração de atribuição* cria novas variáveis e fornece valores a elas, ou altera seu conteúdo após sua declaração:
- mensagem = “Todos estão começando a perceber de programação.”
- n = 74
- pi = 3.14159265359 ... Ou pi = 3,14159265359?

Exercício – variáveis

- Declare duas variáveis, pi1 = 3.14159265359 e pi2 = 3,14159265359
- Mostre o conteúdo destas duas variáveis, bem como seus tipos.

```
pi1 = 3.14159265359
pi2 = 3,14159265359
print(pi1)
print(pi2)
print(type(pi1))
print(type(pi2))
```

Constantes

- Valores que nunca mudam ao longo de um programa.
- horas_do_dia = 24
- minutos_por_hora = 60
- minutos_por_dia = horas_do_dia * minutos_por_hora
- print(minutos_por_dia)

Nomes de variáveis e/ou constantes

- Como dar nomes a variáveis ou a constantes?
  - Não podem começar por números: 74alunos_prog_jur = 74;
  - Não podem ter caracteres especiais: 74alun@s_prog_jur = 74;
  - Não podem ser palavras reservadas: class = “74 alunos”.
- Recomendações:
  - Não começar com sublinha/underscore: _
  - Não começar com maiúsculas.

Nomes de variáveis e/ou constantes

- Nomes de variáveis/constantes têm o tamanho desejado pelo programador.
- esta_turma_de_programacao_para_juristas_tem_um_total_de_cinquenta_e_um_alunos_se_nao_estou_em_erro = 51
- Há distinção entre maiúsculas e minúsculas: spam != Spam != SPAM

Nomes de variáveis e/ou constantes

- Geralmente utilizam-se nomes de variáveis e/ou de constantes que tenham significado para o programador.

```
bfhf8j = 40
j6hbyk = 3.75
ejE6Y4 = bfhf8j * j6hbyk
gwL2KS = ejE6Y4 * 4
print(gwL2KS)
```

```
horas_por_semana = 40
valor_por_hora = 3.75
valor_por_semana = horas_por_semana * valor_por_hora
valor_por_mes = valor_por_semana * 4
print(valor_por_mes)
```