Variáveis e Entrada de Dados

fmasanori@gmail.com

Primeiro programa

```
>>> print ("Primeira mensagem!")
Primeira mensagem!
>>>
```

- Este programa possui apenas uma linha de código
- Observe que as aspas não aparecem na saída
- Precisamos marcar ou limitar o início e o fim de nossas mensagens com um símbolo, nesse caso, as aspas

Primeira mensagem de erro

```
>>> Print ("Primeira mensagem!")
Traceback (most recent call last):
  File "<pyshell#39>", line 1, in <module>
    Print ("Primeira mensagem!")
NameError: name 'Print' is not defined
>>>
```

- Letras maiúsculas e minúsculas são diferentes
- Você reparou que Print não está na cor roxa?

Primeira mensagem de erro

```
>>> print (Primeira mensagem)
SyntaxError: invalid syntax (<pyshell#7>, line 1)
>>>
```

- Se não utilizarmos aspas, o computador interpretará nossa mensagem como um comando da linguagem Python, gerando um erro de sintaxe
- Você reparou que a mensagem não está na cor verde?

Primeira mensagem de erro

```
>>> print "Primeira mensagem!"
SyntaxError: Missing parentheses in cal
l to 'print'. Did you mean print("Prime
ira mensagem!")?
```

- Na versão do Python que usamos os parênteses não são opcionais no print

Primeira mensagem de erro

```
>>>  print ("Primeira mensagem!")

SyntaxError: unexpected indent
```

- Os espaços iniciais possuem um significado em Python que veremos mais adiante

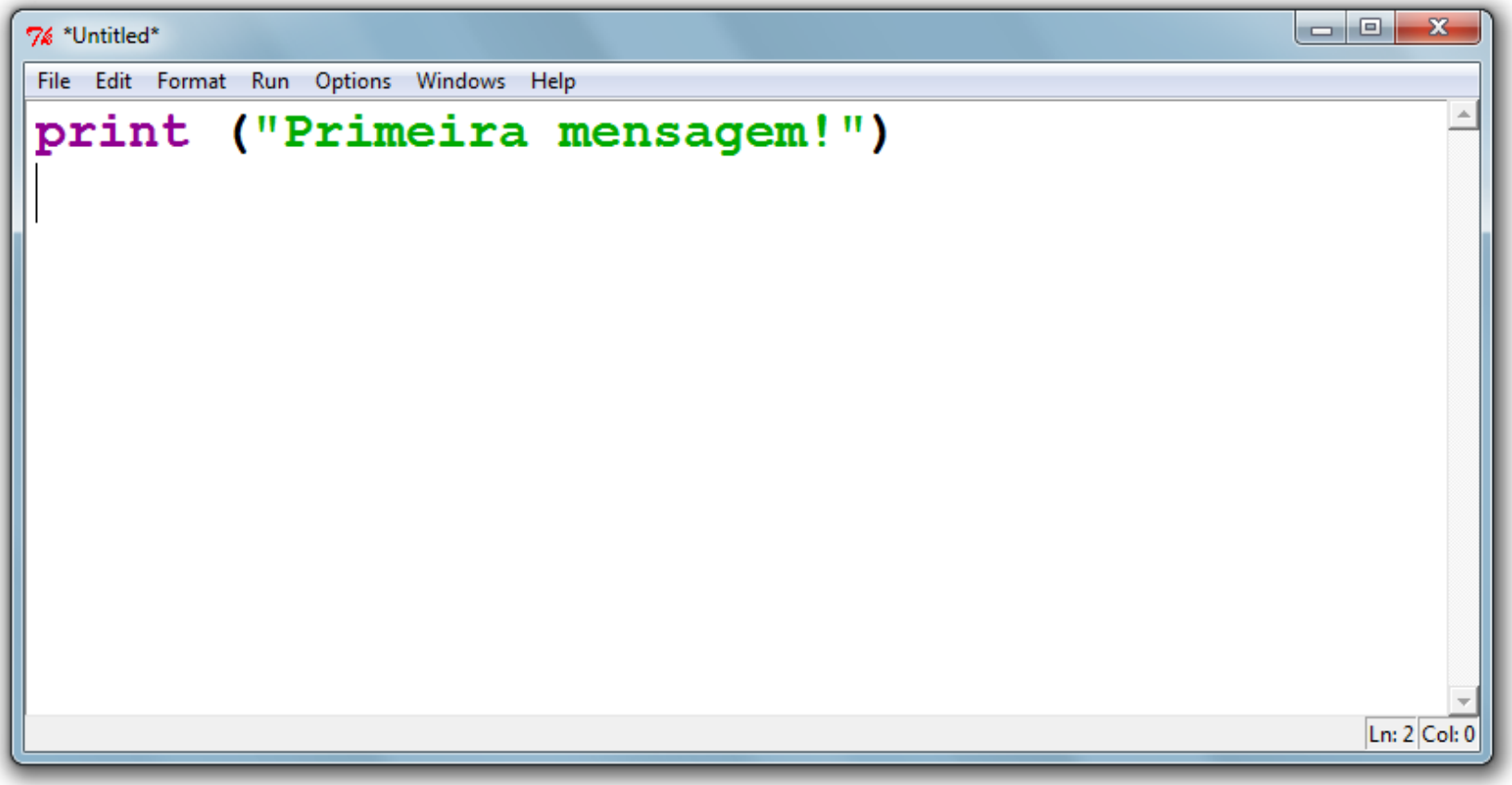
Interpretador Python

- O interpretador é um programa que aceita comandos escritos em Python e os executa, linha a linha
- Sem o interpretador, nossos programas não podem ser executados, sendo considerados apenas um texto
- O interpretador verifica se escrevemos corretamente o programa, mostrando mensagens de erro caso haja algum problema

Interpretador Python

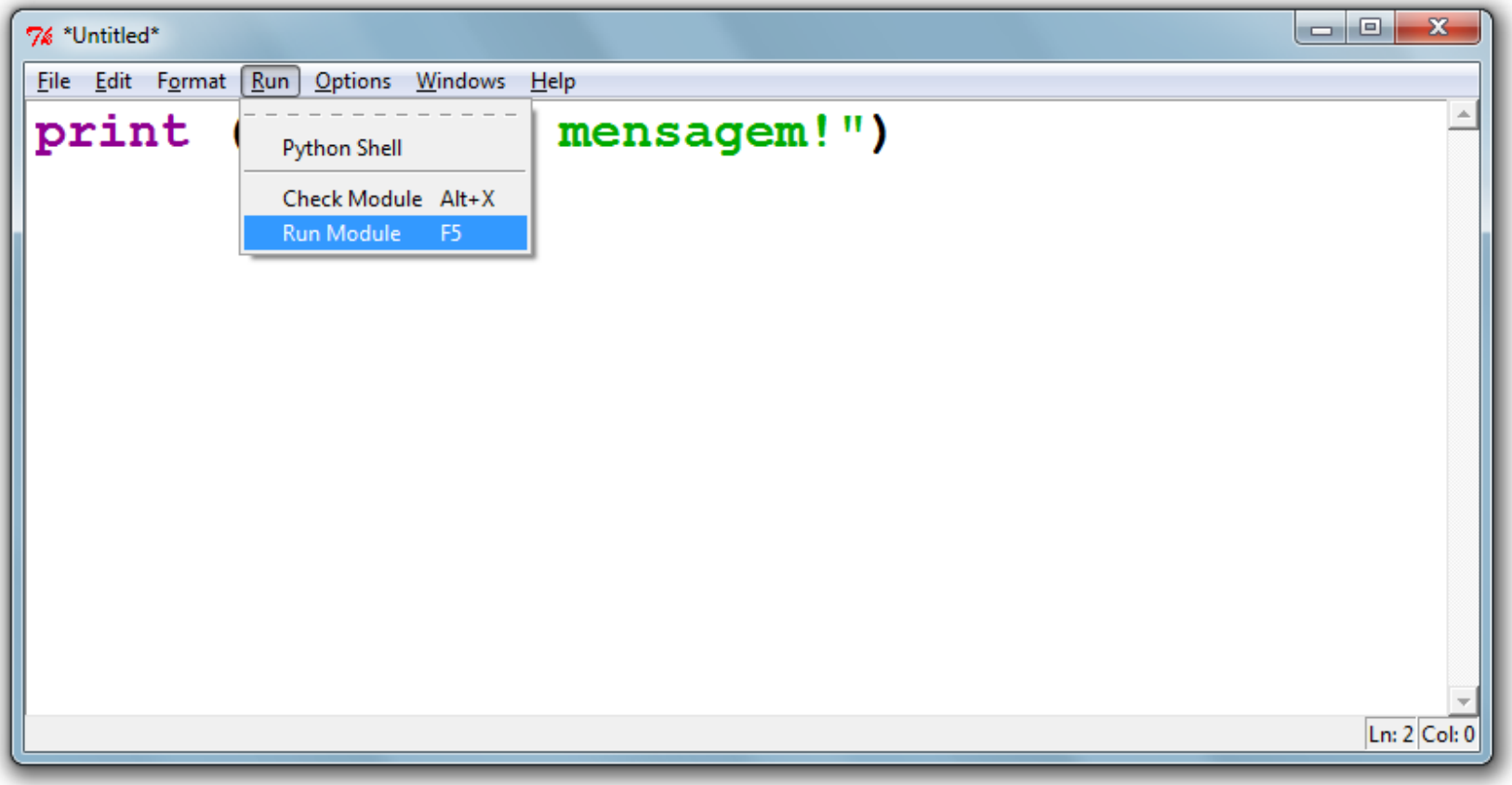
- Existem dois modos do interpretador Python: modo interativo e modo de edição
- Usamos nos exemplos anteriores o modo interativo
- Uma vantagem do modo interativo é poder testar comandos e obter a resposta instantaneamente

Modo edição

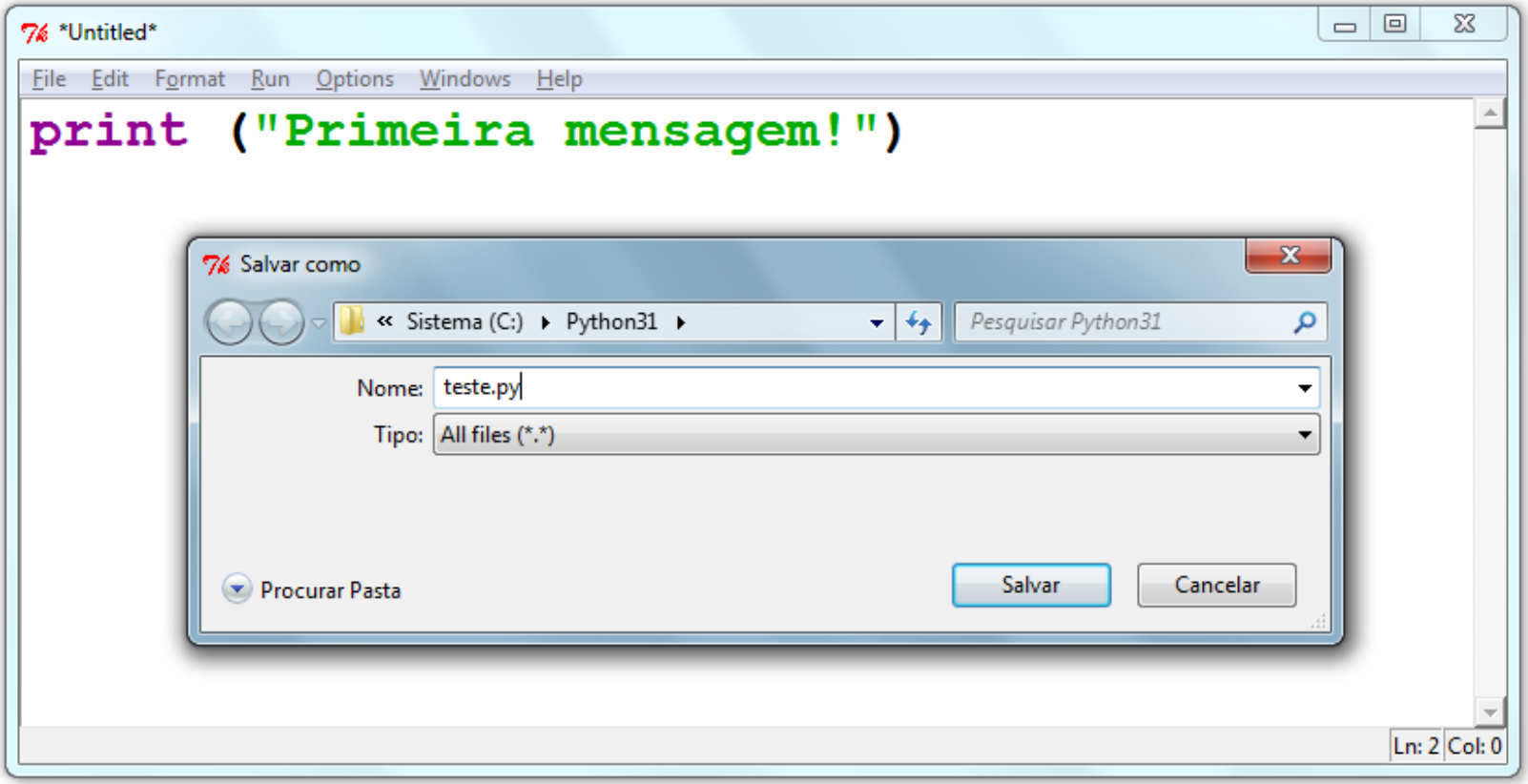


Observe que não aconteceu nada ao digitar enter no final da linha
É necessário “rodar” o programa no modo edição (Run Module F5)

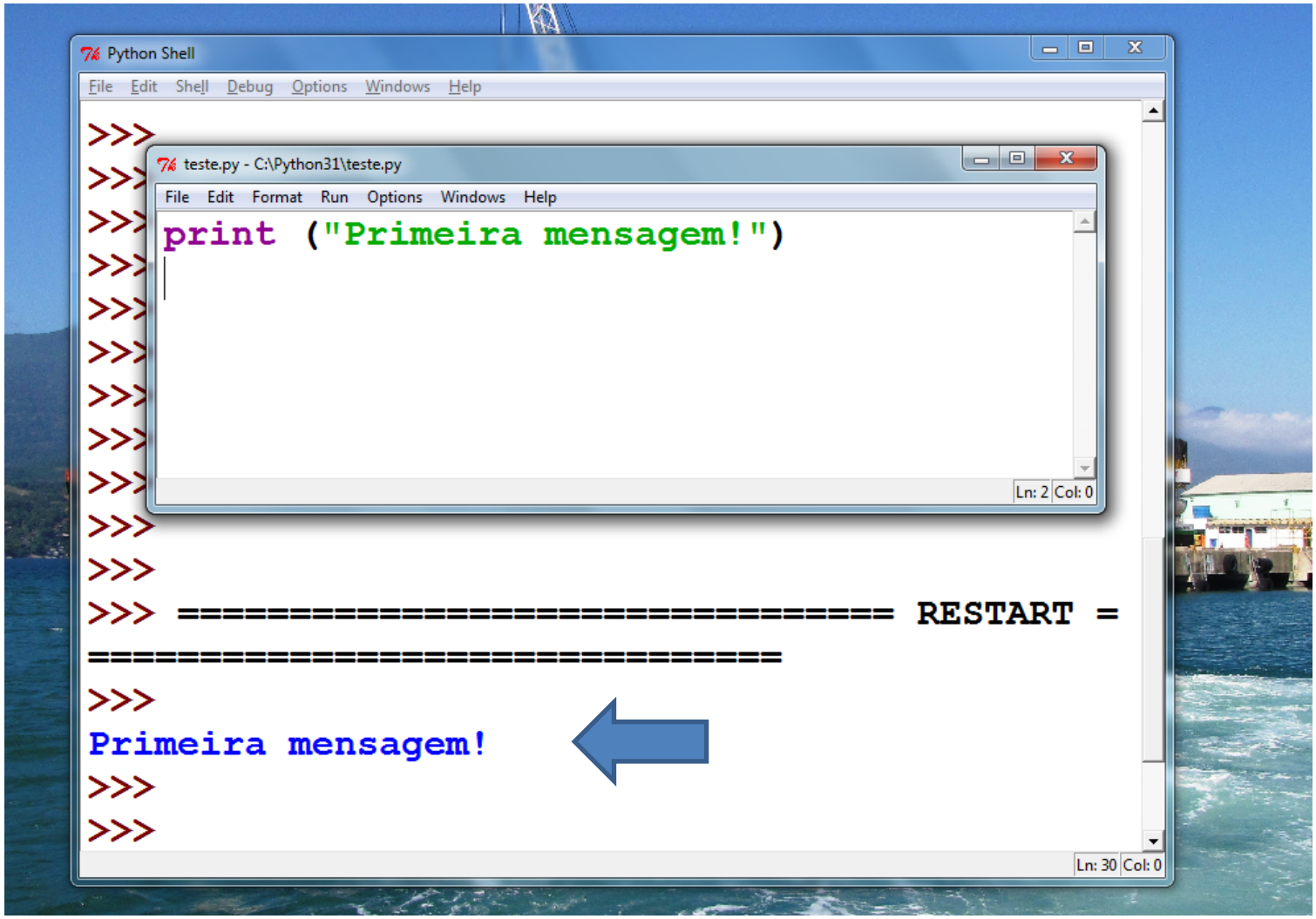
Rodar o programa



Salvar o programa



Mesmo resultado!



Cuidados ao digitar programas

- Letras maiúsculas e minúsculas são diferentes
- Aspas são muito importantes e não devem ser esquecidas. Todas vez que você abrir aspas, não esqueça de fechá-las
- Parênteses não são opcionais em Python. Todo parênteses aberto deve ser fechado.
- Espaços são muito importantes. A linguagem Python se baseia na quantidade de espaços em branco antes do início de cada linha para realizar diferentes operações.

Calculadora no interpretador

```
>>> 2+3
5
>>> 5-3
2
>>> 10-4+2
8
>>> 2*10
20
>>> 10/4
2.5
>>> 2**3
8
>>> 10%3
1
>>> 16%7
2
```

Não esqueça de dar enter

Asterisco para multiplicação

Barra para divisão

Exponenciação

Resto da divisão

Conceitos de variáveis e atribuição

- Variáveis são utilizadas para armazenar valores e para dar nome a uma área da memória do computador
- O símbolo para atribuição é o igual (=)

```
a = 2          a recebe 2
b = 3          b recebe 3
print (a+b)    imprima a + b
```

Conceitos de variáveis e atribuição

- Como em matemática, passamos parâmetros ou valores para uma função usando parênteses
- Função ***f(x)***, onde ***f*** é o nome da função e ***x*** um parâmetro
- No exemplo anterior print é o nome da função e a + b, o valor passado como parâmetro

Conceitos de variáveis e atribuição

- Podemos usar o modo interativo também

```
>>> a = 2
>>> b = 3
>>> print (a+b)
5
```

- As duas primeiras linhas não enviam nada para a tela, por isso, apenas o resultado da terceira linha é mostrado

Conceitos de variáveis e atribuição

- Você pode estar se perguntando por que criamos duas variáveis, a e b, para somar dois números?
- Poderíamos ter obtido o mesmo resultado de diversas formas

```
>>> print (2+3)
5
>>> print (5)
5
```

Conceitos de variáveis e atribuição

- Qual é a diferença entre o primeiro modo e os dois últimos?
- No primeiro caso está incluído a lógica que usamos para obter o resultado
- Deixamos assim explícito o algoritmo que usamos mentalmente para resolver esse problema
- Nos dois últimos casos apenas ordenamos que o computador imprima algo concreto, sem deixar claro a lógica para chegar naquele resultado

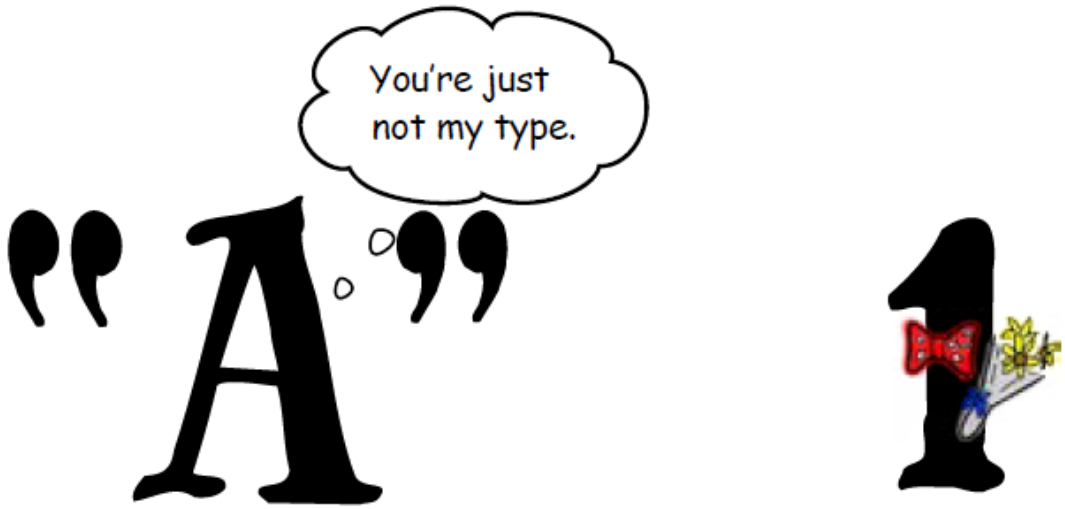
Conceitos de variáveis e atribuição

1. Seu salário atual é de R$ 6500 reais. Faça um programa que calcule o novo salário com um aumento de 5%
2. Escreva um programa que exiba seu nome na tela
3. Calcule a soma de três variáveis
4. O que acontece se eu colocar textos nas três variáveis anteriores?

Nomes de variáveis

- Em Python, nomes de variáveis devem iniciar obrigatoriamente com uma letra ou caracter sublinhado (_)
- Acentos são permitidos!
- Exemplo de nomes válidos: preço, ação, salário, _x, ano_2011, salário_médio
- Exemplo de nomes inválidos: salário médio, 3x, 1ª, @, $

Tipos de variáveis



Strings são diferentes de números

Tipos de variáveis

- O conteúdo de uma variável possui um tipo
- O tipo define a natureza dos dados que a variável armazena
- Os tipos mais comuns são inteiros, números em ponto flutuante e strings (texto)
- Além de poder armazenar números e letras, as variáveis em Python também armazenam valores como True e False

Variáveis numéricas

- Inteiros não possuem casas decimais: 42, -7
- O tipo inteiro em Python é chamado int
- Números em ponto flutuante possuem casa decimal: 1.0, 3.1415, 1234.56
- Note que 1.0, mesmo tendo zero na parte decimal, é um número em ponto flutuante
- O tipo ponto flutuante em Python é chamado float

Exercícios

1. Indique o tipo dos seguintes valores: 5, 5.0, 4.3, -2, 100, 1.333, “10”
2. Experimente digitar type(x) onde x é cada um dos valores acima no Python interativo
3. É possível calcular 2 elevado a um milhão?

Representação de valores numéricos

- Internamente todos os números são representados no sistema binário
- Esse sistema permite apenas os dígitos 0 e 1
- Números em ponto flutuante podem não ter representação exata no sistema binário
- Ex.: Digitando no interpretador 3*0.1 teremos como resposta 0.30000000000000004

Variáveis do tipo lógico

- Podemos armazenar verdadeiro e falso
- A variável se chama lógica ou booleana
- Em Python escrevemos True e False
- Observe que T e F são escritos em maiúsculas

Operadores relacionais

Operador	Operação	Símbolo matemático
==	igual	=
>	maior que	>
<	menor que	<
!=	diferente	≠
>=	maior ou igual	≥
<=	menor ou igual	≤

Observe que o operador de igualdade são dois iguais (==)

Exemplos: operadores relacionais

```
>>> a = 1
>>> b = 5
>>> c = 2
>>> d = 1
>>> a == b
False
>>> b > a
True
>>> a < b
True
>>> a == d
True
>>> b >= a
True
>>> c <= b
True
>>> d!= a
False
>>> d!= b
True
```

a recebe 1
b recebe 5
c recebe 2
d recebe 1
a igual a b?
b maior que a?
a menor que b?
a igual a d?
b maior ou igual a a?
c menor ou igual a b?
d diferente de a?
d diferente de b?

Exemplo importante

- >= ou <= para valores iguais

```
>>> 5 <= 5
True
>>> 5 >= 5
True
```

Exemplo

- Podemos usar operadores relacionais para inicializar variáveis do tipo lógico

```
>>> nota = 8
>>> média = 6
>>> aprovado = nota > média
>>> print (aprovado)
True
```

Operadores Lógicos

- Temos três operadores básicos: not, and e or
- Operador not

```
>>> not True
False
>>> not False
True
```

Operadores Lógicos

- Operador and

```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
```

Operadores Lógicos

- Operador or

```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
```

Expressões Lógicas

- Podemos combinar os operadores lógicos em expressões lógicas
- A ordem de avaliação é not > and > or

Exemplo

- A condição para empréstimo de compra de uma moto é salário maior que R$ 1.000,00 e idade acima de 18 anos. Verificar se o José pode pegar o empréstimo

```
>>> salário = 500.0
>>> idade = 20
>>> salário > 1000 and idade > 18
False
```

Exemplo

- Verifique se um aluno que tirou média para exercícios programa 5.8 e média de provas 7 passou

```
>>> ep = 5.8
>>> p = 7
>>> aprovado = ep >= 6 and p >= 6
>>> print (aprovado)
False
```

Variáveis String

- Armazenam cadeias de caracteres como nomes e textos em geral
- Chamamos cadeias de caracteres uma sequência de símbolos como letras, números, sinais de pontuação, etc
- Para diferenciar seus comandos de uma string utilizamos aspas no início e no final

```
>>> texto = "João e Maria comem pão"
```

Variáveis String

- Note que não há problema de utilizarmos espaços para separar as palavras
- Uma string tem um tamanho associado
- Podemos obter o tamanho através da função embutida len

```
>>> print(len(texto))
22
```

Variáveis String

- Podemos acessar os caracteres da string utilizando um número inteiro para representar sua posição
- Este número é chamado de índice e começamos a contar de zero
- Acessamos o caracter fornecendo o índice entre colchetes ([ ])

```
>>> print(texto[0])
J
```

Variáveis String

- Cuidado: não podemos acessar um índice maior que a quantidade de caracteres da string

```
>>> print(len(texto))
22
>>> print(texto[22])
Traceback (most recent call last):
  File "<pyshell#37>", line 1, in <module>
    print(texto[22])
IndexError: string index out of range
```

Operações com strings

- As operações básicas são fatiamento, concatenação e composição
- O fatiamento permite utilizar parte da string e a concatenação nada mais é do que juntar duas ou mais strings
- A composição é muito utilizada em mensagens que enviamos para a tela e consiste em utilizar strings como modelos onde podemos inserir dados

Concatenação

```
>>> a = "Batatinha "
>>> b = "quando nasce"
>>> print (a + b)
Batatinha quando nasce
>>> print (a * 3)
Batatinha Batatinha Batatinha
```

f-strings

- Juntar várias strings nem sempre é prático
- Podemos usar f-strings. Tudo o que estiver entre chaves {} será substituído, se estiver definido antes. No exemplo .2f significa duas casas decimais.

```
>>> idade = 20
>>> print (f'João tem {idade} anos')
João tem 20 anos
>>> preço = 10.123
>>> print (f'R$ {preço:.2f}')
R$ 10.12
```

Fatiamento

- Fatia do primeiro índice até o anterior do segundo

```
>>> x = "0123456789"
>>> print(x[0:2])
01
>>> print(x[1:2])
1
>>> print(x[2:4])
23
>>> print(x[0:5])
01234
>>> print(x[1:8])
1234567
```

Fatiamento

- Podemos omitir índices, substituindo pelo extremo correspondente e também podemos ter índices negativos: -1 último, -2 penúltimo

```
>>> print(x[:2])
01
>>> print(x[4:])
456789
>>> print(x[4:-1])
45678
>>> print(x[-4:-1])
678
>>> print(x[:])
0123456789
```

Alteração de variáveis com o tempo

- Um programa é executado linha por linha
- Assim, as variáveis podem mudar com o tempo de execução do seu programa

```
>>> dívida = 0
>>> compra = 100
>>> dívida = dívida + compra
>>> compra = 200
>>> dívida = dívida + compra
>>> compra = 300
>>> dívida = dívida + compra
>>> print (dívida)
600
```

Teste de mesa ou simulação

- Entender que o valor das variáveis pode mudar durante a execução de um programa não é tão natural, mas é fundamental para a programação
- Um programa não pode ser lido como um texto, mas cuidadosamente analisado linha a linha
- Você pode treinar com lápis, borracha e papel

Teste de mesa ou simulação

```
>>> dívida = 0
>>> compra = 100
>>> dívida = dívida + compra
>>> compra = 200
>>> dívida = dívida + compra
>>> compra = 300
>>> dívida = dívida + compra
>>> print (dívida)
600
```

dívida	compra	Tela
~~0~~	~~100~~	600
~~100~~	~~200~~	
~~300~~	300	
600		

Não tenha pressa para o teste de mesa

Entrada de Dados

- Até agora nossos programas trabalharam com valores conhecidos
- Vamos começar a pegar os valores durante a execução dos programas e usar mais o modo de edição

```
>>> nome = input('Nome: ')
Nome: Fernando Masanori
>>> print (f'Olá {nome}!')
Olá Fernando Masanori!
```

Conversão da entrada de dados

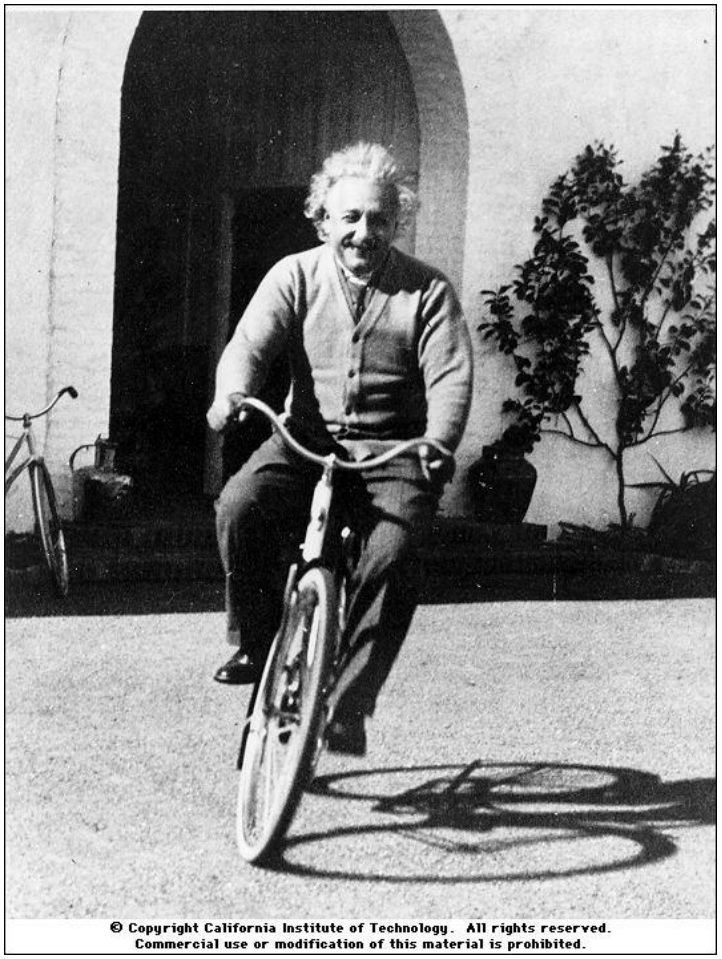
- A função input retorna apenas strings
- Usamos int( ) para converter um valor para inteiro e float( ) para ponto flutuante

```
>>> valor = float(input('Preço: '))
Preço: 1.25
>>> n = int(input('Quantidade: '))
Quantidade: 3
>>> print (f'Total: R$ {n*valor:.2f}')
Total: R$ 3.75
```

Erro comum

- Abrir dois parênteses e fechar apenas um
- O erro vai dar na linha seguinte e fica difícil descobrir
- Sempre que a linha pareça correta, veja a linha imediatamente anterior

Lista de Exercícios



"A vida é como andar de bicicleta. Para manter o equilíbrio, é preciso se manter em movimento". Einstein.