INTRODUÇÃO AO PYTHON

ime INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

Objetivo

- Apresentar o histórico e as características básicas da linguagem Python
- Criar e executar pequenos programas em Python
- Fornecer ao alunos uma visão geral da linguagem de programação Python

Histórico

- Foi concebida no final de 1989 pelo holandês *Guido Van Rossum*
  - O código só foi publicado em 1991
  - A versão 1.0 foi lançada em 1994
- Python é interpretada
- Permite as metodologias de programação:
  - Estruturada
  - Orientada a objetos
  - Funcional

Principais usos da linguagem Python

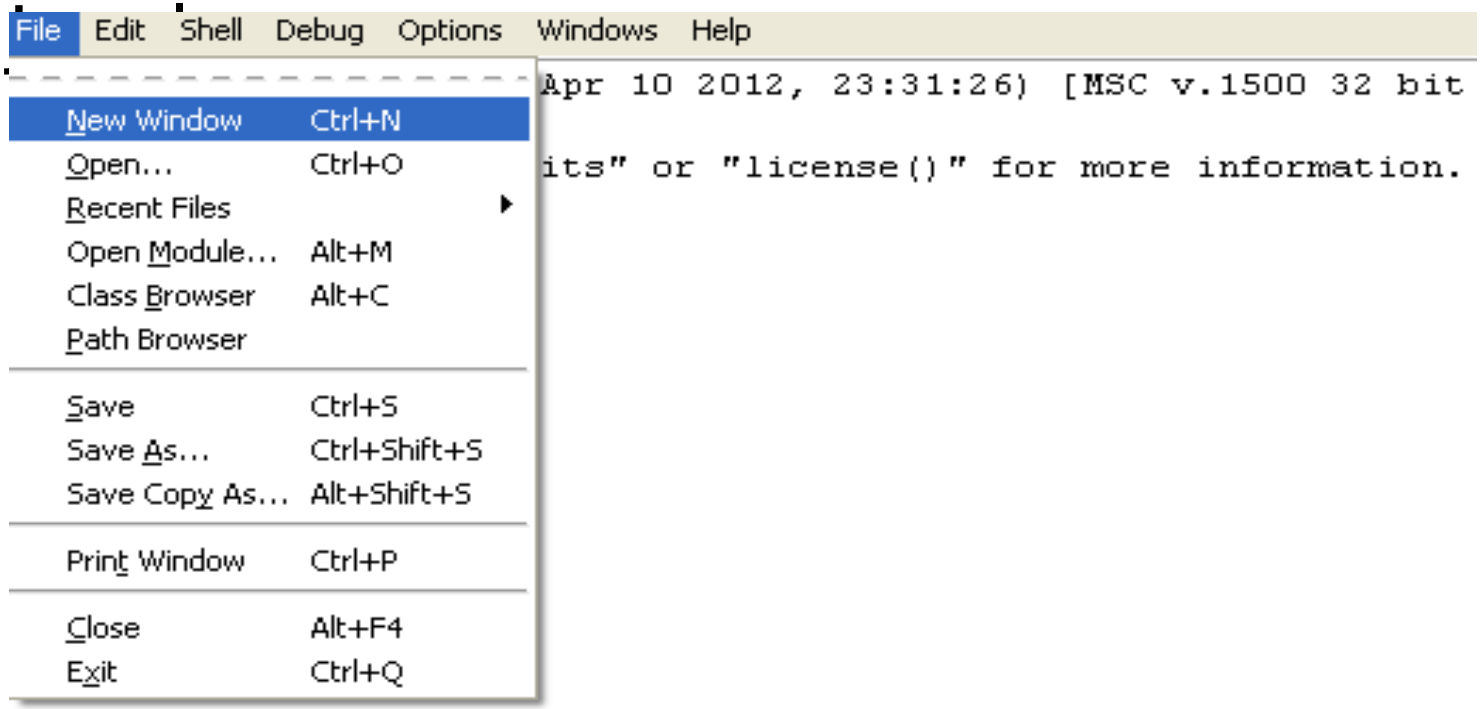
- Quarta linguagem mais popular do mundo (http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages).
- É a linguagem escolhida como a mais adequada para o ensino introdutório de computação na maior parte das principais universidades dos EUA (http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext)
- Uma das linguagens mais propícia para a construção de protótipos de sistemas.
- Muito utilizada para Machine Learning, Finanças, Computação Científica, Programação Web, etc.. Enfim, há uma vasta gama de aplicações utilizando Python atualmente.
- Linguagem escolhida para IPD!!!!!

Versões do Python

- As principais versões são (https://www.python.org/downloads/)
  - 2.7.x (última versão 2.7.12)
  - 3.5.x (última versão 3.5.12)
- Vamos usar a versão 2.7.x por ser um pouco mais simples
- Se quiser saber quando usar cada uma delas acesse a página:
  - http://wiki.python.org/moin/Python2orPython3

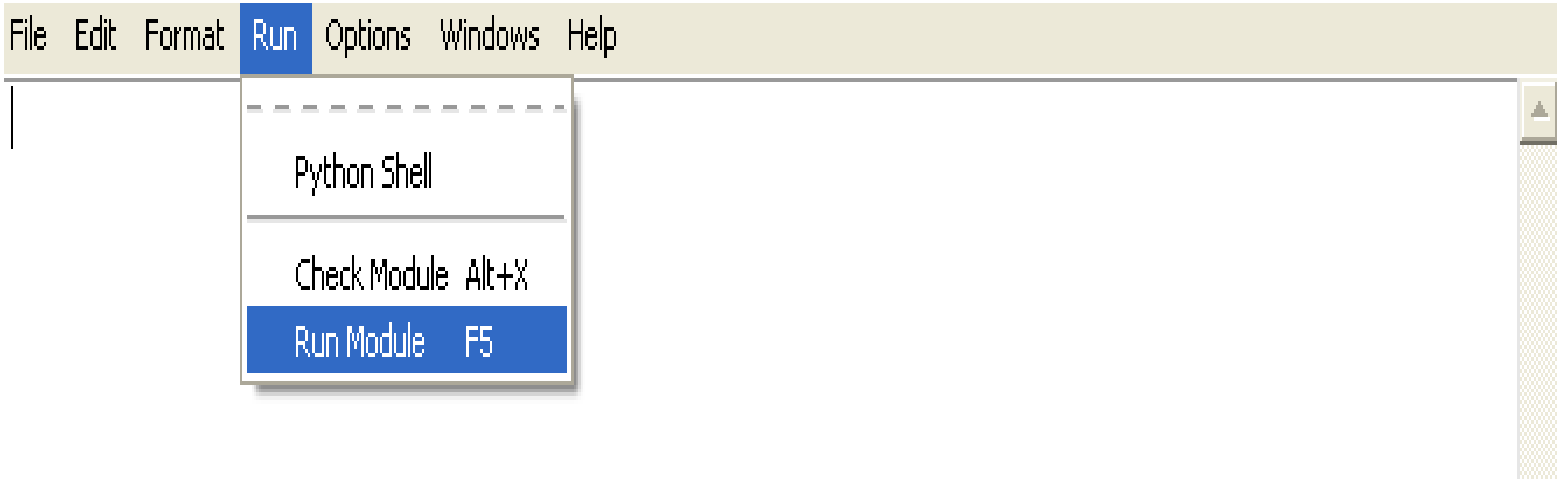
Criando e executando um programa em Python

- Clique no icone do **IDLE** e inicie uma nova



Criando e executando um programa em Python

- Digite os comandos do programa
- Salve este arquivo com o nome:
  - NOME_PROGRAMA.py
- Executar o programa pressionando F5 ou selecionando



Criando e executando um programa em Python

- Digite os comandos do programa
- Salve este arquivo com o nome:
  - NOME_PROGRAMA.py
- Executar o programa pressionando F5 ou selecionando

Características

- Programas, em geral, são mais curtos
- Sintaxe clara e concisa
- Possui várias bibliotecas prontas
- Código livre
- Case-sensitive
- Tipagem forte e dinâmica
- Possui modo Interativo e Programado

Modos

- O Python pode ser usado no modo INTERATIVO ou PROGRAMADO
- No modo INTERATIVO os comandos digitados seguidos de [ENTER] serão executados imediatamente possibilitando que o Python seja usado como se fosse uma calculadora.
- No modo PROGRAMADO os comandos serão: digitados, salvos em um arquivo e depois compilados e executados.

Identificadores

- Um identificador é qualquer nome aceito pela linguagem
  - Variáveis
  - Palavras reservadas
- Python é Case Sensitive
  - Diferencia maiúscula de minúscula
  - AB, Ab, aB e ab são variáveis DIFERENTES
- Podem ter qualquer tamanho
- São formadas por letras, números e sublinhado
  - Iniciam SEMPRE por uma letra

Variáveis

- Uma variável é um identificador
- Podem ter qualquer quantidade de caracteres
  - Ex: notaAlunoUERJPrimeiroSemestre2013
- São usadas para armazenar os dados do programa
- NÃO PRECISAM SER DECLARADAS
  - Tipagem dinâmica
- A partir da atribuição de um valor elas passam a ser do TIPO do valor atribuído
  - Tipagem forte

Tipos Simples

- Inteiros – int
- Inteiros longos – long
- Reais – float
- Números complexos – complex
- Literais – str
- Lógicos – bool

Indentação

- Os blocos de comandos são delimitados em Python pela indentação (**recuo**, derivado da palavra em inglês *indentation*, também grafado nas formas **indentação** e **endentação)**
  - **A INDENTAÇÃO É OBRIGATÓRIA**
- O número de espaços no recuo é variável, mas todas as instruções dentro do bloco têm de ser recuado na mesma quantidade.

Comentários

- São muito importantes para dar clareza aos programas
- São ignorados pelo compilador
  - Ou seja, é como se não existissem
- São precedidos pelo caractere # (tralha ou hash)
  - Ex: # testando comentário
  - 2 + 2

  - 2 + 2 # ignora o que vem depois

Operadores Aritméticos

- + SOMAR
- - SUBTRAIR
- * MULTIPLICAÇÃO
- / DIVISÃO
- // DIVISÃO INTEIRA
- % RESTO
- ** EXPONENCIAÇÃO

Operadores Aritméticos

Testar no PYTHON

- 4 + 3 - 2
- 2 * 3
- 7 / 2
- 7 // 2
- 7 % 2
- 2 ** 3
- a += 1 # a = a + 1

Precedência de operadores

1. ()
2. **
3. *, /, //, %
4. +, -

Em caso de mesma hierarquia resolve-se da esquerda para direita

```
> 4 + (3**2)//3 - 4%3 * 5 - 2
```

Operadores relacionais

- ==, !=, >, >=, <, <=
- Uma relação retorna sempre um valor lógico ***False*** ou ***True***

Testar no PYTHON

› a = 2; b= 3

› a == b

› a != b

› a > b

Operadores lógicos

- *and*, *or* e *not*
- Uma operação lógica retorna sempre um valor lógico ***False*** ou ***True***

Testar no PYTHON

› a = 2; b= 3

› a == b *and* a != b

› a == b *or* a != b

› *not* (a == b *and* a != b)

Entrada / Saída (dados numéricos)

- Os comandos de entrada e saída são usados para enviar dados para o programa e mostrar os resultados do programa
- SINTAXE DO COMANDO DE ENTRADA:
  VARIAVEL = input('mensagem')

Entrada de dados numéricos. Testar no Python

```
>>> a = input('Entre com numero')
>>> a
```

Entrada / Saída (strings)

- SINTAXE DO COMANDO DE ENTRADA:
  VARIAVEL = raw_input(‘mensagem’)

Entrada de dados alfabéticos. Testar no Python

>>> a = raw_input(‘Entre com numero’)

>>> a

Se o usuário digitar 1, qual o valor de a?

‘1’

Entrada / Saída

- SINTAXE DO COMANDO DE SAÍDA:
  print VARIAVEL
  print VARIAVEL, CONSTANTE, ...

Testar no Python

```
>>> a = 5
>>> print a
>>> print ‘num: ‘, a
```

Programa em Python

- Basta digitar os comandos em linhas distintas
- Não esquecer a indentação quando houver comandos dentro de outros comandos
- NÃO se declara variáveis em PYTHON

Exercício

- Fazer um programa para ler 2 números e imprimir a soma

```
a = input('N1: ')
b = input('N2: ')
c = a + b
print 'soma: ', c
```

Exercício

- Fazer um programa para ler 4 números e imprimir a média

```
a = input('N1: ')
b = input('N2: ')
c = input('N3: ')
d = input('N4: ')
result = (a+b+c+d)/float(4)
print 'soma: ', result
```

Exercício

- Fazer um algoritmo para ler o valor do tempo em segundos e imprimir em hora, minuto e segundos
  - Ex: 4000s = 1h 6min 40s

```
totalSeg = input('tempo em segundos: ')
hora     = totalSeg // 3600
minuto   = (totalSeg % 3600)  //  60
segundo = (totalSeg % 3600)  %  60
print hora,':',minuto,':',segundo
```