FUNDAMENTOS DE PROGRAMAÇÃO

Variáveis, expressões e declarações

Prof. Bruno Góis Mateus

Índice

O NOSSO PRIMEIRO PROGRAMA

O nosso primeiro programa

- print (“Oi mãe, agora sei programar!”)

CONSTANTES

Constantes

- Valores fixos como números, letras, e frases(strings) são chamadas constantes
  - O seu valor não muda

```
>>> print 123
123
>>> print 98.6
98.6
>>> print 'Hello world'
Hello world
```

VARIÁVEIS

Variável

- É um local na memória nomeado onde um programador pode salvar dados
  - Depois recuperar estes dados usando o “nome” da variável
- Programadores escolhem os nomes das variáveis
- É possível mudar o conteúdo da variável em uma declaração futura

Regras para nomes de variáveis em Python

- Deve iniciar com uma letra ou underscore _
- Pode conter nomes números e underscores
- Bom: spam eggs spam23 _speed
- Ruim: 23spam #sign var.12
- Case Sensitive (Diferencia Maísculas de Minúsculas)
  - Diferente: spam Spam SPAM

Palavras reservadas

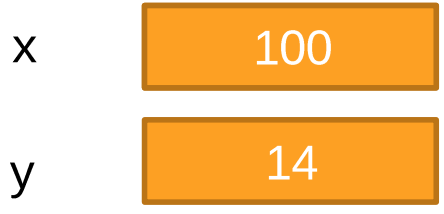
- Palavras que não podem ser usadas para nomear variáveis

and del for is raise assert elif from lambda return break
else global not try class except if or while continue exec
import pass yield def finally in print as with

Instrução de atribuição

- Cria uma nova variável e lhe atribui um valor
- A atribuição é realizado usando operador de atribuição **=**

```
x = 12.2
y = 14
x = 100
```



Instrução de atribuição

- É formada por uma expressão do lado direito do operador e uma variável do lado esquerdo, que armazenará o resultado da expressão.

```
x = 3.9 * x * ( 1 - x )
```

Sentenças

```
x = 2
y = x + 2
print x
```


Instrução de atribuição


Atribuição com expressão


Instrução de impressão (na tela)

Variável

Operador

Constante

Palavra reservada

Instrução de atribuição

1. O lado direito é uma expressão
   - Precisa ser calculada
2. Avalia-se a expressão
3. O resultado da expressão é armazenado
   - Variável do lado direito

x 0.97

```
x = 3.9 * x * ( 1 - x )
x = 3.9 * 0.5 * ( 1 – 0.5 )
x = 3.9 * 0.5 * 0.5
x = 0.97
```

EXPRESSÕES NUMÉRICAS

Expressões numéricas

- Devido a ausência de símbolos usamos a linguagem do computador
  - O asterisco é a multiplicação
  - A exponenciação é bastante diferente do que estamos acostumados

Operador	Operação
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
**	exponenciação
%	Módulo (resto)

Expressões numéricas

```
>>> xx = 2
>>> xx = xx + 2
>>> print xx
4
>>> yy = 440 * 12
>>> print yy
5280
```

```
>>> zz = yy / 1000
>>> print zz
5
>>> jj = 23
>>> kk = jj % 5
>>> print kk
3
>>> print 4 ** 3
64
```

Ordem de avalição

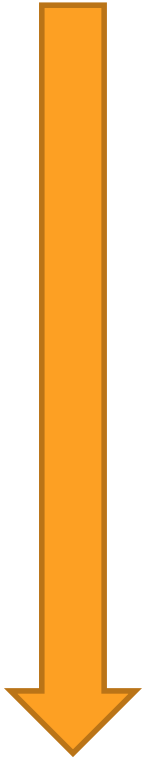
- Quando nós encadeamos operadores numa mesma expressão Python precisa saber o que fazer primeiro.
- Isso é chamado de “precedência de operadores”
- Quais operadores tem “prioridade” sobre os outros?

```
x = 1 + 2 * 3 - 4 / 5 ** 6
```

Precedência de operadores

- Da maior precedência a menor precedência:
  - Parêntesis são sempre respeitados.
  - Exponenciação
  - Multiplicação, Divisão e Módulo(resto)
  - Adição e Subtração
  - Da esquerda para a direita.

( )
**
* /
+ -



Precedência de operadores

```
x = 1 + 2 ** 3 / 4 * 5
x = 1 + 8 / 4 * 5
x = 1 + 2 * 5
x = 1 + 10
x = 11
```

```
()
**
* /
+ -
```

Precedência de operadores

- Use parêntese!!!

Divisão inteira -> Estranho!

- Divisão inteira gera resultados truncados
- Divisão de pontos flutuantes geram pontos flutuantes

```
>>> print 10 / 2
5
>>> print 9 / 2
4
>>> print 99 / 100
0
>>> print 10.0 / 2.0
5.0
>>> print 99.0 / 100.0
0.99
```

Misturando inteiros e pontos flutuantes

- Operações que envolvem inteiros e ponto flutuantes
  - Resultado: ponto flutuante
  - Os inteiros são convertidos em pontos flutuantes antes da operação ser executada

```
>>> print 99 / 100
0
>>> print 99 / 100.0
0.99
>>> print 99.0 / 100
0.99
>>> print 1 + 2 * 3 / 4.0 - 5
-2.5
```

TIPOS DE DADOS

TIPOS DE DADOS

Tipo de dados

- Em Python, variáveis, literais e constantes possuem um tipo
  - Python sabe a diferença entre inteiros e uma cadeia de caracteres (string)
  - Exemplo:
    - Operador +
      - Adição quando usado com inteiros
      - Concatenação quando usado com strings
        - Emendar :D

```
>>> ddd = 1 + 4
>>> print ddd
5
>>> eee = 'hello ' + 'there'
>>> print eee
hello there
```

Tipo de dados

- Python sabe o tipo de tudo
- Algumas operações são proibidas
  - Você não pode adicionar um número a uma string
- Você pode perguntar o tipo um valor

```
>>> eee = 'hello ' +
'there'
>>> eee = eee + 1
Traceback (most recent
call last):
  File "<stdin>", line 1,
in <module>
TypeError: cannot
concatenate 'str' and
'int' objects
>>> type(eee)
<type 'str'>
>>> type('hello')
<type 'str'>
>>> type(1)
<type 'int'>
>>>
```

Os vários tipos de números

- Os números têm dois tipos principais
  - Inteiros são números ~~(inteiros)~~ sem casa decimais
    - -14, -2, 0, 1, 100, 401233
- Números de ponto flutuante possuem casas decimais
  - 2,5, 0,0, 98,6, 14,0
- Existem outros tipos de números
  - São variações de inteiros e pontos flutuantes

```
>>> xx = 1
>>> type (xx)
<type 'int'>
>>> temp = 98.6
>>> type(temp)
<type 'float'>
>>> type(1)
<type 'int'>
>>> type(1.0)
<type 'float'>
```

Conversão de tipos

- Operações que envolvem inteiros e ponto flutuantes resultam em pontos flutuantes
  - Conversão implícita
- Nós podemos controlar esse comportamento
  - Funções int( ) e float( )

Conversão de tipos

```
>>> print float(99) / 100
0.99
>>> i = 42
>>> type(i)
<type 'int'>
>>> f = float(i)
>>> print f
42.0
>>> type(f)
<type 'float'>
>>> print 1 + 2 * float(3) / 4 - 5
-2.5
```

Conversão de string

- Podemos usar as funções int( ) e float( ) para converter string em inteiros ou float
  - Erros podem acontecer no caso da string não conter números

Conversão de string

```
>>> sval = '123'
>>> type(sval)
<type 'str'>
>>> print sval + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int'
>>> ival = int(sval)
>>> type(ival)
<type 'int'>
>>> print ival + 1
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
```

ENTRADA DE DADOS

Entrada de dados

- Nós podemos pedir ao Python uma pausa para ler dados
- raw_input
  - Recebe um argumento opcional
    - Mensagem que irá aparecer na tela
  - Retorna uma string

```
name = raw_input(‘Who are you?’)
print 'Welcome', name
```

Convertendo a entrada de dados

- Se você precisa ler um número a partir da entrada de dados será necessário realizar uma conversão
- Depois trataremos más entradas de dados

```
inp = raw_input('Europe floor?')
usf = int(inp) + 1
print 'US floor', usf
```

COMENTÁRIOS

Comentários

- Qualquer coisa depois de um # é ignorado pelo Python
- Por que comentar?
  - Descrever o que vai acontecer em uma sequência de código
  - Documentar o código ou outras informações complementares
  - Desativar uma linha de código
    - Talvez temporariamente

Comentários

```
# Get the name of the file and open it
name = raw_input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()

# Count word frequency
counts = dict()
for word in words:
    counts[word] = counts.get(word,0) + 1

# Find the most common word
bigcount = None
bigword = None
for word,count in counts.items():
     if bigcount is None or count > bigcount:
          bigword = word
          bigcount = count
# All done
print bigword, bigcount
```

OPERAÇÕES COM STRING

Operações com String

- +
  - Concatenação
- *
  - Múltipla concatenação

```
>>> print 'abc' + '123’
 abc123
 >>> print 'Hi' * 5
 HiHiHiHiHi
```

NOMEANDO VARIÁVEIS

Nomeando variáveis

- Programadores têm a chance de escolher os nomes de variáveis
  - Boas práticas
- Devemos nomear variáveis para nos ajudar
  - Lembrar o que elas fazem a lembrar
  - Lembrar o que devemos armazenar

Nomeando variáveis

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd *
x1q3z9afd
print x1q3p9afd
```

```
a = 35.0
b = 12.50
c = a * b
print c
```

```
hours = 35.0
rate = 12.50
pay = hours * rate
print pay
```

EXERCÍCIOS

Exercício 1

- Qual a ordem de avaliação da operações aritméticas na seguinte expressão:
  - 2 + (3 – 1) * 10/5 * (2 + 3)

Exercício 2

- Adicione parênteses na expressão a seguir de forma que o resultado da avaliação seja -6
  - 6 * 1 - 2

Exercício 3

- Escreva um programa que recebe dois números e calcula a soma desses números e a média aritmética

Exercício 4

- Escreva um programa que pergunta a quantidade de horas trabalhas e o valor de cada hora trabalha e mostre na tela o valor a ser pago
- Ex:
  - Quantidade de horas trabalhas: 35
  - Valor da hora trabalhada: 2.75
  - Valor a ser pago: 96.25

Exercício 5

- Escreva um programa que pergunte a distância percorrida pelo veículo e a quantidade de combustível gasto. O programa deve mostrar o consumo por litro de combustível.

Exercício 6

- Escreva um programa que pergunte a distância percorrida pelo veículo e calcule quanto tempo será necessário para percorrer essa distância considerando as seguintes velocidades médias:
  - 60 km/h
  - 80 km/h
  - 100 km/h

Exercício 7

- Escreva um programa que recebe uma quantidade de tempo em segundos e mostre essa quantidade de tempo no seguinte formato HH:mm:ss
- Exemplo:
  - 3600 segundos -> 01:00:00

Exercício 8

- Considere que você utiliza o relógio no formato 24 horas.
- Você deseja colocar um novo alarme, porém o seu aplicativo é muito paia. Não possibilita escolher a data. A única opção é informar quantas horas depois o alarme deve tocar.
- Escreva um programa que ao receber o tempo em horas, informe a que horas o alarme vai tocar.
  - Considere a hora atual do computador
  - Desafio:
    - Considerando a data atual, que dia o alarme irá tocar também

A seguir

- Estruturas condicionais