SCC 124 - Introdução à Programação para Engenharias

Python

Professor: André C. P. L. F. de Carvalho, ICMC-USP
Pos-doutorando: Isvani Frias-Blanco
Monitor: Henrique Bonini de Britto Menezes

Aula de hoje

- Introdução
- Por que Python?
- História da linguagem Python
- Programas na linguagem Python
- Erros de programação na linguagem Python
- Empresas que usam Python

Habilidades

- Reconhecer os componentes de um programa Python típico
- Entender os tipos de dados da linguagem Python
  - E como eles podem ser utilizados para armazenar informações em um programa
- Conhecer as principais bibliotecas da linguagem Python

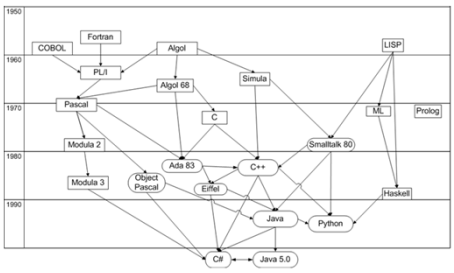
Habilidades

- Reconhecer os comandos condicionais e os repetitivos
  - E ser capaz de utilizá-los em programas simples
- Aprender a decompor um programa em funções individuais
  - E entender como essas funções operam
- Ser capaz de escrever programas que combinem os conceitos aprendidos

Linguagens de alto nível

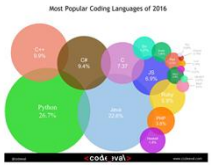
- Maior parte da computação envolvia o cálculo de fórmulas
  - Fórmulas eram traduzidas para a linguagem de máquina
    - Por que não escrever programas parecidos com as fórmulas a serem aplicadas?
  - 1950: IBM produz a versão inicial da linguagem Fortran (**for**mula **tran**slation)
    - Primeira linguagem de alto nível

Evolução das linguagens de alto nível

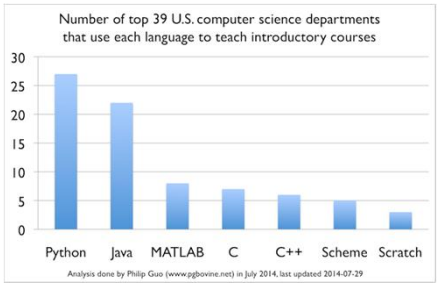


Linguagens de alto nível

- Várias outras linguagens de alto nível foram criadas
  - Uma das mais bem sucedidas foi a linguagem Python



Python



Por que Python?

- Código aberto
- Legibilidade
- Portabilidade
- Expansibilidade
- Concisão

Por que Python?

- Legibilidade
  - Programas mais curtos
  - Código mais próximo de um texto em linguagem natural (inglês)
- Portabilidade
  - Mesmo programa pode ser executado em diferentes computadores e sistemas operacionais

Por que Python?

- Expansão de uso
  - Estímulo ao desenvolvimento de bibliotecas
  - Facilidade de escrita de programas mais complexos
  - Concisão
  - Conjunto poderoso de comandos
  - Gera códigos menores e mais elegantes
  - Aumenta produtividade de programadores

História de Python

- 1980s: desenvolvimento da ling. ABC
  - Para ensino e prototipagem
  - Projeto mal sucedido
- 1990: desenvolvimento de Python para substituir ABC e escrever o SO Amoeba
  - Criador de Python, Guido van Rossum, trabalhou na linguagem ABC
- 1991: Python é disponibilizada

História de Python

- 1994: Primeiro Python workshop no NIST
- 2000: lançamento da versão 2.0
  - Utilizando conceito de listas
- 2001: Formação da Python Software Foundation (PSF)
- 2008: lançamento da versão 3.0
  - Python 3000" or "Py3K
  - Conserta problemas e remove redundâncias

Python

- Inspiração para o nome
  - Grupo humorístico Monty Python
  - Também conhecidos como The Pythons
    - Criadores do Monty Python's Flying Circus
      - Série de comédia da TV britânica

Python

- Mitologia grega
  - Serpente, dragão da terra de Delphi
  - Vivia nas cavernas do Monte Parnassus, considerado o centro da terra
  - Morta a flechadas por Apollo

Python

- Serpente carnívora, capaz de engolir mamíferos de grande porte
  - Semelhante a Jibóia
- Mede de 0,5 a 10 metros
- Uma das espécies, *Python reticulatus* , é considerada a maior serpente do mundo

Linguagem Python

- Linguagem orientada a objeto
  - Paradigma de linguagem de programação baseado no conceito de objetos
  - Objeto é uma estrutura de dados que inclui:
    - Dados
      - Armazenam valores a serem manipulados
    - Funções
      - Operações que manipulam dados
  - Python possui vários objetos pré-definidos

Blocos de comandos

- Programa em Python é formado por blocos de comandos
  - Identificados usando indentação
    - Avanço do texto em relação à margem esquerda
  - Outras linguagens usam *begin-end* ou o par { }
  - Indentação deve ser consistente no programa
    - Caso contrário o interpretador não funcionará corretamente
    - Cuidado ao misturar TABs com espaços

Blocos de comandos

- Blocos de comandos em Python são detectados automaticamente pela indentação de linhas
  - Todos os comandos indentados a uma mesma distância da margem esquerda
  - Bloco termina em:
    - Linha com menor indentação
    - Final do arquivo
- Blocos encaixados
  - Comandos apresentam indentação mais à direita que bloco externo

```
def fib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a
```


Escrita de programas

- Python funciona em modo interativo
  - Programador interage diretamente com o interpretador
  - Boa forma de começar
- Também funciona em modo script
  - Código pode ser salvo em um arquivo, chamado script
  - Interpretador pode executar o script

Linguagem Python

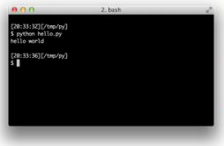
- A principal implementação de Pythn (Cpython) é escrita na linguagem C
  - Compila um programa em Python para um bytecode intermediário
    - Forma de conjunto de instrução projetado para a execução eficiente de um interpretador
    - Códigos numéricos compactos

Modo interativo

- Python interpreter prompt
  - Comandos podem ser digitados diretamente na linha de comando

Prompt

```
$ (Comando em Python)
(resultado produzido)
```



```
$ print ("Hello, world")
Hello, world
```

$ ou >>>

Mesmo exemplo em C

```
#include <stdio.h>
main()
{
    printf ("Hello, world");
}
Hello, world
```


Mesmo exemplo em Java

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Os três

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

```
#include <stdio.h>
main()
{
    printf ("Hello, world");
}
Hello, world
```

```
$ print ("Hello, world")
Hello, world
```


Modo script

- Escrever programa na linha de comando cada vez que for executá-lo
  - Impraticável para programas grandes
    - Dificulta salvar, recuperar e alterar
- Programa pode ser gravado em (lido de) um arquivo
  - Permite escrever e salvar programas para serem usados e modificados mais tarde
  - Scripts em Python terminam com .py

Escrita de programas

- Editor convencional
  - Não destaca as diferentes partes de um programa
    - Ex.: comentário, comando, ...
  - Não checa erros de sintaxe
  - Não suporta indentação de linhas de código
- Melhor usar editor adequado para escrita de programas

Software PyCharm

- Software que integra editor com interpretador
- Programa pode ser gravado em arquivos
  - e lido de arquivo

Tradução

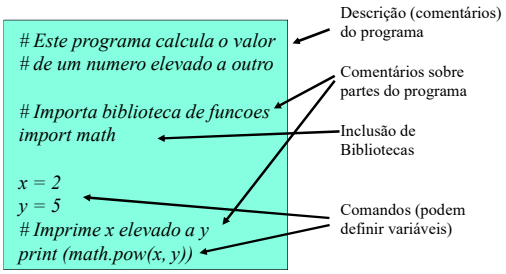
- Python é uma linguagem interpretada
  - Reduz tempo gasto no desenvolvimento de programas
    - Poder ser usado interativamente
      - Facilita experimentar novas características da linguagem

Interpretador Python
- Checa arquivo
- Executa cada linha na sua ordem
- Termina na última linha

Padrões de programação

- Nomes de variáveis com significado
- Código estruturado
- Código adequadamente indentado
  - Essencial em Python
- Boa documentação
  - Nome do programador, contato, datas
  - Descrição geral do programa
  - Bons comentários

Estrutura de um programa Python



Comentários

- Ignorados pelo interpretador, fornecem informações importantes
  - Para outros programadores
  - Para quem escreveu o programa
- Formato: # isto é um comentário
  - Pode ter mais de uma linha

```
# comentario
```

```
#
# Comentario mais
# longo
#
```

```
################
# Comentario mais
# rebuscado
################
```


Comentários

- Mais úteis quando destacam características do código que não são óbvias
  - Suposições assumidas pelo programador
  - Decisões importantes
  - Detalhes importantes
  - Tarefas pendentes
  - Problemas existentes no programa
- Código informa como e comentários informam porque

Comentários

- Destacam características do programa que não são óbvias
  - Comentário redundante
    - V = 5 # atribui 5 a v
  - Comentário útil
    - V = 5 # velocidade em metros / segundo

Bibliotecas

- Conjunto de funções escritas por outros que executam operações úteis
  - Ao serem incluídas, suas funções são disponibilizadas
- Ex.: *import nome-da-biblioteca*
  - Ex. *import math*
  - Várias já vêm com o interpretador Python
  - Várias são disponibilizadas por outros programadores

Biblioteca *math*

- Possui várias funções

```
Valor absoluto: a = abs(-7.5)
Arco seno: x = asin(0.5) #returns in rads
Arredondamento (topo): print(ceil(4.2))
Cosseno: a = cos(x) #x in rads
Graus: a = degrees(asin(0.5)) #a=30
Exp: y = exp(x) #y=e^x
Truncamento: a = floor(a+0.5)
Log: x = log(y); #Natural Log
  x = log(y,5); #Base-5 log
Log Base 10: x = log10(y)
Max: mx = max(1, 7, 3, 4) #7
  mx = max(arr) #max value in array
Min: mn = min(3, 0, -1, x) #min value
Potência: x = pow(y,3) #x=y^3
Radianos: a = cos(radians(60)) #a=0.5
Random:
  x = random() #Inteiro aleatório no intervalo [0.0, 1.0)
  y = randint(a,b) #Inteiro aleatório no intervalo [a, b]
Arredondamento: print round(3.793,1; #3.8 - rounded to 1 decimal
  a = round(3.793,0) #a=4.0
Seno: a = sin(1.57) #in rads
Raiz quadrada: x = sqrt(10) #3.16...
Tangente: print tan(3.14)# #in rads
```

Biblioteca *math*

```
# importar funções da biblioteca math
import math

print ("Raizes de 0 to 9 formatados como float com 4 decimais:“)
for valor in range(10) :
    raiz = math.sqrt(valor)
    print ("sqrt(%d) = %.4f" % (valor, raiz))
```


Biblioteca *math*

```
Raizes de 0 to 9 formatados como float
com 4 decimais:
sqrt(0) = 0.0000
sqrt(1) = 1.0000
sqrt(2) = 1.4142
sqrt(3) = 1.7321
sqrt(4) = 2.0000
sqrt(5) = 2.2361
sqrt(6) = 2.4495
sqrt(7) = 2.6458
sqrt(8) = 2.8284
sqrt(9) = 3.0000
```


Programa em Python

```
# Este programa calcula o valor
# de um numero elevado a outro

# Importa biblioteca de funcoes
import math

x = 2
y = 5
# Imprime x elevado a y
print (math.pow(x, y))
```

```
# Este programa calcula o valor
# de um numero elevado a outro

x = 2
y = 5
# Imprime x elevado a y
print (x**y)
```


Comandos

- Especificam ações
- Comandos utilizados no programa apresentado:
  - Ex1. Atribuição
    - *x = 2*
    - *y = 5*
  - Ex2. Saída
    - *print (math.pow(x, y))*

Programa em Python

```
# cria uma funcao que converte Fahrenheit para Celsius
def convertFahrenheit2Celsius(fahrenheit):
    celsius = 0.555 * (fahrenheit - 32)
    return celsius

# usa a funcao (deve ser criada antes de chamada)
print (“Tabela Fahrenheit para Celsius:”)
# varia de -40 ate 220 com incrementos de 10
for tf in range(-40, 220, 10):
    print ("%5.1f F = %5.1f C" % ( tf, convertFahrenheit2Celsius(tf) ))
```


Estrutura de um Programa Python

```
Tabela Fahrenheit para Celsius:
-40.0 F = -40.0 C
-30.0 F = -34.4 C
-20.0 F = -28.9 C
-10.0 F = -23.3 C
  0.0 F = -17.8 C
 10.0 F = -12.2 C
 20.0 F =  -6.7 C
 30.0 F =  -1.1 C
 40.0 F =   4.4 C
 50.0 F =  10.0 C
 60.0 F =  15.5 C
 70.0 F =  21.1 C
 80.0 F =  26.6 C
 90.0 F =  32.2 C
100.0 F =  37.7 C
110.0 F =  43.3 C
120.0 F =  48.8 C
130.0 F =  54.4 C
140.0 F =  59.9 C
150.0 F =  65.5 C
160.0 F =  71.0 C
170.0 F =  76.6 C
180.0 F =  82.1 C
190.0 F =  87.7 C
200.0 F =  93.2 C
210.0 F =  98.8 C

Process finished with exit code 0
```

Mesmo programa em C

```
/* Imprime tabela de conversao Fahrenheit-Celsius */
# include <stdio.h>

main (){
  int fahr, celsius;
  int inicio, fim, incremento;

  inicio = -40;          /* limite inferior da tabela */
  fim = 220;             /* limite superior */
  incremento = 10;
  fahr = inicio;
 Printf ("Tabela Fahrenheit para Celsius: ");
  while (fahr <= fim){
     celsius = 5*(fahr -32)/9;
     printf ("%d \ F t %d C\n", fahr, celsius);
     fahr = fahr + incremento;
  }
}
```


Erros na programação

- Muitas vezes, os programadores cometem erros ao escrever seus programas
  - Digitação do programa
  - Lógica do programa para resolver o problema
- Um programa em Python pode apresentar 3 tipos de erros
  - Erros sintáticos
  - Erros de execução
  - Erros semânticos

Erros sintáticos

- Código não obedece às regras de sintaxe da linguagem – não interpreta
  - Sintaxe de uma linguagem é a sua "gramática"
- Erros são indicados por meio de mensagens de erros resumidas na tela do computador
  - Mensagens de erros mais detalhadas podem ser encontradas em manuais ou livros da linguagem
  - Feitas as correções necessárias, a interpretação pode ser reinicializada

Erros sintáticos

```
values = [1, 3, 5 0.1]
total = sum(values)
X = 3)
print "resultado:
print (final)
```

Ausência de ,

Fecha parênteses sem abre parênteses

Trocar : por " Falta parênteses

Não existe a variável *final*

Erros de execução

- Código obedece às regras de linguagem
  - Programa é interpretado
  - Mas ao ser executado, código executa operações não permitidas
    - Exemplos.: dividir por zero, raiz quadrado de número negativo, laço (loop) infinito
  - Podem produzir mensagem de erro
  - Também chamados de *exceptions*
- Mais difíceis de descobrir e entender que erros sintáticos

Erros de execução

```
import math
value = -3
raiz= math.sqrt(value)
print  ("resultado: ", raiz)
```

Raiz quadrada de um número negativo

Erros semânticos

- Semântica = significado
- Erros no projeto lógico do programa
  - Forma utilizada para resolver o problema
- Provoca erros indesejáveis no programa
  - Mensagens de erro não são apresentadas
  - Sua correção exige mudanças na concepção do programa
  - Ex.: trocar igual (==) por atribuição (=)
- São os erros de detecção mais difícil

Erros semânticos

- Programa que deve retornar raiz quadrada de 3:

```
import math
value = 3
raiz= math.sqrt(value)/2
print  ("resultado: ", raiz)
```


Debugar

- Procurar bugs (erros) em um programa que apresenta problemas
  - Boa prática para aprendizado:
    - Explorar o que ocorre ao cometer erros
  - Procurar erro, corrigir, testar
  - Habilidade útil não apenas em programação

Python para engenharia

- SciPy ecossistema baseado em Python- de software open-source para:
  - Ciência
  - Engenharia
  - Matemática
  - Possui vários pacotes
- Implementação: Anaconda (gratuito)

Python para engenharia

- Pacotes do SciPy
  - NumPy 
    - Operações com matrizes
  - SciPy library (biblioteca) 
    - Biblioteca básica para computação científica
  - Matplotlib 
    - Gráficos 2D

Python para engenharia

- Pacotes do SciPy (continuação)
  - Ipython 
    - Comandos adicionais para usar no console
  - Sympy 
    - Operações para matemática simbólica
  - Pandas 
    - Análise e estruturas de dados

Quem usa Python

- Airbus
- AstraZeneca
- Boeing
- CIA
- *Disney*
- *Finnish Civil Aviation Administration (FCAA)*
- Google
- Honeywell
- IBM
- NASA
- *New York Stock exchange*
- Nokia
- Philips
- RedHat
- Yahoo!

Conclusão

- Linguagens de programação
- Linguagem Python
  - Por que Python?
  - História de Python
  - Criação de programas em Python
- Exemplo de programa em Python
- Erros de programação

Perguntas