Características da Linguagem e introdução à prática

Anteriormente...

- Já vimos grande parte do aspecto histórico e contextual que envolve a linguagem python. Iremos agora passar para a parte “prática” de nosso minicurso.

- Neste primeiro momento, vamos nos familiarizar com as principais características da linguagem e em seguida, aprender a configurar nossa máquina para podermos programar em python.

Características da Linguagem

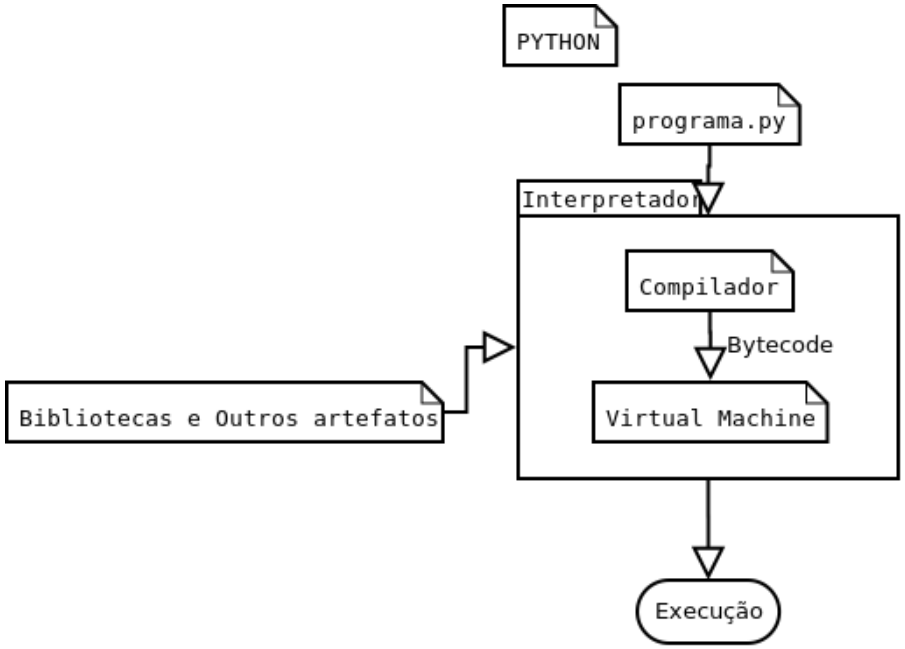
A linguagem Python possui diversas características quanto à sua funcionalidade, dentre elas:

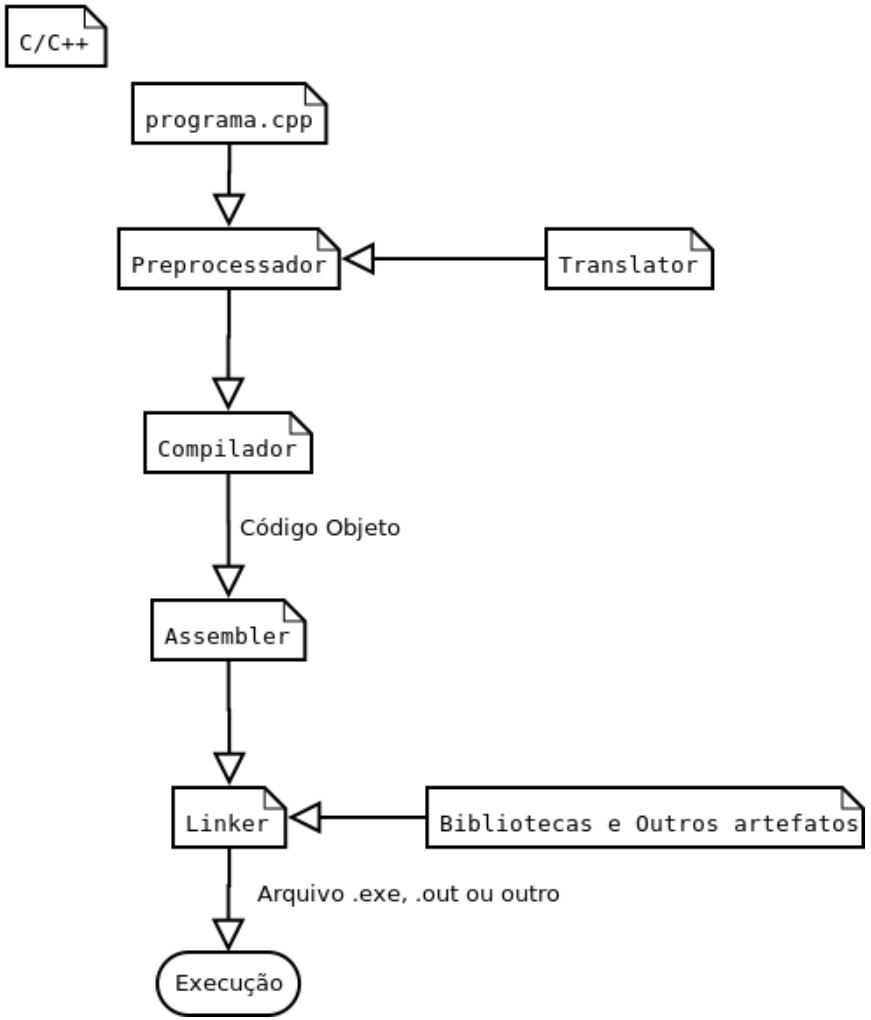
- Sintaxe simples;
- Simplicidade em aprendizado, leitura, compreensão e manutenção dos códigos;
- Implementação de estruturas de dados facilitada;
- Multiparadigma (Já apresentado);
- Multiplataforma;
- Dentre outros...

Implementações de Python

- Por implementação, não queremos dizer “como o programa será criado”, mas sim, “como a linguagem irá rodar (transcrever e executar) seu código”. Como paradigmas de implementação, temos duas linhas principais de trabalho: As linguagens compiladas e as linguagens interpretadas.
- Já dissemos que python é uma linguagem interpretada, mas o que isso significa, na prática?

- Em uma linguagem interpretada, seu código será traduzido em um outro tipo de código (no caso de python, algo conhecido como bytecode) e então, será analisado por uma ferramenta conhecida como interpretador. Python opera com o esquema de máquina virtual, ou seja, uma “simulação” de um ambiente que contém os meios para se executar o bytecode.
- Algumas vantagens disso são a fácil adaptação do código aos mais diferentes “tipos de ambiente” e a facilidade em se encontrar “erros”(pois os mesmos serão observáveis em tempo de execução). Como desvantagem, podemos citar a demasiada carga de tempo requerida, quando comparado ao processo de compilação. A seguir, um esquema comparativo entre como python opera X como C/C++ opera.


PYTHON
programa.py
Interpretador
Compilador
Bytecode
Virtual Machine
Bibliotecas e Outros artefatos
Execução


C/C++
programa.cpp
Preprocessador
Translator
Compilador
Código Objeto
Assembler
Linker
Bibliotecas e Outros artefatos
Arquivo .exe, .out ou outro
Execução

Quanto às implementações da linguagem por sí, python possui diversas, para os mais diferentes tipos de fim.

A implementação considerada “padrão” (implementação de referência) é conhecida como CPython, e é também a que é distribuída por padrão e utilizada como base pela comunidade de desenvolvedores. A mesma é, como o nome diz, implementada na linguagem C.

Dentre as outras implementações, vale citar a Jython (criada em Java e, consequentemente, com mais afinidade a tal linguagem) e IronPython (Mais afinidade às linguagens .NET).

Requisitos

- Python, quanto aos requisitos de sistema, não se mostra uma linguagem muito exigente, quando comparada a outras linguagens de alto nível.
- Vale lembrar, no entanto, que tudo depende da aplicação que estará sendo desenvolvida. Um sistema com altos requerimentos de hardware (para processamento gráfico, análise de dados, etc.) irá requerer, logicamente, uma máquina com bom “condicionamento” para ser desenvolvido.
- Sendo assim, eis alguns requisitos mínimos, recomendados pela própria comunidade de desenvolvedores python:

Hardware

- Processador com capacidade igual ou superior à 800Mhz;
- Pelo menos 500Mb de memória RAM;
- Pelo menos 300Mb livres em disco (Para instalador, editor e os arquivos referentes à própria linguagem após a instalação);
- Dispositivos de saída (Monitor) e entrada (Teclado, Mouse);

Estes requisitos servem apenas para ter uma base. Podemos ver que python não é tão exigente quanto a essa questão do Hardware.

Software

Por ser uma linguagem distribuída de forma livre, python conta com diversas adaptações e ferramentas específicas, fazendo com que possa ser desenvolvida na maior parte dos sistemas operacionais modernos. Dentre eles:

- Linux
- MacOS
- Windows
- Sistemas Mobile (Android, IOS, Sistemas de alguns celulares Nokia, etc.)
- Dentre outros...

- O interpretador de python também é necessário que esteja instalado em sua máquina, bem como os arquivos fonte, necessários à linguagem.
- Por conveniência, tudo isto pode ser obtido através do site:

https://www.python.org/

Além dos próprios arquivos fonte de python instalados e do compilador, também é interessante que tenhamos um editor de texto para operarmos, como por exemplo, o Geany(Linux) ou o Notepad++ (Windows), tal como qualquer outro.

Futuramente, estaremos aprendendo como fazer uso de uma IDE para desenvolvimento em python.

Versões do python

Uma coisa que pode ser um tanto quanto confusa a primeira vista são o esquema de versões do python disponíveis para desenvolvimento, sendo elas:

- Python 2.X, com sua versão final liberada em 2007 e com suporte “marcado para término” em 2020;
- Python 3.X, versão atualmente cotada, liberada em 2008;

Ambas permitem o desenvolvimento eficiente e são consideradas esforços “oficiais” da comunidade. Então, qual é a “correta” de se utilizar?

Existem sim, diferenças entre as duas, como por exemplo, o tipo de suporte a unicode, aspectos da linguagem em si, funções com retornos diferentes e etc. Como iria tomar muito tempo explicar todas as diferenças entre as duas, no slide extra temos um link com essas informações.

Fato é que, até o momento, as duas linguagens permanecem incompatíveis entre si.

O bom senso nos diz que, dado que a versão 3.X se trata da mais recente, a mesma seria indicada para o desenvolvimento de projetos em geral. No entanto, é fato que algumas bibliotecas, até o momento, não foram “portadas” para a versão 3.X. Outra coisa a ser avaliada é quando trabalhamos com software que exigem ferramentas inerentes a uma versão específica do python.

Resumindo, você como programador, será responsável por decidir qual será a versão do python com a qual irá trabalhar em determinado projeto, visto os requisitos específicos de tal projeto.

Por questão de praticidade, nos fixaremos em apenas uma das duas versões.

Nosso caso

- Estarei utilizando um computador “comum”.
- Sistema operacional: Lubuntu 16.04 LTS
- Também irei utilizar o editor de texto Geany em nossos exemplos. Futuramente, irei demonstrar outras IDEs.
- Sobre a versão do python, utilizaremos a versão

Instalação

Os métodos de instalação para os sistemas operacionais Windows 7 (e similares) e Lubuntu 16.04 (e similares) estão disponíveis em um slide à parte.

Após realizarmos tais procedimentos, podemos seguir nosso minicurso.

Um primeiro programa

Como nosso primeiro programa em python, vamos começar com o famoso “Hello World!”

Uma coisa importante sobre este primeiro programa é que o mesmo servirá para testarmos nosso ambiente de desenvolvimento e verificar se tudo está ok.

Crie uma pasta com o nome “Minicurso Python – Seu Nome Aqui”, onde ficarão armazenados os códigos que você irá criar durante este minicurso.

Hello World!

- Dentro da pasta, crie um arquivo comum, coloque o nome “hello.py” .
- Abra o arquivo e digite o seguinte:

```
print “Hello World!”
```

- Salve o arquivo.

- Deverá ter sido criado algo parecido com:

```
1 print "Hello World!"
```

Executando seu código em python

- Para executar

```
antonio@misao:~/Documentos/Ificina/Educacional/Minicursos/MinicursoPythonBásico/Slides/Exemplos/HelloWorld$ #Dentro da sua pasta do minicurso
antonio@misao:~/Documentos/Ificina/Educacional/Minicursos/MinicursoPythonBásico/Slides/Exemplos/HelloWorld$ python hello.py
```

O comando “python” nos permitirá executar nosso código pelo terminal/cmd. O mesmo opera da mesma forma em sistemas baseados em Unix (Linux, Mac, etc).

Analise: O que acabamos de fazer?

Parabéns! Você acabou de criar seu primeiro código em python. Já pode fazer um Sistema Operacional completo. Ou não.

Brincadeiras à parte, vamos analisar o que acabou de ser feito.

Primeiramente, você criou um arquivo com nome “hello.py”. Essa extensão “.py” é a extensão padrão para códigos fonte na linguagem python. Todo arquivo de código que manipularemos neste minicurso terá esta extensão.

- Em seguida você escreveu, em uma linha de código, o programa.
- Você utilizou a keyword “print”, que chama uma função específica do python (veremos isso mais a fundo depois), que basicamente, imprimiu na tela o que foi escrito em seguida, “Hello World!”.
- A instrução “print” tem a função de escrever o que lhe foi passado na tela. Ela trabalha de forma similar as outras instruções de impressão de outras linguagens.
- Seguindo, você executou o código e obteve a saída “Hello World”.

- Tenha em mente que, mesmo que você não tenha ainda conhecimento sobre o conceito de funções, não precisa se preocupar. Basta ter em mente àquilo que acabamos de verificar sobre o “print”.

Erros

Caso você tenha tido algum problema, vamos repassar o conteúdo do slide de erros comuns, que se encontra a parte.

- O problema mais comum, nesse caso, é se o python não estiver instalado corretamente. Também pode acontecer de não se localizar no diretório correto.

Um fato interessante

- Observe:
- Um código que faz o “Hello World!”, porém em C++;

```cpp
#include <iostream>

using namespace std;

int main(){
        cout<<"Hello World!"<<endl;
        return 0;
}
```

- Ficou um pouco grande não?
- Em python, o mesmo problema acabou de ser resolvido em apenas uma linha. Este é só um exemplo do verdadeiro “poder da linguagem”. Em programas maiores você irá perceber que esta abstração tende a ser um dos grandes benefícios de se programar em python.

Programa ou Script?

Esta aqui é uma dúvida comum, sobretudo para aqueles que estão começando a aprender programação.

Resumidamente, um programa diz respeito à um arquivo executável pelo próprio computador, enquanto que um script deve ser executado por um programa intermediário.

Como já citamos anteriormente, python é uma linguagem que é interpretada.

Dependendo das funcionalidades do seu código, o mesmo poderá ser denoninado programa ou script.

- “Então não há nenhum tipo de compilação em python?” Não é bem assim.
- Experimente tentar executar o mesmo código anterior, agora com:

```
$ python -m hello.py
```

- Atente ao diretório.


hello.py


hello.pyc

- Este arquivo “.pyc” consiste no já citado bytecode Python.
- Esta operação “-m”, chamada diretiva, serve para operação com módulos em linha de comando. Só para o caso de você querer saber.
- Não se preocupe se isto lhe parece um tanto confuso. Geralmente não serão necessárias operações neste nível em seu uso diário.

Um ídolo para muitos projetos

- Após ter instalado o python em sua máquina, você pode ter notado que um outro programa foi instalado também, chamado IDLE.
- Ele consiste em uma IDE (Ambiente Integrado de Desenvolvimento) de python, distribuída pela própria comunidade python. É Open Source e, atualmente, está disponível para Linux, MacOS e Windows. O IDLE foi feito totalmente na linguagem python.
- Seu nome vem da sigla “Integrated DeveLopment Environment”.

- O IDLE pode ser usado em dois “modos” distintos: Shell e Editor.
- Shell (Modo Interativo): Basicamente, ele permite rodar linhas de código python como “comandos”, escrevendo rodando cada um linha a linha.
- Editor: Permite codificar os programas “da forma comum”, através da edição de arquivos de código fonte. Esta é a mesma maneira que usamos no nosso primeiro exemplo.
- Dentre as diferenças dos dois, além do propósito, estão também as opções dos menus de contexto dos mesmos.

- Caso queira testar o modo Shell do IDLE, pode usar a seguinte expressão:

```
>>> 2 + 2
```

- Que deverá ter como saída:

```
4
```

- Ainda não estudamos esta parte de álgebra em python, porém isto vale apenas como um “teste básico”.

- O Shell também é uma ferramenta bastante útil para testes de comandos. Quando em dúvida sobre o funcionamento de um comando específico, é interessante que o programador passe a recorrer a ele.
- Em nosso minicurso, no entanto, trabalharemos mais com os editores. Caso queira, pode utilizar o editor do próprio IDLE.

Outras IDEs

- Além do IDLE, existem diversas outras IDEs para a programação em python. Em nosso minicurso, apesar de indicarmos o uso apenas de um editor de texto simples, você pode usar o editor ou IDE de sua preferência.
- Mais para frente, iremos verificar funcionalidades próprias das IDES.

Exercício #1 (5 min.)

Modifique o código que criamos anteriormente (o Hello World) para imprimir as seguintes informações, linha a linha:

- “Nome: Seu nome”
- “Idade: Sua idade”
- “Opiniao: Sua opinião sobre o minicurso”

```
Luiz Antonio
20
Opiniao: Se melhorar, estraga...
```

- Dica: Neste primeiro exercício, você irá precisar repetir o “print” algumas vezes a mais que no código anterior.

Dúvidas?
Sugestões?
Críticas?