UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Linguagem de Programação I

Prof. Tiago Eugenio de Melo

tmelo@uea.edu.br
www.tiagodemelo.info

Sumário

- Introdução
- Conceitos preliminares
- Introdução
- Variáveis
- Comandos Condicionais

Por que aprender a programar?[1]

- Programar é uma atividade básica de um cientista ou de engenheiro.
- Eu não sou da Computação... por que programar?
  - Porque é legal!
  - Retorno financeiro.
  - É uma disciplina obrigatória.
  - Programação é uma atividade essencial nas mais diversas áreas.

Referência [1]: http://www.ic.unicamp.br/~heiko/teaching/2016s2/mc102/aulaPy01.pdf

Por que aprender a programar?

- Exemplos:
  - Como engenheiro você deverá ser capaz de automatizar algum processo.
  - Como engenheiro você deverá ser capaz de desenvolver novas ferramentas e protótipos.
  - Você poderá enxergar situações onde uma solução computacional pode trazer benefícios.

Por que aprender a programar?

- Eu sou das áreas científicas! Matemática, Física, Química, etc.
  - Como cientistas vocês podem propor uma hipótese e testá-la.
  - Você deverá resolver sistemas complexos de equações que não necessariamente podem ser resolvidos por softwares padrões (como MatLab).
  - Simulações.

O que esperar deste curso?

- Vocês aprenderão o básico para desenvolver programas.
- Utilizaremos a linguagem Python.
- Vocês **NÃO** irão aprender a usar programas neste curso (ex: Office).
- Vocês **VÃO** ter uma boa noção de como criar programas (ex: Office).

O que será necessário?

- Você deverá ter acesso a um computador.
- Para criar um programa, utilizaremos um editor de texto (para escrever o código do programa) e um compilador/interpretador.
- O compilador é o que transforma o código em um programa executável.
- O interpretador é um programa que executa diretamente os comandos da linguagem.
- Se você usa Linux, Mac OS, ou Windows, você poderá utilizar qualquer editor simples.

O que será necessário?

- Para ir bem neste curso:
  - Estude todos os dias.
  - Não deixe acumular assunto.
  - Faça e implemente as listas de exercícios.
  - Finalmente, faça e implemente as listas de exercícios.

Conceitos preliminares

- Lógica de programação
  - Técnica de encadear pensamentos para atingir um determinado objetivo.
- Sequência lógica
  - Passos executados até atingir um objetivo ou a solução de um problema.
- Instruções
  - Conjunto de regras ou normas definidas para realização de algo.

Conceitos preliminares

- Algoritmo
  - É uma sequência finita de passos que levam a execução de uma tarefa.
- Programa
  - São algoritmos escritos em uma linguagem de programação (C, Java, Pascal) e que são interpretados e executados por uma máquina.

Conceitos preliminares

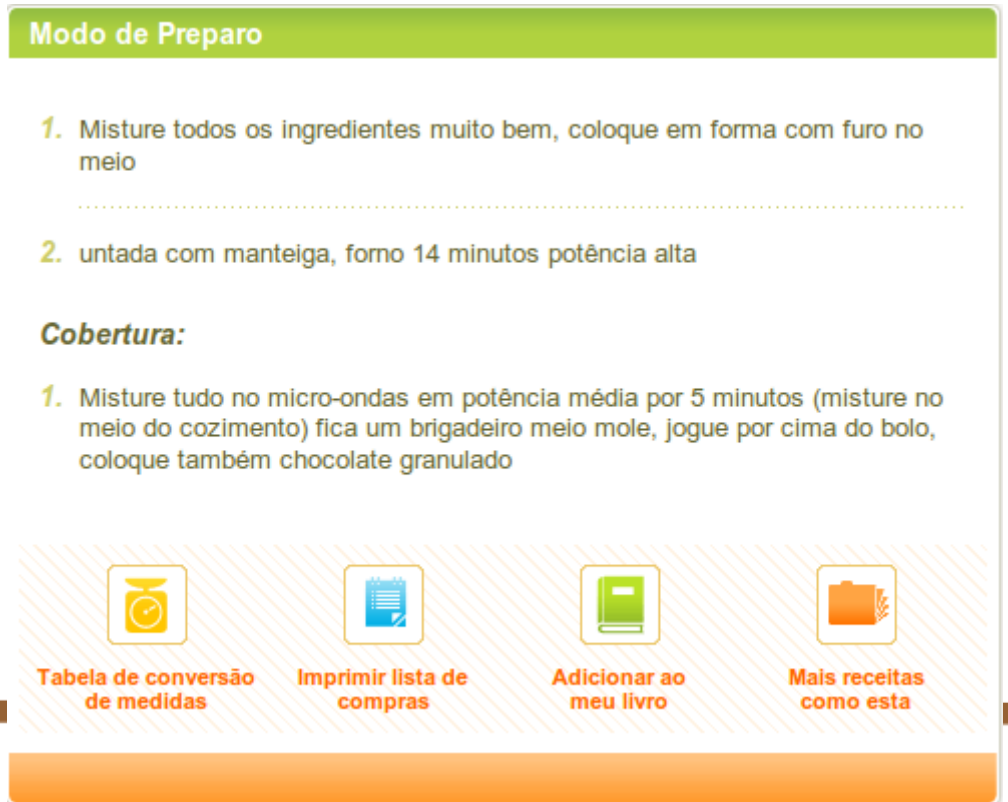
- Algoritmo
  - Conjunto finito de regras que provê uma sequência de operações para resolver um tipo de problema específico.
  - Sequência ordenada e não ambígua de passos que levam à solução de um determinado problema.
  - Processo de cálculo, ou resolução de um grupo de problemas semelhantes, em que se estipulam, com generalidade e sem restrições, as regras formais para a obtenção do resultado ou da solução do problema.

Conceitos preliminares

- Características dos algoritmos
  - São finitos.
  - Não devem ser ambíguos.
  - Capacidade de receber dados de entrada do mundo exterior.
  - Podem gerar informações de saída para o mundo exterior.

Conceitos preliminares

- Formas de representação
  - Descrição narrativa
    - Faz uso do idioma para descrever algoritmos.
    - Exemplo: receita de bolo.

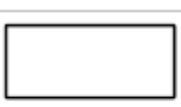
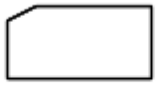
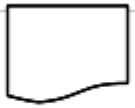
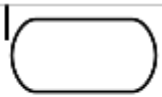


Conceitos preliminares

- Formas de representação
  - Descrição narrativa
    - Vantagens:
      - O idioma é conhecido por todos.
    - Desvantagens:
      - Imprecisão.
      - Pouca confiabilidade (imprecisão).
      - Extensão desnecessária.

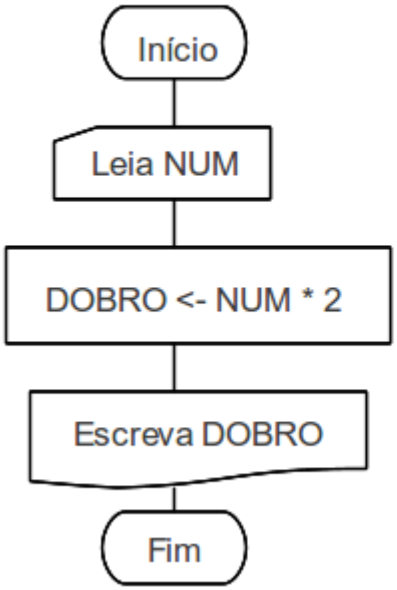
Conceitos preliminares

- Formas de representação
  - Fluxograma
    - Utilização de símbolos para representar algoritmos.

				
Cálculo	Decisão	Entrada	Saída	Início/Fim

Conceitos preliminares

- Formas de representação
  - Fluxograma

EXEMPLO	EXPLICAÇÃO
	Início do algoritmo Entrada do número Cálculo do dobro do número Apresentação do resultado Fim do algoritmo

Conceitos preliminares

- Formas de representação
  - Fluxograma
    - Vantagens
      - Uso de ferramentas conhecidas.
      - Figuras dizem mais que palavras.
      - Padrão mundial.
    - Desvantagens
      - Pouca atenção aos dados.
      - Complica-se à medida que o algoritmo cresce.

Conceitos preliminares

- Formas de representação
  - Linguagem algorítmica
    - Consiste na definição de uma pseudolinguagem de programação, cujos comandos são escritos em português para representar os algoritmos.
    - Exemplo:

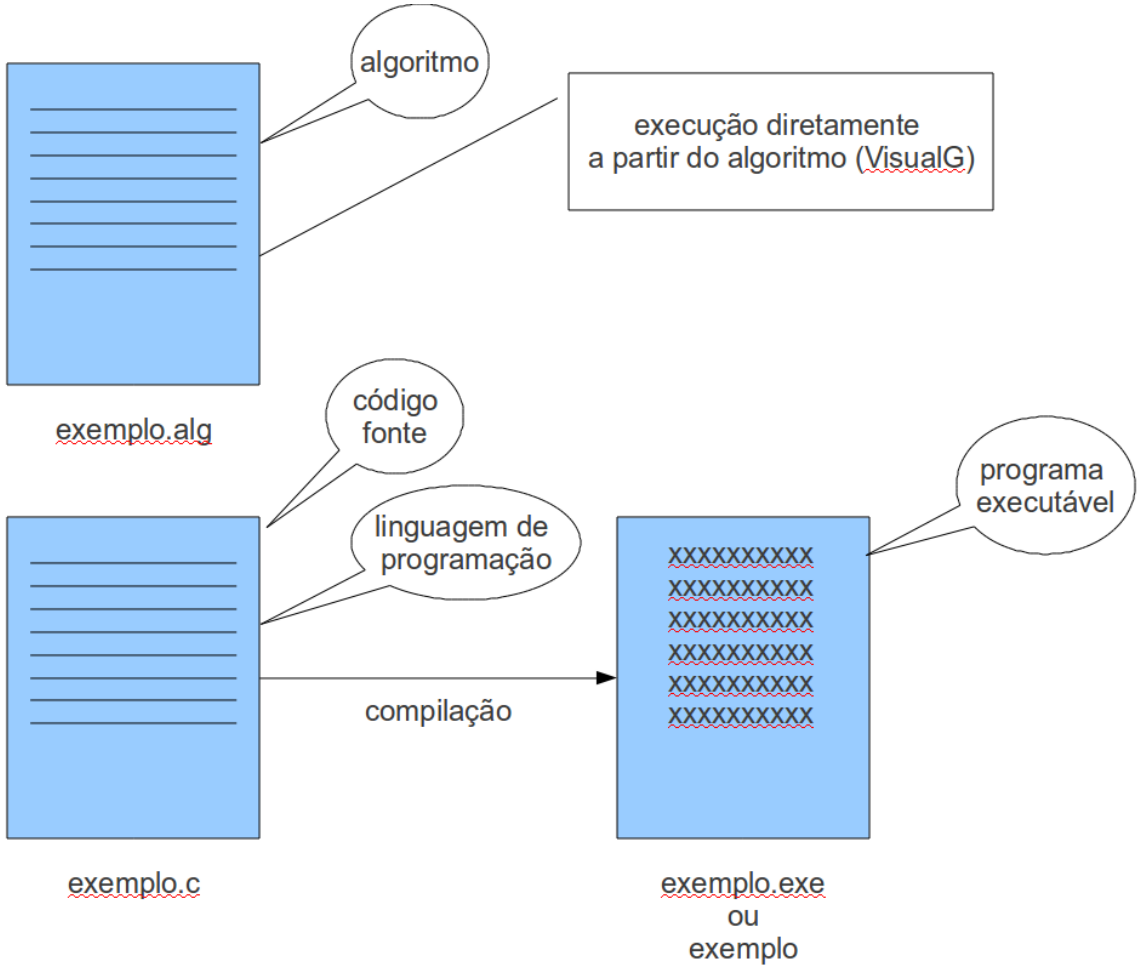
```
algoritmo "Algoritmo conversor"
var
   entrada, saida: inteiro
inicio
      leia (entrada)
      saida <- entrada * 60
      escreva (entrada, " minutos correspondem a ", saida, " segundos.")
fimalgoritmo
```

Conceitos preliminares

- Formas de representação
  - Linguagem algorítmica
    - Vantagens
      - Independência física da solução.
      - Usa o português como base.
      - Passagem quase imediata do algoritmo para a linguagem de programação.
    - Desvantagens
      - Exige a definição de uma linguagem não real para o trabalho.
      - Não padronizado.

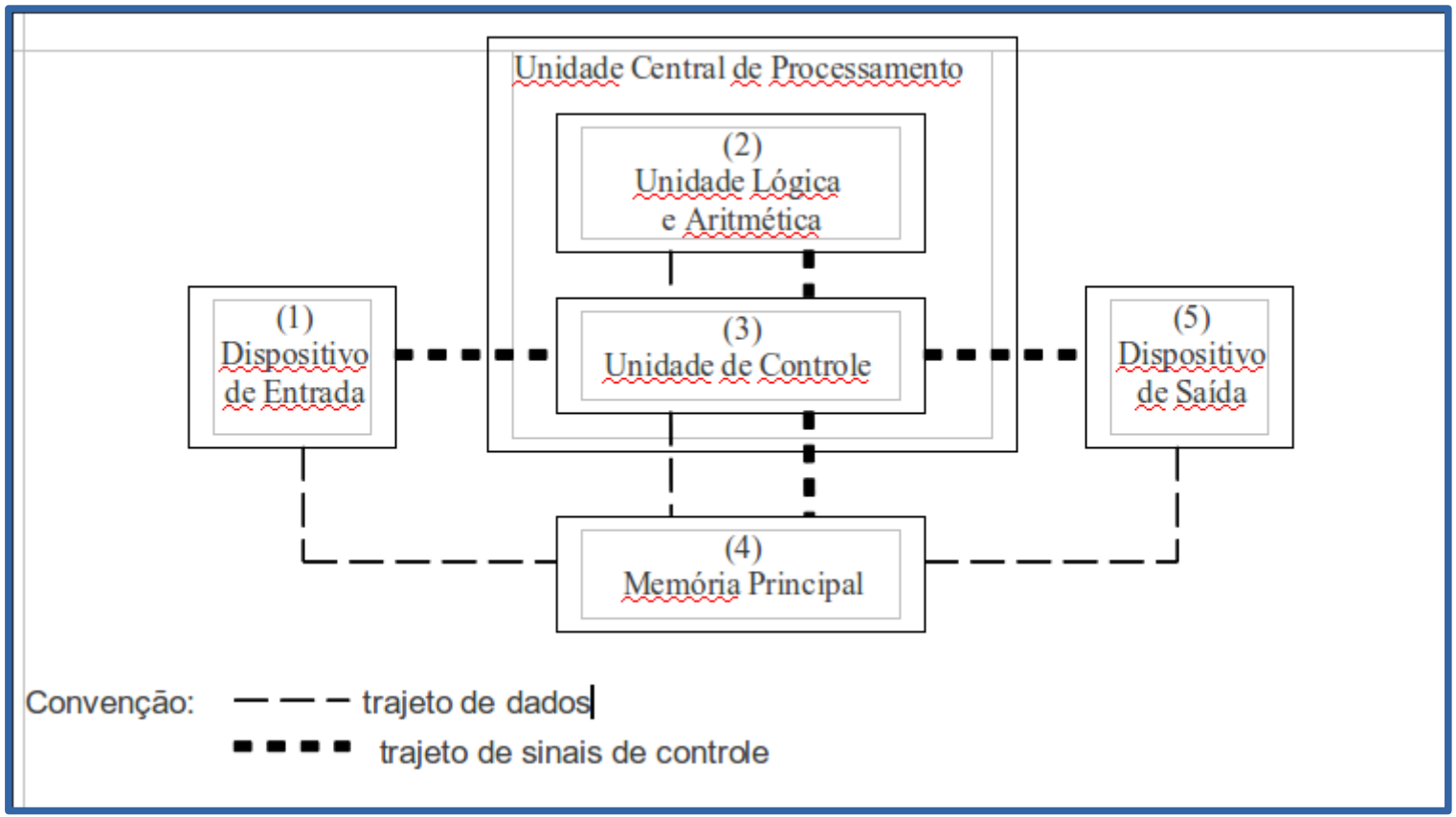
Conceitos preliminares

- Processo de execução



Conceitos preliminares

- Arquitetura de um computador padrão



Introdução

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		99.7
3. Java		97.5
4. C		96.7
5. C#		89.4
6. PHP		84.9
7. R		82.9
8. JavaScript		82.6
9. Go		76.4
10. Assembly		74.1

Introdução

- Python
  - Simples e eficaz.
  - Primeira linguagem de programação.
  - Criada por Guido van Rossum em 1991.
  - Interpretada.
- Ambiente
  - IDE (*Integrated Development Environment*)
  - IDLE (*Integrated Development and Learning Environment*)
- Versões
  - 2.7.X e 3.x.

Introdução

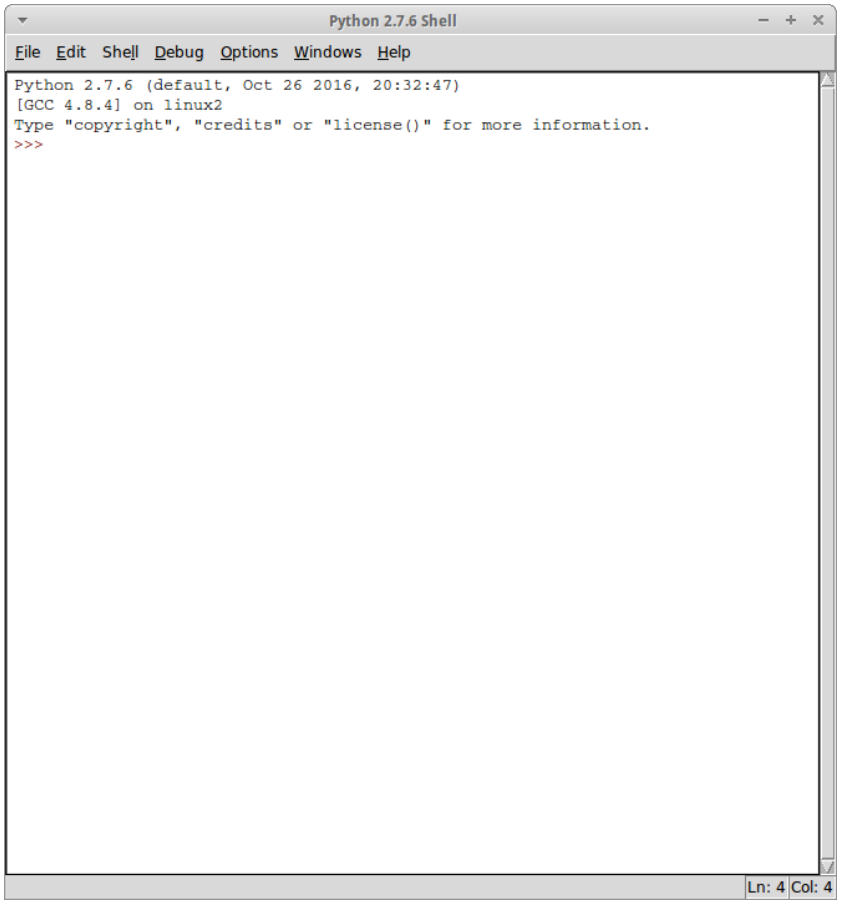
- Sistema operacional
  - Python pode ser executado nos principais sistemas operacionais.
  - Neste curso usarei Linux.

Introdução

- Python
  - Fácil de aprender.
  - Fácil leitura e compreensão.
  - Fácil manutenção.
  - Multiplataforma.
  - Modo interativo.
  - Extensível.
  - Acesso aos principais banco de dados.
  - GUI (*Graphical User Interface*).
  - Escalável.
  - Multiparadigma.
  - Script e compilada.

Introdução

- IDLE
  - Code complete.
  - Editor de scripts.

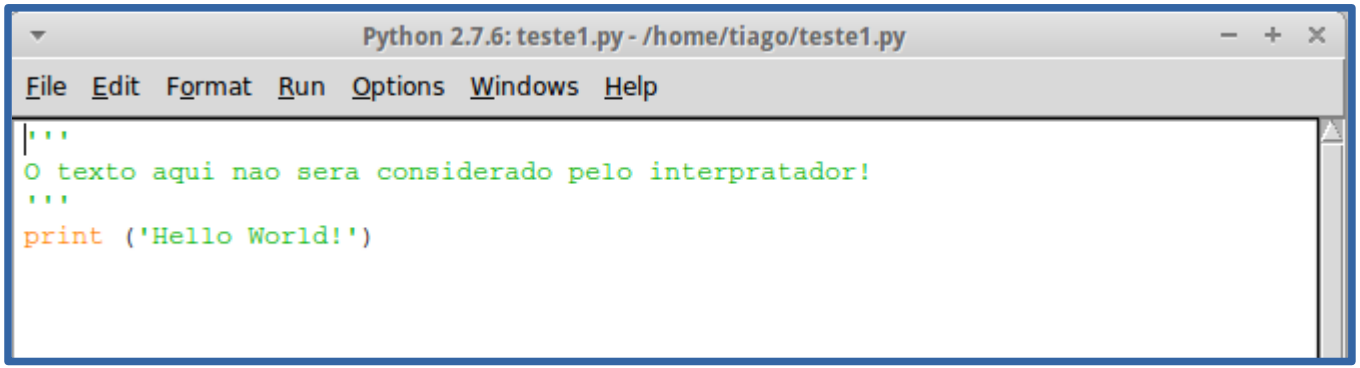


Introdução

- Extensão *.py
- Imprimindo na tela
  - `print (' ')`
- Exemplo:
  - `print ('Hello World!')`

Introdução

- Comentários
  - O objetivo é adicionar descrições a partes específicas do código.
  - Notação *inline*
    - *# comentario*
  - Notação *multilines*



Introdução

- Indentação
  - É o recuo do texto em relação a sua margem.
- Blocos

Variáveis

- É um espaço de memória que reservarmos para armazenar valores temporários que estão sendo processados ou manipulados.
- Toda variável possui um tipo.
- O tipo é inferido conforme a informação inicial que atribuirmos para a variável.
- Uma variável pode ter o seu valor alterado a qualquer momento.
- Não há limite na quantidade de variáveis em um programa.
- É necessário sempre inicializá-la antes de fazer uso delas.

Variáveis

- Características
  - Nome
  - Tipo
  - Espaço
  - Valor

Variáveis

- Características das variáveis
  - Nome
    - Forma de referência aos valores.
    - Não é permitido o uso de caracteres especiais.
    - Pode conter números, desde que não seja o primeiro caracter.
    - Palavras reservadas não podem ser usadas.

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

Variáveis

- Características das variáveis
  - Tipo
    - Inferido pela máquina virtual (VM).
    - Tipos primitivos
      - String
        - Conjunto de caracteres disposto numa determinada ordem.
      - Inteiro
      - Real
      - Lógico (booleano)
        - `True` ou `False`
    - Conversão (coerção)

Variáveis

- Exemplo de conversão

Python 2.7.6 Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.6 (default, Oct 26 2016, 20:32:47)
[GCC 4.8.4] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> x = 10
>>> type (x)
<type 'int'>
>>> y = "20"
>>> type (y)
<type 'str'>
>>> x + y

Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    x + y
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> x + (int) (y)
30
>>> (str) (x) + y
'1020'
>>>
```

Variáveis

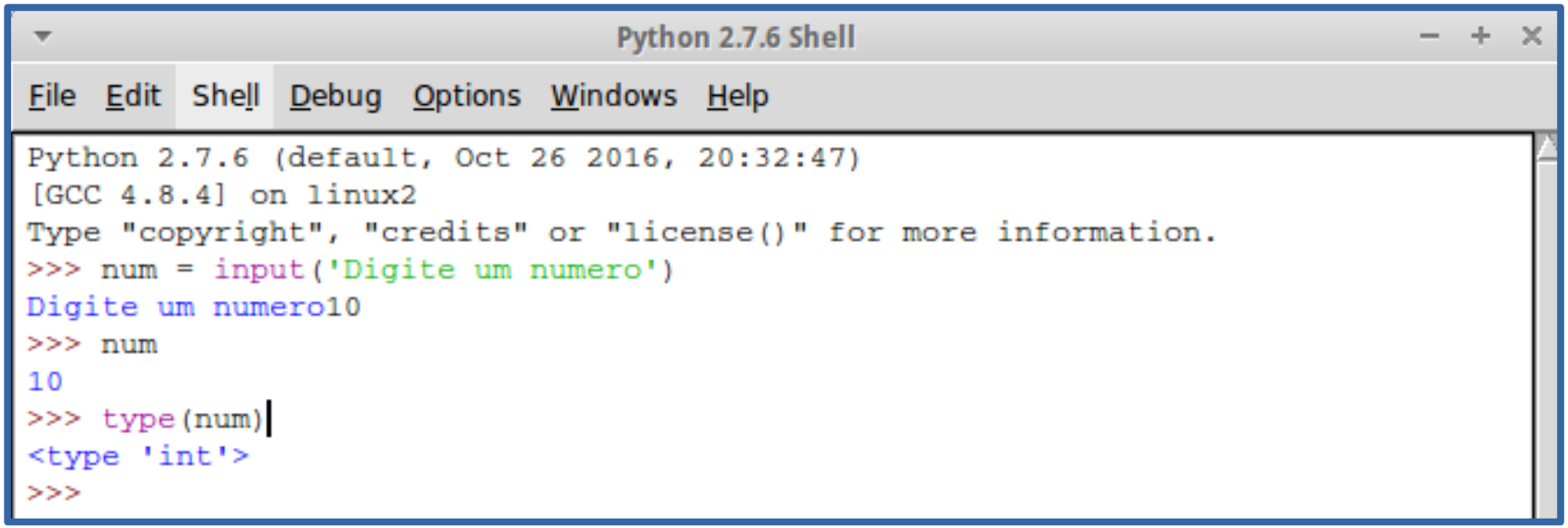
- Características das variáveis
  - Espaço
    - São armazenadas na memória RAM do computador.
  - Valor

Atribuição de Valores

- É a passagem de informação a determinada variável.
- Operador de atribuição em Python: `=`
- Exemplo:
  - `x = 10`
  - A parte do lado esquerdo do operador de atribuição sempre receberá o valor no lado direito do operador.

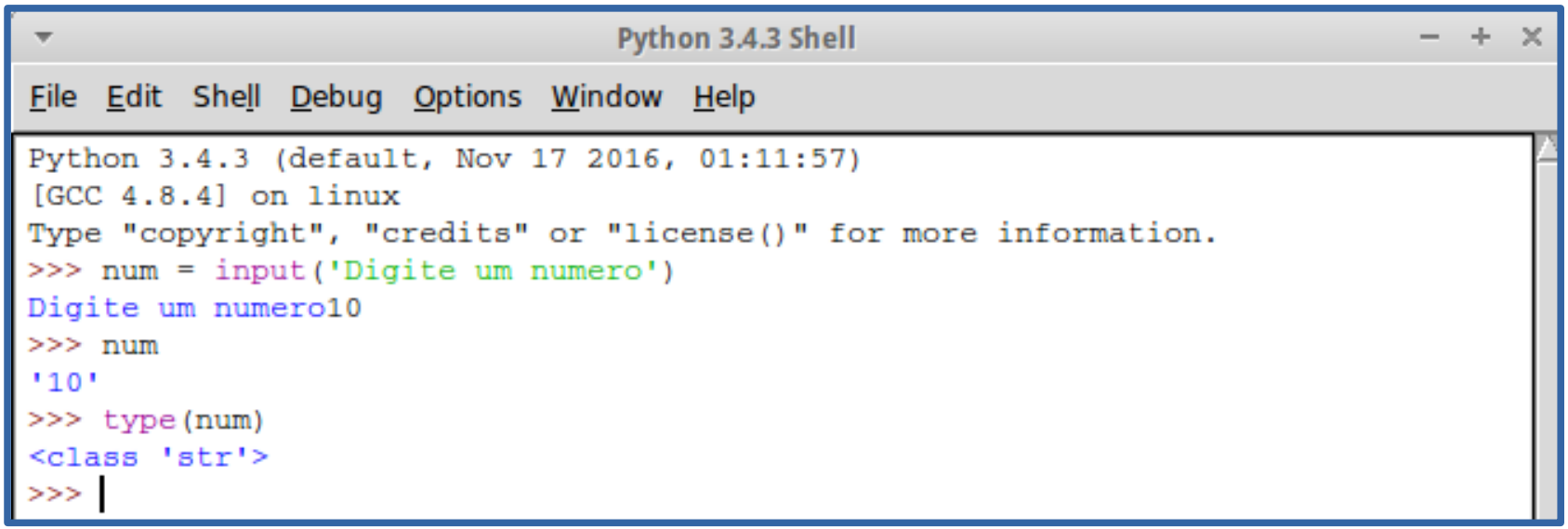
Entrada de dados

- Forma de iteração com os usuários.
- Comando `input()`
- Exemplo (Python 2.7):



Entrada de dados

- Comando `input()`
- Exemplo (Python 3):



Operadores Aritméticos

- Operações elementares

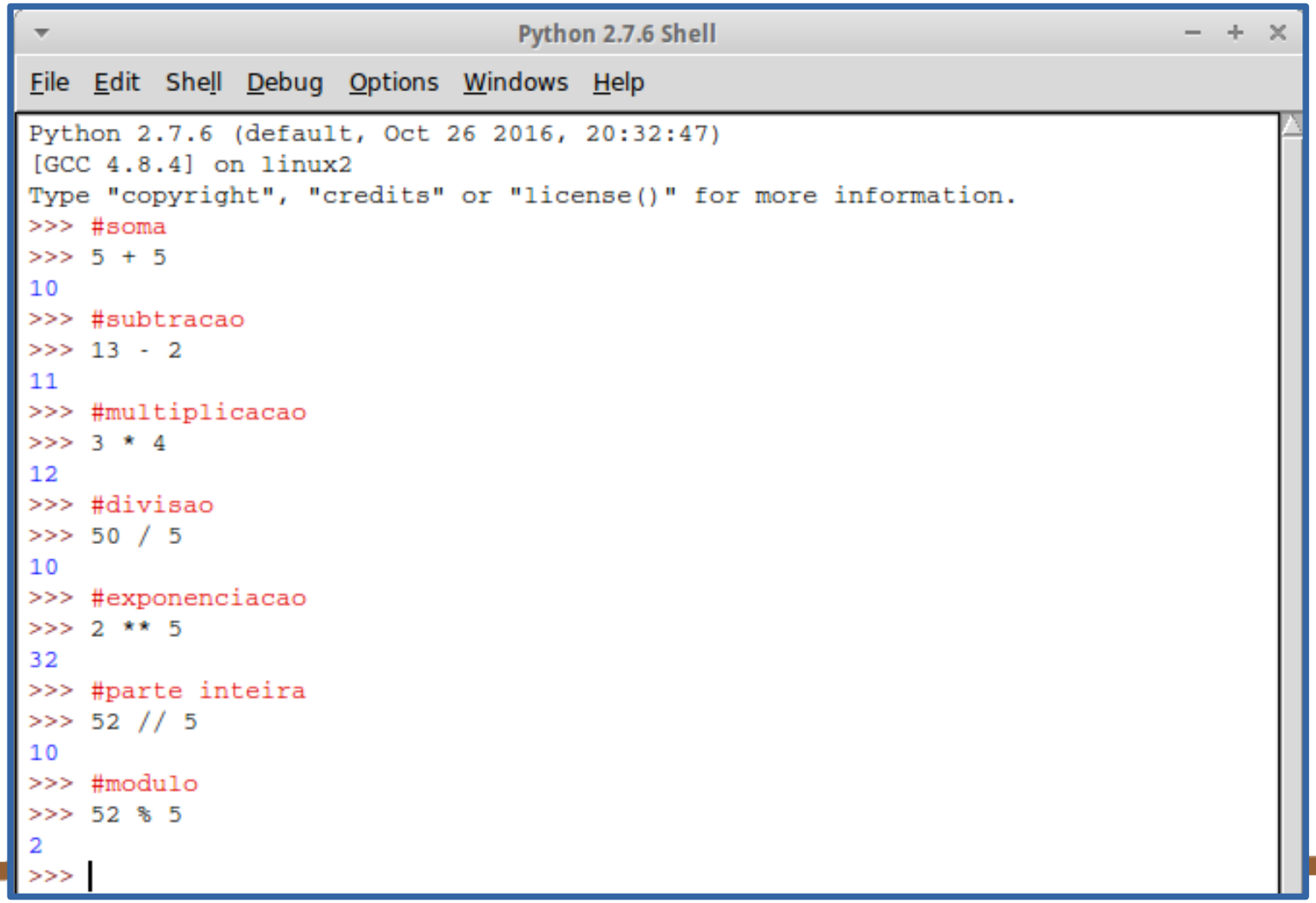
Operação	Operador
adição	+
subtração	-
multiplicação	*
divisão	/

- Operações avançadas

Operação	Operador
exponenciação	**
parte inteira	//
módulo	%

Operadores Aritméticos

- Exemplos



Operadores Aritméticos

- Como obter valores reais de uma divisão?

```
>>> ================================ RESTART ================================
>>> 23 / 5
4
>>> 23 / 5.0
4.6
>>> 23.0 / 5
4.6
>>> 23 / float(5)
4.6
>>>
```

Operadores Aritméticos

- Exponenciação / Radiciação

```
>>> ================================ RESTART ================================
>>> 5 ** 2
25
>>> 5 ** 0.5
2.23606797749979
>>>
```

Operadores Relacionais

- O resultado é sempre do tipo lógico (`True` ou `False`)
- Operadores:

Descrição	Operador
Maior que	>
Menor que	<
Igual a	==
Maior ou igual a	>=
Menor ou igual a	<=

Operadores Relacionais

- Exemplos

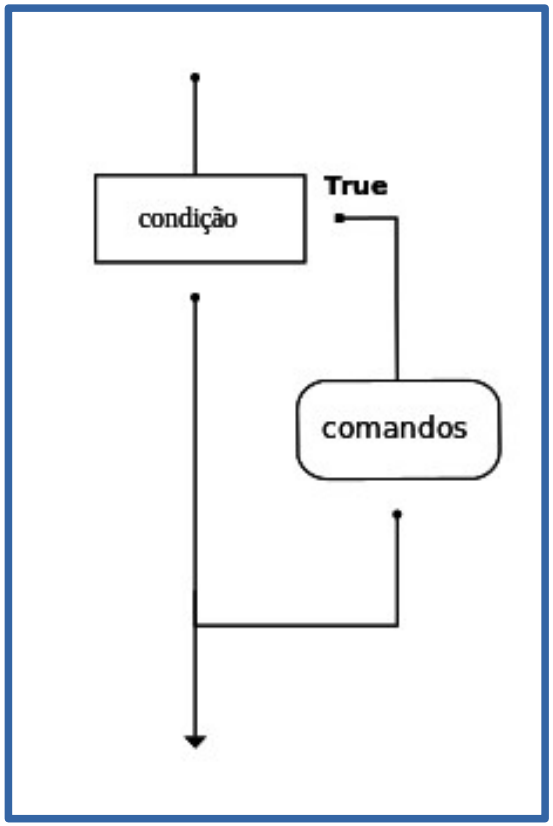
```
>>> ================================ RESTART ================================
>>> 10 == 4
False
>>> 3 > 4
False
>>> 3 >= 3
True
>>> 5 != 4
True
>>>
```

Comandos Condicionais

- Alteração do fluxo sequencial dos programas.
- Criação de blocos de comandos.

Comandos Condicionais

- Comando `if ( )`
- Ideia



Comandos Condicionais

- Exemplos

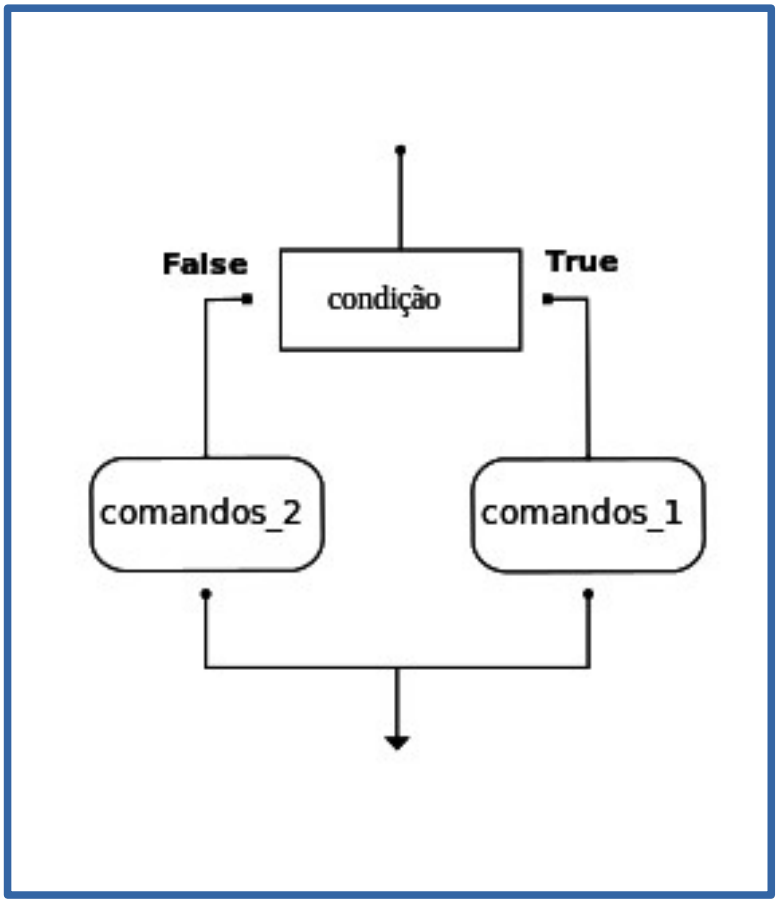
```
>>> num = input ('Digite um numero')
Digite um numero 10
>>> if (num > 5):
        print ('O numero digitado e maior que 5.')


O numero digitado e maior que 5.
>>>
```

```
1 x = 10
2 if x < 0:
3     print("O numero negativo ",  x, " nao e' valido aqui.")
4 print("Isto e' sempre impresso.")
5
6
```

Comandos Condicionais

- Comando `if ( ) - else ( )`
- Ideia geral



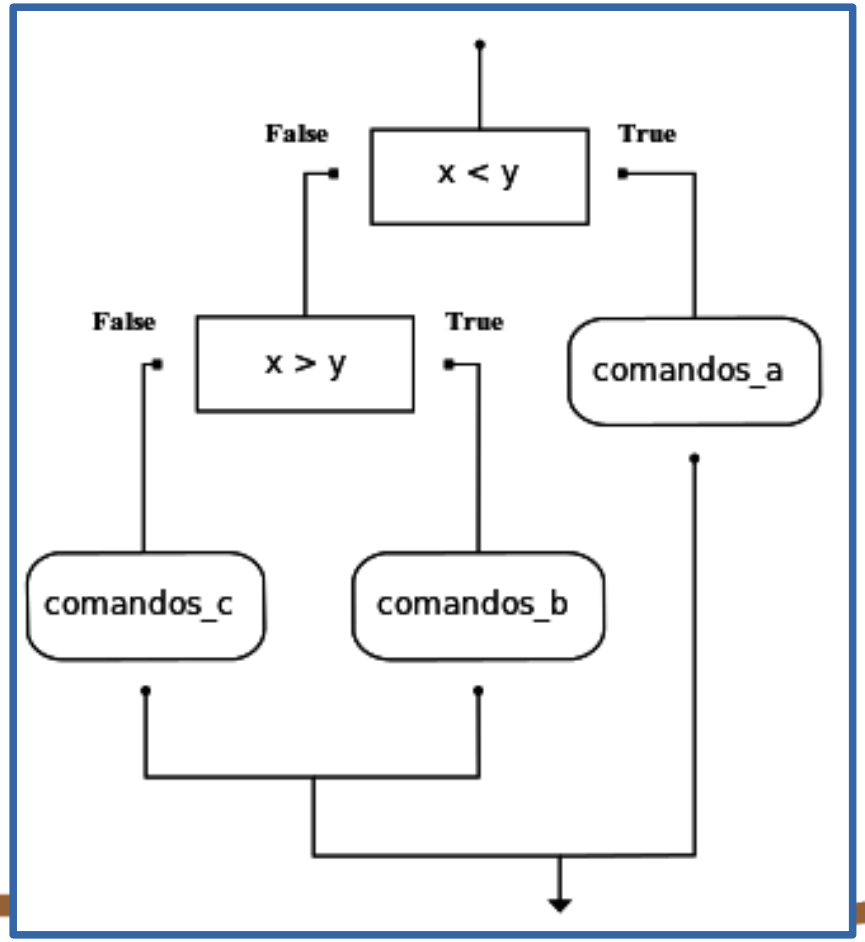
Comandos Condicionais

- Exemplos

```
>>> x = 11
>>> if x % 2 == 0:
...     print 'Par'
... else:
...     print 'Impar'
...
Impar
```

Comandos Condicionais

- Os comandos condicionais podem ser aninhados em vários níveis



Comandos Condicionais

- Exemplo

```
if x < y:
    print("x e' menor do que y.")
else:
    if x > y:
        print("x e' maior do que y.")
    else:
        print("x e y devem ser iguais.")
```

Depuração de Código

- É a técnica de programação em que é possível manipular a execução de cada linha e verificar os valores das variáveis.
- A depuração é comumente usada na detecção de erros e para entender o funcionamento de um programa.

Operadores Lógicos

- Conectivos
  - Conjunção E: and
  - Disjunção OU: or
  - Negativo: not
- TABELA

A	**B**	**A `and` B**	**A `or` B**	**`not` A**
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Operadores Lógicos

- Exemplo

```
>>> if (nota >= 6) and (frequencia >= 0.75):
...     print 'Aprovado'
... else:
...     print 'Reprovado'
```

Referências

- http://excript.com/python/iteracao-python.html