

Variáveis, Expressões e Declarações

Prof. Max Vieira Santiago

Constantes

- **Valores fixos** como números, letras e frases (strings) são chamadas “**constantes**”, pois o seu valor não muda
- **Constantes** numéricas são como você espera
- Strings (cadeias de caracteres) usam aspas únicas (') ou aspas duplas (")

```
>>> print(123)
```

```
123
```

```
>>> print(98.6)
```

```
98.6
```

```
>>> print('Hello world')
```

```
Hello world
```

Variáveis

- Uma **variável** é um local nomeado na memória onde um programador pode armazenar dados e depois recuperar estes dados usando o “nome” da **variável**
- Programadores podem escolher os nomes das **variáveis**
- Você pode mudar o conteúdo da **variável** em uma declaração futura

x = 12.2

y = 14

x 12.2

y 14

Variáveis

- Uma **variável** é um local nomeado na memória onde um programador pode armazenar dados e depois recuperar estes dados usando o “nome” da **variável**
- Programadores podem escolher os nomes das **variáveis**
- Você pode mudar o conteúdo da **variável** em uma declaração futura

x = 12.2

y = 14

x = 100

x

~~12.2~~ 100

y

14

Regras Para Nomes de Variáveis no Python

- Deve iniciar com uma letra ou underscore _
- Pode conter letras, números e underscores
- Case Sensitive (diferencia maiúsculas de minúsculas)
- **BOM:** spam eggs spam23 _speed max_speed
- **MAL:** 23spam #sign var.12
- **Diferentes:** spam Spam SPAM

Palavras Reservadas

- Você não pode usar **palavras reservadas** como nomes de variáveis ou identificadores.

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

Declarações

Fluxo de execução



x = 2



Declaração de atribuição

x = x + 2



Expressão com atribuição

print(x)



Declaração de impressão

Variável

Operador

Constante

Palavras
Reservadas

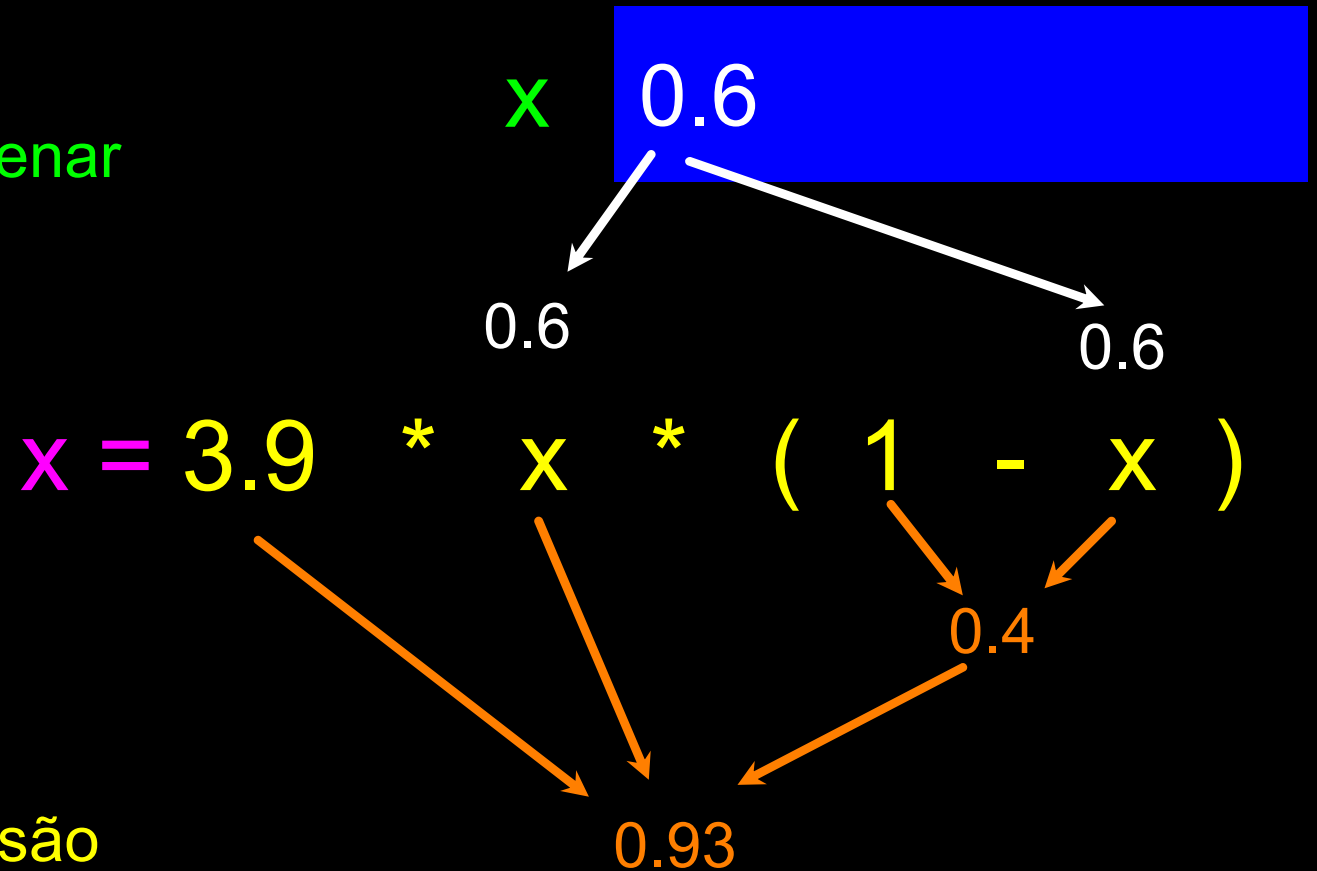
Python Tutor: <https://goo.gl/rfVBGZ>

Declarações de Atribuição

- Nós atribuímos um valor a uma variável usando o **operador de atribuição (=)**
- Uma **declaração de atribuição** é formada por uma **expressão do lado direito do operador de atribuição (=)** e uma **variável** do lado esquerdo, que armazenará o resultado da expressão.

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

Uma variável é um local na memória usado para armazenar um valor (0.6)



O lado direito é uma expressão
Quando a expressão é processada, o
resultado é colocado em (atribuído a)
x.

Uma variável é um lugar na memória utilizado para armazenar um valor.

O valor armazenado em uma variável pode ser atualizado, substituindo o valor antigo (0.6) com o novo valor (0.93)



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

O lado direito é uma expressão
Quando a expressão é processada, o resultado é colocado em (atribuído a) x.

0.93

A red arrow points from the '0.93' text to the 'x' in the equation $x = 3.9 * x * (1 - x)$.

Variáveis em Expressões

- Uma **variável** deve ser inicializada (armazenar algo) antes de ser utilizada em uma **expressão**

Qual o valor da variável **y**?
Essa variável existe?

x = 100

x = **x** + **y**

x

100

y

?

Variáveis em Expressões

- Uma **variável** deve ser inicializada (armazenar algo) antes de ser utilizada em uma **expressão**

```
wage = 1000
bills = 200
rent = 500
food = 200
savings = wage - bills - rent - food
```

Python Tutor: <https://bit.ly/3r1Ej61>

Expressões Numéricas

- Por falta de símbolos matemáticos nos teclados de computadores, nós usamos a “linguagem computacional” para expressar as operações clássicas da matemática.

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

Expressões Matemáticas

```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4
>>> yy = 440 * 12
>>> print(yy)
5280
>>> zz = yy / 1000
>>> print(zz)
5.28
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print(kk)
3
>>> print(4 ** 3)
64
```

$$\begin{array}{r} 4 \text{ R } 3 \\ 5 \overline{) 23} \\ \underline{20} \\ 3 \end{array}$$

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

Ordem de Execução

- Quando nós encadeamos operadores numa mesma expressão, o 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
```

Regras de Precedência de Operadores

Regra de maior precedência a menor precedência:

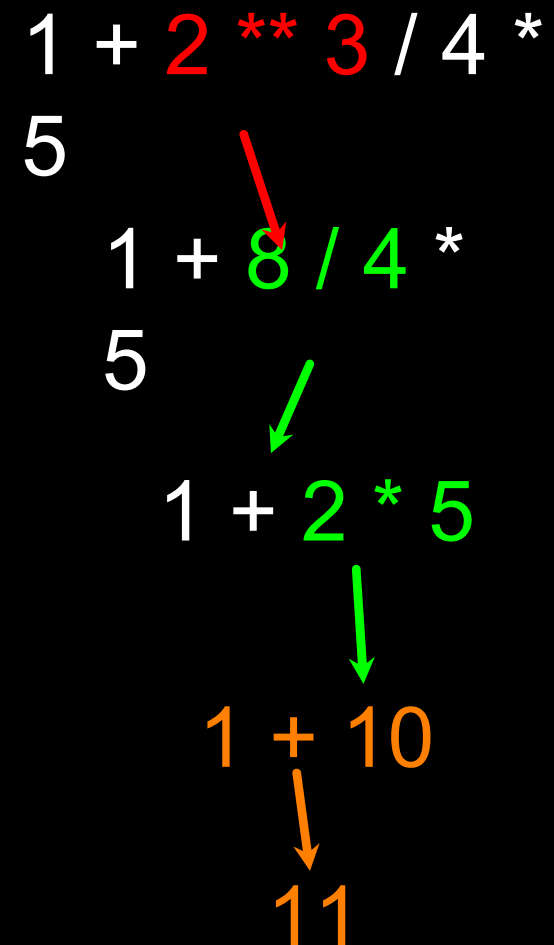

- Parêntesis são sempre respeitados
- Exponenciação (elevado a potência)
- Multiplicação, Divisão e Módulo(resto)
- Adição e Subtração
- Da esquerda para a direita.

Parêntesis
Potência
Multiplicação
Adição
Esquda para
Direita




```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11
>>>
```

Parêntesis
Potência
Multiplicação
Adição
Esquada para
Direita



Precedência dos Operadores

- Lembre a regra, de cima para baixo
- Ao escrever o código - **use parênteses**
- Ao escrever o código - mantenha as expressões matemáticas simples o suficiente para que elas sejam fáceis de entender
- Quebrar uma longa série de operações matemáticas até torná-las mais claras
 - Armazenar resultados de operações intermediárias em **variáveis**

Parêntesis
Potência
Multiplicação
Adição
Esquerda para Direita



O que significa “Type (tipo)”?

- Em Python, variáveis e constantes têm um “type” (tipo)
- Python sabe a diferença entre um número e uma string
- Por exemplo: “+” significa “adicionar” se algo for um número e “concatenar” se algo for uma string

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

concatenar = juntar, colocar junto

O Type Faz Diferença

- Python sabe o “type” de tudo
- Algumas operações são proibidas
- Você não pode “adicionar 1” a uma string
- Podemos perguntar ao Python o tipo de algo usando a função `type()`

```
>>> 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'>
>>>
```

Vários Tipos de Números

- Números possuem dois tipos principais:
- Inteiros são todos os números:
-14, -2, 0, 1, 100, 401233
- Números de Ponto Flutuante têm partes decimais:
-2.5 , 0.0, 98.6, 14.0
- Existem outros tipos de números - eles são variações de **float** e **integer**

```
>>> 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

- Ao colocar um inteiro e um float em uma expressão, o inteiro será **implicitamente** convertido para float
- Você pode controlar isso com as funções embutidas `int()` e `float()`

```
>>> print(float(99) / 100)
0.99
>>> i = 42
>>> type(i)
<type 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<type 'float'>
>>>
```

Conversão de String

- Você também pode usar `int()` e `float()` para converter entre strings e inteiros
- Você irá obter um **erro** se a string não possuir **apenas** caracteres numéricos

```
>>> 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
>>> nsval = 'hello bob'
>>> niv = int(nsval)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
```

Dados Inseridos pelo Usuário

- Podemos instruir o Python para pausar e ler dados do usuário utilizando a função `input()`
- A função `input()` retorna uma string

```
nam = input('Who are you? ')\nprint ('Welcome', nam)
```

Who are you? **Chuck**
Welcome Chuck

Convertendo dados inseridos pelo usuário



- Se queremos ler um número inserido pelo usuário, devemos converter a string para número usando a função de conversão de tipos
- Em aulas futuras, vamos lidar com dados de entrada incorretos

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

Europe floor? 0
US floor 1

Formatação da Saída

- Considere o código abaixo:

```
>>> print(1/3)  
0.3333333333333333
```

- Existe alguma forma de melhorar a exibição do resultado?
 - Exemplo: exibir apenas **duas** casas decimais

Formatação da Saída

```
>>> print('Olá %s, você comprou %d items e gastou R$ %.2f' %  
('Guilherme', 5, 14.538))
```

Olá **Guilherme**, você comprou **5** items e gastou R\$ **14.54**

- Placeholder:
 - **%s** : exibe uma **string**
 - **%d**: exibe um **número inteiro**
 - **%.xf**: exibe um **número de ponto flutuante** com **x** casas decimais
- Formato geral:
 - `print('mensagem formatada' % (constantes, variáveis ou expressões))`

Comentários em Python

- Tudo depois de `#` é ignorado pelo Python
- Por que comentar?
 - Descrever o que vai acontecer em uma sequência de código
 - Documentar quem escreveu o código ou outras informações auxiliares
 - Desconsiderar uma linha de código - talvez temporariamente

```
# Obtém o nome do arquivo e o abre
name = input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()

# Contar a frequência de uma palavra
counts = dict()
for word in words:
    counts[word] = counts.get(word,0) + 1

# Encontrar a palavra mais comum
bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# tudo feito
print(bigword, bigcount)
```

Operações de String

- Alguns operadores se aplicam a strings
 - `+` significa “concatenação”
 - `*` significa “múltiplas concatenações”
- Python sabe quando ele está lidando com uma string ou um número e se comporta de forma adequada

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

Nomes de Variáveis

Mnemônicas

- Uma vez que nós programadores escolhemos como nomear nossas variáveis, há um pouco de “boas práticas”
- Nós nomeamos as variáveis para ajudar a nos lembrar o que pretendemos armazenar nela (“mnemônico” = “fácil de ser memorizado”)
- Isso pode confundir estudantes iniciantes, pois variáveis bem nomeadas, muitas vezes “soam” tão bem que parecem ser palavras-chave

<https://pt.wikipedia.org/wiki/Mnem%C3%B3nica>

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

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

```
hora = 35.0
taxa = 12.50
pagamento = hora * taxa
print(pagamento)
```


Exercício 1

Escreva um programa que solicita ao usuário a quantidade de horas trabalhadas e a taxa da hora trabalhada e calcula o salário bruto.

Exemplo:

Entre com as horas: 35

Entre com a taxa: 2.75

Pagamento: 96.25

Resposta

```
horas = input("Entre com as horas:")  
taxa = input("Entre com a taxa:")  
pagamento = int(horas) * float(taxa)  
print("Pagamento: %.2f Kg" % (pagamento))
```

Exercício 2

Escreva um programa que solicita ao usuário uma temperatura em graus Fahrenheit e imprima na tela a temperatura convertida em graus Celsius. ($T_{(F)} = T_{(C)} * 9/5 + 32$).

Entre com a temperatura em graus Fahrenheit: 32
Temperatura em graus Celsius: 0

Resposta

```
fahrenheit = input("Entre com a temperatura em graus  
Fahrenheit: ")  
celsius = (float(fahrenheit) - 32) * (5 / 9)  
print("Temperatura em graus Celsius: %.2f" % celsius)
```

Exercício 3

Escreva um programa que solicita ao usuário um inteiro de três algarismos e imprima na tela o seu valor invertido.

Digite um inteiro de três algarismos: 123

Valor invertido: 321

Resposta

```
numero = input("Digite um inteiro de três  
algarismos:")  
numero = int(numero)  
  
unidade = numero % 10  
dezena = (numero % 100) // 10  
centena = numero // 100  
  
invertido = 100 * unidade + 10 * dezena + centena  
  
print("Valor invertido:", invertido)
```



Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and open.umich.edu and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Translation: Jonathan Cardozo, Luis Otávio, Silvio L Carvalho, Maria A. Pysklevicz Sakiyama

Estes slides são de autoria 2010 - Charles R. Severance (www.dr-chuck.com), da Universidade de Michigan Escola de Informação e open.umich.edu e disponibilizado através de uma Licença Creative Commons Attribution 4.0. Por favor manter este último slide em todas as cópias do documento em conformidade com os requisitos de atribuição da licença. Se você fizer uma alteração, sinta-se livre para adicionar seu nome e organização à lista de contribuidores nesta página.

Desenvolvimento inicial: Charles Severance, da Universidade de Michigan School of Information

Tradução: Jonathan Cardozo, Luis Otávio, Silvio L Carvalho, Maria A. Pysklevicz Sakiyama.