

Variáveis, Expressões e Declarações

Capítulo 2



Python for Everybody
www.py4e.com



Constantes

- **Valores fixos**, como números, letras e strings, são chamados de "**constantes**" porque seu valor não muda
 - As **constantes** numéricas são como você espera
 - As **constantes** de string usam aspas simples (') ou aspas duplas (")
- ```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
```

# Palavras reservadas

Você não pode usar **palavras reservadas** como nome ou identificadores de variáveis

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

# Variáveis

- A **variável** is a named place in the memory where a programmer can store data and later retrieve the data using the **variável** “name”
- Programmers get to choose the names of the **variáveis**
- You can change the contents of a **variável** in a later statement

**x** = 12.2

**y** = 14

**x**

12.2

**y**

14

# Variáveis

- A **variável** is a named place in the memory where a programmer can store data and later retrieve the data using the **variável** “name”
- Programmers get to choose the names of the **variáveis**
- You can change the contents of a **variável** in a later statement

**x** = 12.2

**y** = 14

**x** = 100

**x**

~~12.2~~ 100

**y**

14

# Python Variable Name Rules

- Must start with a letter or underscore \_
- Must consist of letters, numbers, and underscores
- Case Sensitive

Good:       spam       eggs       spam23       \_speed

Bad:        23spam       #sign       var.12

Different:       spam       Spam       SPAM

# Mnemonic Variable Names

- Since we programmers are given a choice in how we choose our variável names, there is a bit of “best practice”
- We name variáveis to help us remember what we intend to store in them (“mnemonic” = “memory aid”)
- This can confuse beginning students because well-named variáveis often “sound” so good that they must be keywords

<http://en.wikipedia.org/wiki/Mnemonic>

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

O que esse  
pedaço de código  
está fazendo?



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

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

O que esses bits  
de código estão  
fazendo?

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

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

O que esses bits  
de código estão  
fazendo?

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

# Sentenças ou Linhas

`x = 2`



Declaração de atribuição

`x = x + 2`



Atribuição com expressão

`print(x)`



Declaração de impressão

Variável

Operador

Constante

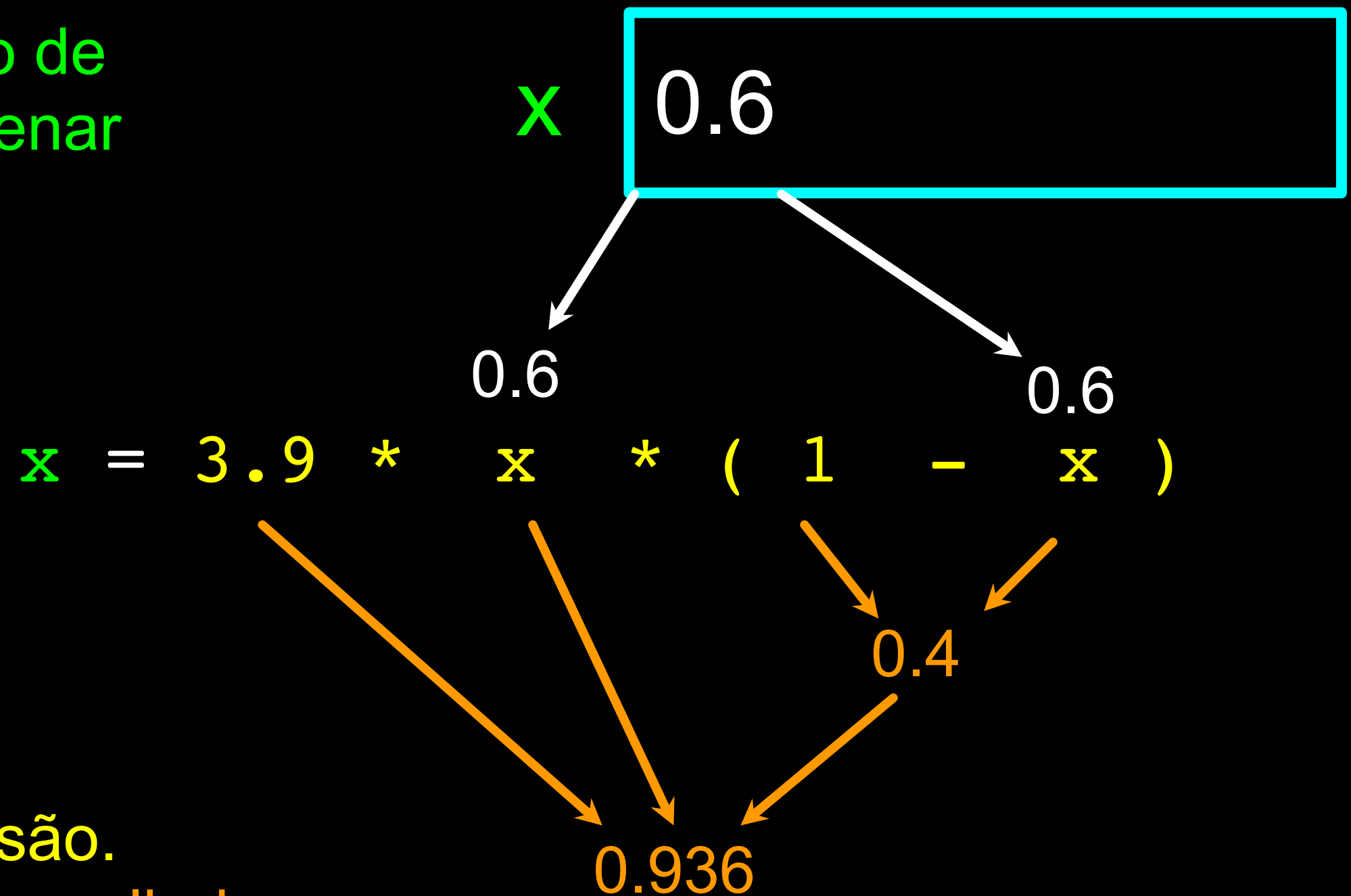
Função

# Declarações de Atribuição

- Atribuímos um valor a uma variável usando a declaração de atribuição (=)
- Uma declaração de atribuição consiste em uma **expressão no lado direito** e uma **variável** para armazenar o resultado

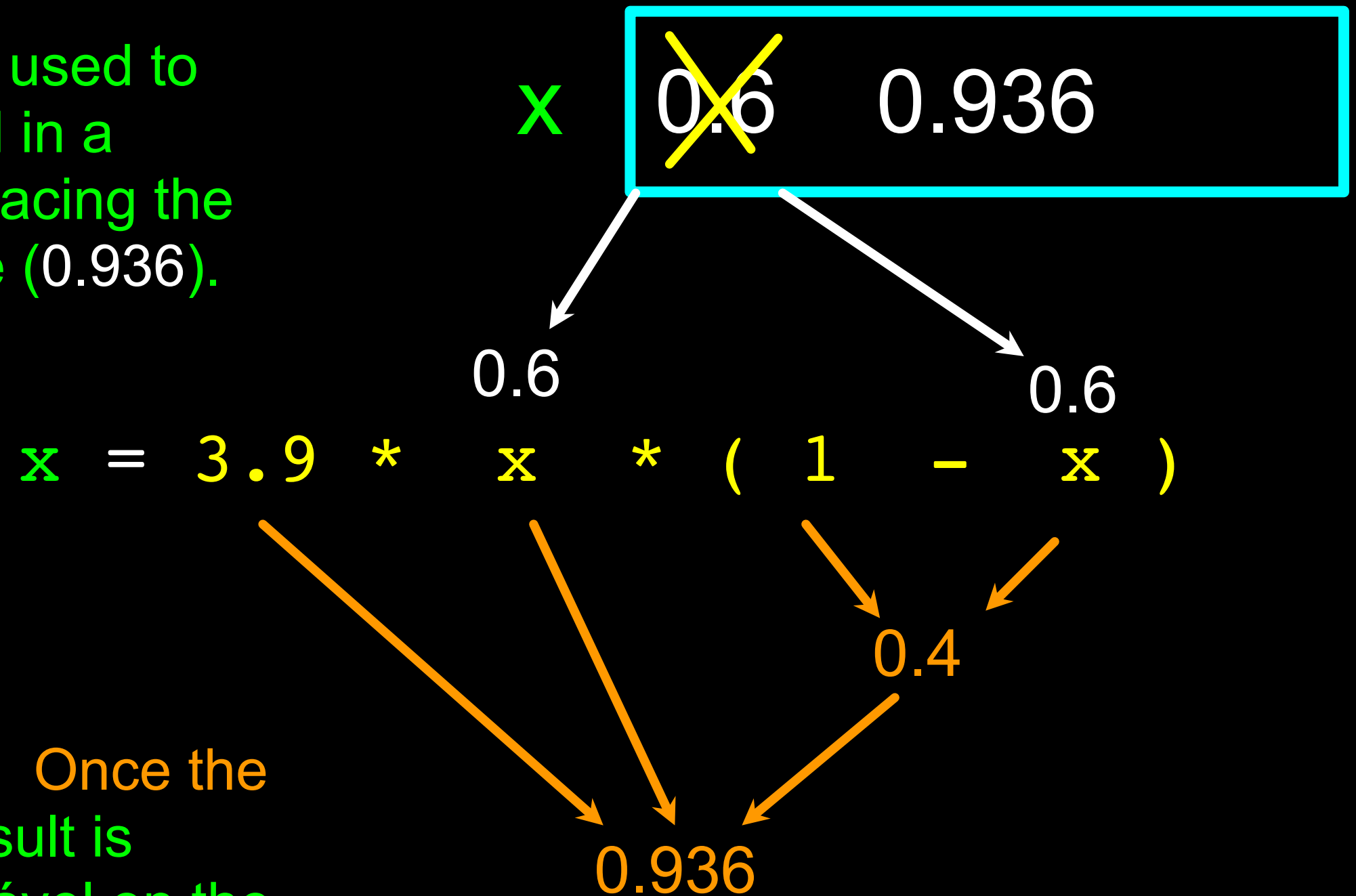
**x** = 3.9 \* **x** \* ( 1 - **x** )

Uma variável é uma posição de memória usada para armazenar um valor (0.6)



O lado direito é uma expressão.  
Uma vez que a expressão é avaliada, o resultado é colocado em (atribuído a)  $x$ .

A variável is a memory location used to store a value. The value stored in a variável can be updated by replacing the old value (0.6) with a new value (0.936).



The right side is an expression. Once the expression is evaluated, the result is placed in (assigned to) the variável on the left side (i.e., x).

Expressões...

# Expressões Numéricas

- Devido à falta de símbolos matemáticos nos teclados de computadores - usamos "linguagem do computador" para expressar as operações matemáticas clássicas
- Asterisco é multiplicação
- Exponenciação (elevar a uma potência) parece diferente do que em matemática

| Operador | Operação      |
|----------|---------------|
| +        | Adição        |
| -        | Subtração     |
| *        | Multiplicação |
| /        | Divisão       |
| **       | Potência      |
| %        | Resto         |



# Expressões Numéricas

```
>>> 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
```

5  $\overline{) 23}$   
20  

---

3

4 R 3

| Operator | Operation      |
|----------|----------------|
| +        | Addition       |
| -        | Subtraction    |
| *        | Multiplication |
| /        | Division       |
| **       | Power          |
| %        | Remainder      |

# Ordem de Avaliação

- Quando reunimos operadores - Python deve saber qual deles calcular primeiro
- Isso é chamado de “precedência de operador”
- Qual operador tem precedência sobre os outros?


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

# Regras de Precedência de Operador

Regra de precedência mais alta para regra de precedência mais baixa:

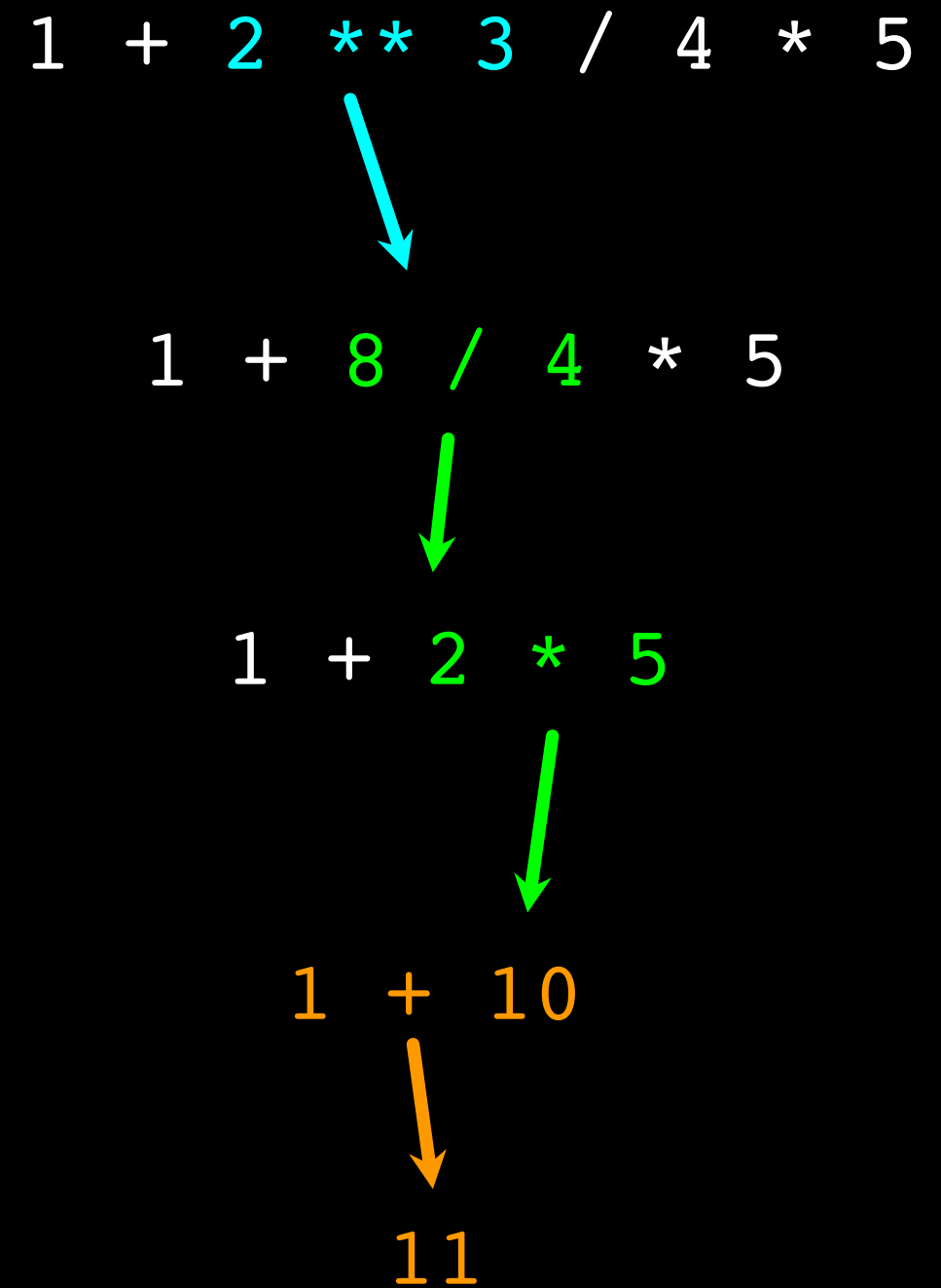

- Parênteses são sempre respeitados
- Exponenciação (elevar a potência)
- Multiplicação, Divisão e Resto
- Adição e Subtração
- Esquerda para a direita

Parênteses  
Potência  
Multiplicação  
Adição  
Esquerda  
para Direita



```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
>>>
```

Parênteses  
Potência  
Multiplicação  
Adição  
Esquerda  
para Direita



# Precedência de Operador

- Lembre-se das regras de cima para baixo
- Ao escrever código - use parênteses
- Ao escrever código - mantenha expressões matemáticas simples o suficiente para que sejam fáceis de entender
- Quebre longas séries de operações matemáticas para torná-las mais claras

Parênteses  
Potência  
Multiplicação  
Adição  
Esquerda  
para Direita



# O que significa “tipo”?

- Em Python variáveis, literais e constantes têm um “tipo”
- Python sabe a diferença entre um número inteiro e uma string
- Por exemplo, “+” significa “adição” se algo for um número e “concatenar” se algo for uma sequência

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

concatenar = colocar junto

# O Tipo importa

- Python sabe o “**type**” de tudo
- Algumas operações são proibidas
- Você não pode “adicionar 1” a uma string
- Podemos perguntar a Python qual o tipo de alguma coisa com a função **type()**

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>TypeError: Can't convert
'int' object to str implicitly
>>> type(eee)
<class 'str'>
>>> type('hello')
<class 'str'>
>>> type(1)
<class 'int'>
>>>
```

# Diferentes Tipos de Números

- Números têm dois tipos principais:
  - **Integers** números inteiros:  
-14, -2, 0, 1, 100, 401233
  - **Floating Point Numbers** tem partes decimais: -2.5 , 0.0, 98.6, 14.0
- Existem outros tipos de números - são variações de número flutuante e número inteiro

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



# Conversão de Tipos

- Quando você coloca um número inteiro e um ponto flutuante em uma expressão, o número inteiro é **implicitamente** convertido em um número flutuante
- Você pode controlar isso com as funções internas `int ()` e `float ()`

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>>
```

# Divisão de Inteiros

A divisão de inteiros produz um resultado de ponto flutuante

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

Isso era diferente no Python 2.x

# Conversão de Strings

- Você também pode usar `int()` e `float()` para converter entre seqüências de caracteres e números inteiros
- Você terá um **erro** se a seqüência não contiver caracteres numéricos

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class '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()
with base 10: 'x'
```

# Entrada do Usuário

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

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

Who are you? **Chuck**  
Welcome Chuck

# Convertendo a Entrada do Usuário



- Se quisermos ler um número do usuário, devemos convertê-lo de uma string para um número usando uma função de conversão de tipo
- Mais tarde, lidaremos com dados de entrada incorretos

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

Europe floor? 0  
US floor 1

# Comentários em Python

Qualquer coisa depois de um # 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
- Desativar uma linha de código - talvez temporariamente

```
Get the name of the file and open it
name = input('Enter file:')
handle = open(name, 'r')

Count word frequency
counts = dict()
for line in handle:
 words = line.split()
 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)
```

# Resumo

- Tipo
- Palavras reservadas
- Variáveis (mnemônicas)
- Operadores
- Operador precedente
- Divisão Inteira
- Conversão entre tipos
- Entrada do usuário
- Comentários (#)



## Exercício

Escreva um programa para solicitar ao usuário horas e taxas por hora para calcular o salário bruto.

Digite as horas: 35

Digite a taxa: 2.75

Pagamento: 96.25



# Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance ([www.dr-chuck.com](http://www.dr-chuck.com)) of the University of Michigan School of Information 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

Traduzido para o Português Brasileiro por Filipe Calegario

...