

LINGUAGEM DE PROGRAMAÇÃO PYTHON

PYTHON PROGRAMMING LANGUAGE

Igor Rodrigues Sousa Silva,
Rogério Oliveira da Silva

RESUMO

Este artigo foi desenvolvido para apresentar um pouco sobre a linguagem de programação Python, e mostrar porque é preciso investir nesta tecnologia. Para este fim, será apresentado um pouco de sua história, benefícios, funcionamento e alguns exemplos e comparações, fortalecendo o uso da ferramenta. Contudo, ao final deste artigo o leitor será capaz de responder a pergunta, “Por que usar Python?”.

Palavras chave: Python, investir, benefícios, funcionamento, alguns exemplos.

ABSTRACT

This article was developed to introduce a little about the python programming language, and to show why it is necessary to invest in this technology. To this end, you will learn a little bit about its history, benefits, functioning, some examples and comparisons, strengthened the use of the tool. However, at the end of this article the reader will be able to answer the question, "Why use Python?".

Keywords: Python, investing, benefits, running, some examples.

INTRODUÇÃO

Com a evolução da tecnologia, a programação nos dias de hoje, se tornou uma necessidade – mesmo que implicitamente – pois, o ser humano já não vive sem tecnologia e, a tecnologia não se mantém sem a programação. Devido a esse avanço, novas ferramentas e linguagens de programação foram surgindo, cada qual com suas especialidades, e nesse contexto surge a linguagem Python. Apesar de poucos a conhecerem, muitos a utilizam, pois, ela está vários ambientes, principalmente na web.

Gerida pela PSF (Python Software Foundation) a linguagem de programação Python é extremamente simples e robusta. Sua arquitetura bem projetada, fornece um bom desempenho e legibilidade do código. Esta linguagem

está presente em várias aplicações do nosso cotidiano, dando suporte as demais tarefas.

A linguagem possui ampla versatilidade, podendo atuar bem na área comercial, ou em áreas mais específicas como por exemplo: no desenvolvimento científico, geoprocessamento e em aplicações mobile, tanto isoladamente ou integrada a outras ferramentas, o que expande ainda mais o mundo Pythônico.

Neste artigo será discutido as características e vantagens da linguagem Python. Este estudo se foca em apresentar, algumas comparações com determinadas ferramentas, além de cobrir onde esta tecnologia está sendo aplicada e, através dessas informações, será levado ao leitor os motivos do porquê usar a linguagem de programação Python.

O objetivo geral deste artigo consiste em popularizar a linguagem de programação Python, de modo que seja abordado suas características gerais, tais como sua história, arquitetura e benefícios. Também será debatido o impacto desta tecnologia na atualidade, tanto para o desenvolvedor que poderá usufruir de tais utilidades, quando para o usuário que será capaz de desfrutar de softwares mais ricos e poderosos.

A HISTÓRIA DO PYTHON

Em 1982, na cidade de Amsterdã, capital da Holanda, Guido Van Hossum, um dos desenvolvedores da linguagem de programação ABC, trabalhava no CWI (Instituto de Pesquisa Nacional para Matemática e Ciência da Computação) em um sistema operacional distribuído chamado amoeba. Devido as grandes falhas deste sistema com a linguagem C, Guido resolve então criar uma linguagem que possa resolver tais problemas. Van Hossum Queria desenvolver uma tecnologia fácil e intuitiva pois, segundo ele, determinados softwares programados em C eram bastante complexos pelo fato de, possuírem uma codificação extensa e, apenas programadores experientes conseguiam entender alguns programas escritos em C. Após Criação da nova linguagem veio a parte da nomeação. Para esse fim o CWI possuía um padrão de nomeações, que era baseado em algum nome referente a Televisão. Logo o Holandês batizou a linguagem de Python devido a seu programa favorito, o *Monty Python's Flying Circus*. Até então seu nome não tinha nenhuma relação com a serpente píton, porém, o primeiro livro de programação em Python

foi produzido pela editora O'Reilly, e cada livro dessa editora, possui um animal em sua capa, e o animal para este livro foi a própria serpente píton.

Exemplo 01



Programming Python: <http://learning-python.com/about-pp1e.html>.

Na década de 90, fora finalizado o projeto Python. Com isso Guido Van Hossun se muda para os Estados Unidos e, inicia o projeto CP4E (Computer programming for everybody – programação de computadores para todos) financiado pela DARPA (Defense Advanced Research Projects Agency - Agência de Projetos de Pesquisa Avançada de Defesa) onde ele ensina programação para todos que tenham interesse. Em 2001 foi fundada a PSF (Python Software Foundation) que é uma organização sem fins lucrativos que, mantém e coordena o uso da linguagem. Atualmente a PSF é apoiada por grandes empresas como o Google, Microsoft e a Globo.com que também utiliza o Python nos seus sistemas.

Com o passar do tempo o Python foi evoluindo e, aos poucos sua estrutura foi ganhando mais componentes (Guido van Hossun, 2005), como por exemplo, as compreensões de lista (list comprehension), as funções filter, map, e diversas outras características. Hoje em dia, a linguagem é uma das mais usadas no mercado, e já vem instalada de fábrica em vários sistemas operacionais, como por exemplo, o AmigaOS, netBSD, MacOS e todas as distribuições Linux.

ONDE O PYTHON É APLICADO?

A linguagem Python é aplicada conforme seu propósito, entretanto, esta ferramenta não possui propósito específico, ou seja, não existe uma tarefa definida que o idioma deva suprir, pois, seu objetivo é geral (Lucas prado, 2017). Por

exemplo, a linguagem PHP é voltada para o desenvolvimento web, já a edição SE(Standard Edition) do Java é voltado para o desenvolvimento desktop, já o Python não possui objetivo. A linguagem oferece suporte a desktops, desenvolvimento web, aplicações mobile, geoprocessamento, processamento de imagens, robótica, Data Science, programação para hardware (Harduíno e Raspbarrypi), desenvolvimento de games, biotecnologia e também no desenvolvimento científico, pois, trabalha com números grandes e complexos, e possui também, diversas bibliotecas para essa finalidade como Scipy e NumPy. Além de aplicações comerciais, empresariais e científicas, com linguagem Python é possível produzir filmes com computação 3D, ou desenvolver games. Alguns jogos desenvolvidos com Python foi, Civilization 4, QuArk e um dos mais conhecidos, Battlefield 2.

Exemplo 02

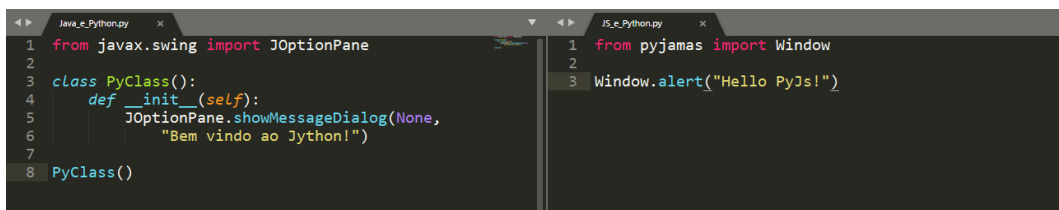


Battlefield: http://iphone.mob.org.pt/game/battlefield_2.html.

Atualmente o Python se encontra no cotidiano de muitos usuários, pois, ele está presente em vários lugares, como por exemplo, nos buscadores do Google processando pesquisas, ou nas transmissões de vídeo do YouTube, e também nos algoritmos bem elaborados da NetFlix. A linguagem faz parte de diversas outras grandes outras empresas como Dropbox, Yahoo, Zope Corporation, Industrial Light & Magic, Walt Disney Feature Animation, Pixar, NASA, NSA, Red Hat, Nokia, IBM, Yelp, Intel, Cisco, HP, Qualcomm, e JPMorgan Chase.

O Python trabalha de forma isolada, porém, com esta linguagem também é possível realizar integrações a outras linguagens, como por exemplo com Java, C, JavaScript, linguagens de marcação como Html entre outras. Abaixo está uma pequena implementação do Python sendo compilado sobre o Java.

Python Sendo Executado com Java e JavaScript. - Exemplo 03



```
1 from javax.swing import JOptionPane
2
3 class PyClass():
4     def __init__(self):
5         JOptionPane.showMessageDialog(None,
6             "Bem vindo ao Jython!")
7
8 PyClass()
```

```
1 from pyjamas import Window
2
3 Window.alert("Hello PyJs!")
```

Fonte: Igor Rodrigues Sousa Silva.

CARACTERÍSTICAS DA LINGUAGEM

Uma linguagem simples e elegante voltada para o desenvolvimento ágil. O Python segue algumas metodologias de desenvolvimento como, RAD (Rapid Application Development - Desenvolvimento rápido de aplicações). É uma metodologia focada na redução dos desperdícios e, melhorando assim a qualidade do código, diminuindo tempo, e custos de produção. Segue técnicas de desenvolvimento DRY (*Don't Repeat Yourself* - Não se repita). Onde, a codificação não deve possuir ambiguidades e cada porção de conhecimento do sistema deve ser independente, para que, no momento de um problema, outras reponsabilidade do software não sejam afetadas. E o princípio KISS (Keep it simple stupid – Mantenha simples, idiota). Onde, sua ideia é que o software deve ser simples, pois, quanto mais simples mais rápido ele será executado, será mais leve, e a curva de aprendizado do usuário será menor.

“Bonito é melhor do que feio Simples é melhor do que complexo” (Tim Petters, 2004).

Saída Padrão em Python, C e Java. Respectivamente. – Exemplo 04



```
1 print('Hello Python!')
```

```
1 #include <stdio.h>
2 int main(){
3     printf("Hello C!");
4     return 0;
5 }
```

```
1 package appteste;
2 public class AppTeste {
3     public static void main(String[] args) {
4         System.out.println("Hello java!");
5     }
6 }
```

Fonte: Igor Rodrigues Sousa Silva.

“Contagens de legibilidade. Embora a praticidade supera a pureza.” (Tim Petters, 2004).

Uma linguagem de altíssimo nível e bastante próxima do entendimento humano. Muitos componentes são escritos com a própria palavra, como os operadores lógicos E, OU, igual e diferente. Em linguagens tradicionais, estes

operadores são escritos respectivamente “&&, |, ==, != ” como no Java, C, Php, C# entre outras. Já em Python estes mesmos operadores são escritos com as próprias palavra “and, or, is, is not” respectivamente.

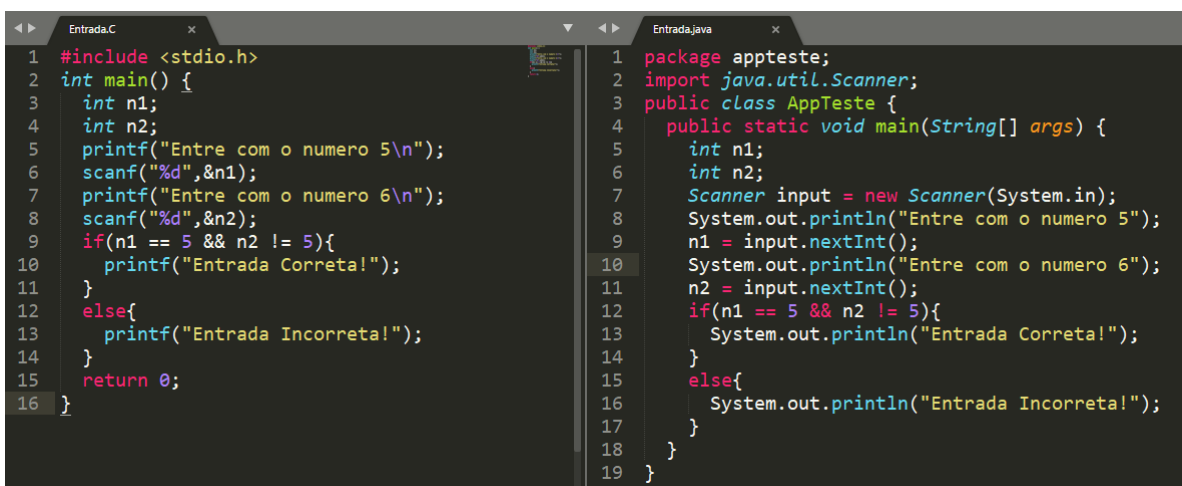
O Python é uma linguagem com variáveis fortemente tipadas, e sua tipagem é feita de forma dinâmica, ou seja, não é necessário declarar tipos primitivos, basta definir os nomes das variáveis e adicionar o valor que internamente a linguagem realiza as conversões necessárias.

A linguagem Python não possui “;” (ponto e vírgula) para o fechamento de uma linha de código, entretanto, toda linha necessita de ser finalizada para iniciar outra, e em Python as linhas são finalizadas de forma implícita pelo caractere \0, que não é preciso ser digitado, pois, é o caractere final de qualquer linha de código ou texto.

Em blocos de código a linguagem não trabalha com “{}” (chaves), basta identificar o início do bloco com “:” (dois pontos) e indentar a codificação dentro do escopo do bloco, dessa forma é entendido como deverá ser compilado. A indentação é válida para funções, classes, loops ou qualquer bloco de código. “Se a implementação é difícil de explicar, é uma má ideia. Se a implementação for fácil de explicar, pode ser uma boa ideia.” (Tim Petters, 2004).

Abaixo está a demonstração de um pequeno software que, pede ao usuário que entre com os valores 5 e 6 respectivamente, caso as entrada sejam obedecidas será exibido na tela “Entrada Correta!”, caso contrário será mostrado “Entrada Incorreta!”.

Entrada de Valores com C e Java. Respectivamente. - Exemplo 05



```
Entrada.C
1 #include <stdio.h>
2 int main() {
3     int n1;
4     int n2;
5     printf("Entre com o numero 5\n");
6     scanf("%d",&n1);
7     printf("Entre com o numero 6\n");
8     scanf("%d",&n2);
9     if(n1 == 5 && n2 != 5){
10        printf("Entrada Correta!");
11    }
12    else{
13        printf("Entrada Incorreta!");
14    }
15    return 0;
16 }

Entrada.java
1 package appteste;
2 import java.util.Scanner;
3 public class AppTeste {
4     public static void main(String[] args) {
5         int n1;
6         int n2;
7         Scanner input = new Scanner(System.in);
8         System.out.println("Entre com o numero 5");
9         n1 = input.nextInt();
10        System.out.println("Entre com o numero 6");
11        n2 = input.nextInt();
12        if(n1 == 5 && n2 != 5){
13            System.out.println("Entrada Correta!");
14        }
15        else{
16            System.out.println("Entrada Incorreta!");
17        }
18    }
19 }
```

Fonte: Igor Rodrigues Sousa Silva.

Entrada de Valores com Python. - Exemplo 06

```
Entrada.py x
1 n1=input('Entre com numero 5')
2 n2=input('Entre com numero 6')
3 if n1 is 5 and n2 is not 5:
4     print('Entrada Correta!')
5 else:
6     print('Entrada Incorreta!')
```

Fonte: Igor Rodrigues Sousa Silva.

Nos exemplos anteriores e posteriores, foi optado por não usar acentuações para não alongar ainda mais a codificação!

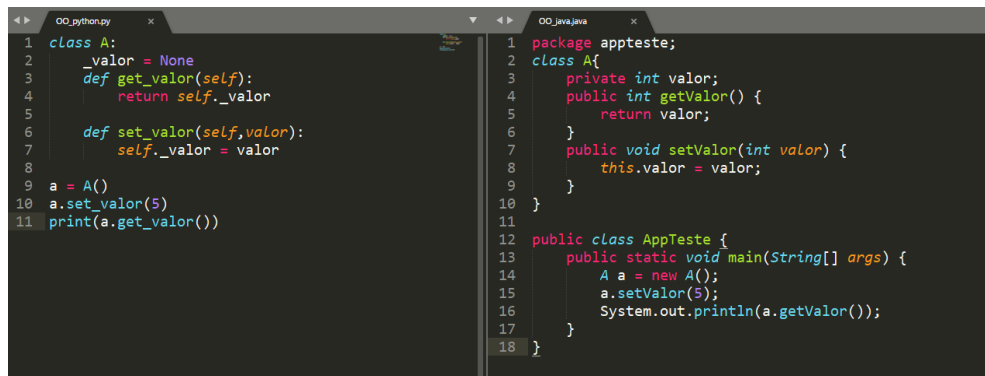
Python é uma linguagem Orientada a Objetos mas também é possível desenvolver de forma estruturada. A linguagem possui vários componentes já inclusos na sua biblioteca padrão, sem a necessidade de downloads adicionais, como, Tkinter para interfaces gráficas, RE(Regular Expressions - Expressões regulares), para manipulação de textos, entre outras bibliotecas. A linguagem Python possui coleções de dados bem elaboradas que estão presentes diretamente no núcleo da linguagem, sem necessidade de importações extras, que são, Listas (arrays), Tuplas (lista de dados imutáveis), e Dicionários (conjunto de dados organizados por chave e valor). Estas coleções possuem diversos métodos auxiliares como, “lista.append()” para adicionar mais um item na lista, utilizando um alocação dinâmica de memória. “dicionário.get(“chave”)” para pesquisar algum valor pela chave, entre outros.

A linguagem Python por padrão, vem com um terminal interativo que, permite que código seja executado em tempo real, e que o resultado (ou erro) da codificação seja exibida em tempo de execução (ou assim que o usuário pressionar a tecla “Enter”), o que ajuda muito em testes rápidos. A linguagem de programação Python é bastante explícita na questão de erros, o que ajuda muito o desenvolvedor a resolver problemas.

“O explícito é melhor do que o implícito. Os erros nunca devem ser transmitidos silenciosamente.” (Tim Petters, 2004).

Abaixo está pequena codificação orientada a objetos. Nela contém os métodos assessores, get e set, variáveis de instância, e controle de visibilidade.

Orientação Objeto com Python e Java. Respectivamente. – Exemplo 07



```
OO_python.py x
1 class A:
2     _valor = None
3     def get_valor(self):
4         return self._valor
5
6     def set_valor(self, valor):
7         self._valor = valor
8
9 a = A()
10 a.set_valor(5)
11 print(a.get_valor())

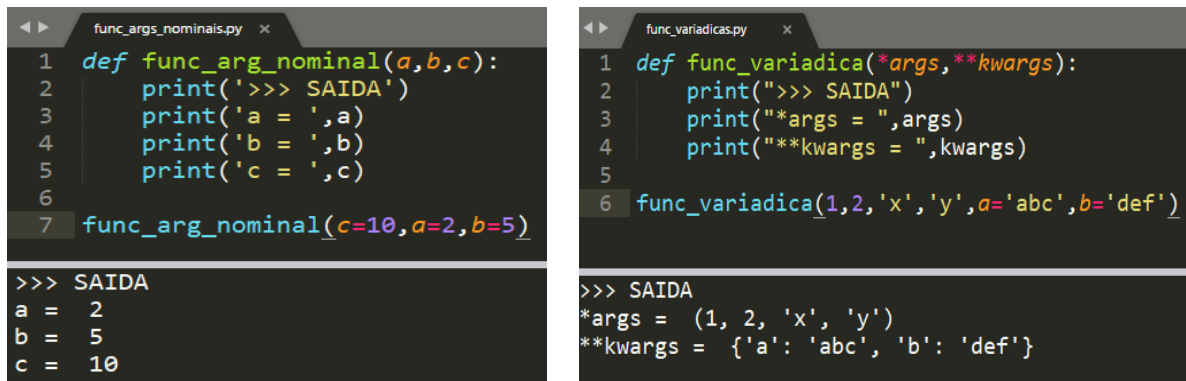
OO_java.java x
1 package appteste;
2 class A{
3     private int valor;
4     public int getValor() {
5         return valor;
6     }
7     public void setValor(int valor) {
8         this.valor = valor;
9     }
10 }
11
12 public class AppTeste {
13     public static void main(String[] args) {
14         A a = new A();
15         a.setValor(5);
16         System.out.println(a.getValor());
17     }
18 }
```

Fonte: Igor Rodrigues Sousa Silva.

_ (underline) no início, diz que o componente é privado. Self é o argumento padrão de todo método de classe, o mesmo é a instância da própria classe, e serve para utilizar variáveis de instâncias e métodos dentro do método que está chamando os demais componentes. Self representa o escopo da classe como um todo.

“Como muitas outras linguagens de programação, o Python suporta modularidade, na medida em que você pode quebrar grandes pedaços de código em peças menores e mais gerenciáveis”. (Paul Barry, 2017). O Python é extremamente modular, onde cada arquivo é um módulo, e a importação do mesmo é executada apenas uma vez em toda a aplicação, o que incentiva ainda mais o desenvolvedor a utilizar vários módulos em seus aplicativos. Outras características interessantes são, as funções que recebem argumentos passados de forma nominal, ou seja, não é necessário passar parâmetros na ordem em que foram definidos, basta passar nome e valor que o Python entende onde o valor deve ser inserido. Além de funções com argumentos nomeados a linguagem possui funções Variádicas, isto é, quando implementadas recebem estes dois parâmetros (*args, **kwargs) e, desta forma, é possível inserir quantos valores o desenvolvedor desejar. A notação (*args) recebe valores em tuplas, e (**kwargs) recebe chave e valor.

Funções e Argumentos. – Exemplo 08



```
func_arg_nominalis.py x
1 def func_arg_nominal(a,b,c):
2   print('>>> SAIDA')
3   print('a = ',a)
4   print('b = ',b)
5   print('c = ',c)
6
7 func_arg_nominal(c=10,a=2,b=5)

>>> SAIDA
a = 2
b = 5
c = 10

func_variadicis.py x
1 def func_variadic(*args,**kwargs):
2   print(">>> SAIDA")
3   print("*args = ",args)
4   print("**kwargs = ",kwargs)
5
6 func_variadic(1,2,'x','y',a='abc',b='def')

>>> SAIDA
*args = (1, 2, 'x', 'y')
**kwargs = {'a': 'abc', 'b': 'def'}
```

Fonte: Igor Rodrigues Sousa Silva.

O FUNCIONAMENTO DA LINGUAGEM PYTHON

Python é uma linguagem interpretada e multiplataforma, podendo ser encontrada em relógios, televisores, celulares, entre outros aparelhos. A interpretação Python funciona da seguinte forma: Arquivos de extensão .py são codificados pelo desenvolvedor e quando compilados geram um arquivo .pyc (conhecido como byte code) logo, estes arquivos são interpretados pela máquina virtual Python (VM) e se transformam em código nativo da máquina.

Em Python *tudo é um objeto*. Tudo existente na linguagem é um objeto, variáveis são objetos, classes são objetos, módulos são objetos, números são objetos e tudo existente na linguagem. Em C/C++ uma variável é um espaço/local na memória, o valor da variável é armazenada neste local, ao atribuir a variável será modificado este valor. Portanto uma variável é um espaço de memória com ou sem valor. Já em Python uma variável é um *name* (nome) que possui um *binding* (ligação) com um objeto, mas ao atribuir esta variável o objeto em si não será totalmente modificado, apenas uma parte será alterada.

Para comprovar isso, existe uma função chamada "dir()", que verifica todos os atributos dos objetos. Quando esta função é usada no número 5 será retornado:

Atributos de um Número. - Exemplo 09

```
>>> dir(5)
['_abs_', '_add_', '_and_', '_bool_', '_ceil_', '_class_', '_delatt
r_', '_dir_', '_divmod_', '_doc_', '_eq_', '_float_', '_floor_', '_
floordiv_', '_format_', '_ge_', '_getattr_', '_getnewargs_', '_g
t_', '_hash_', '_index_', '_init_', '_init_subclass_', '_int_', '_in
vert_', '_le_', '_lshift_', '_lt_', '_mod_', '_mul_', '_ne_', '_ne
g_', '_new_', '_or_', '_pos_', '_pow_', '_radd_', '_rand_', '_rdiv
mod_', '_reduce_', '_reduce_ex_', '_repr_', '_rfloordiv_', '_rlshift
_', '_rmod_', '_rmul_', '_ror_', '_round_', '_rpow_', '_rrshift_', '_
rshift_', '_rsub_', '_rtruediv_', '_rxor_', '_setattr_', '_sizeof_',
'_str_', '_sub_', '_subclasshook_', '_truediv_', '_trunc_', '_xor_'
, '_bit_length_', '_conjugate_', '_denominator_', '_from_bytes_', '_imag_', '_numerator_',
'_real_', '_to_bytes_']
```

Fonte: Igor Rodrigues Sousa Silva.

Se 5 realmente fosse um número ele não conteria estes atributos.

A linguagem Python disponibiliza um sistema de gerenciamento de memória automático, que é responsável por alocar e desalocar memória quando os objetos não possuem nenhuma referência. De tempos em tempos o Garbage Collector (Coletor de Lixo) percorre uma lista de objetos registrados e verifica o número de referência para cada um deles. Se as referências dos objetos forem equivalentes a 0, ou seja, não existe referência para aquele objeto, o GC desaloca a memória usada por aquele objeto. Dessa forma, o software fica mais leve e sem resíduos que atrapalhe o desempenho da aplicação.

DESENVOLVIMENTO WEB COM DJANGO

Django é um framework para desenvolvimento web escrito em Python. Ele baseia-se no padrão MTV (Model, Template, View). No ano de 2003, dois Desenvolvedores (Adrian Holovaty e Simon Willison) da World online, uma empresa jornalística no Kansas na cidade de Lawrence, abandonaram o PHP e começaram a utilizar o Python para o desenvolvimento de seus sites. Ao construir sites ricos e iterativos, eles começaram a extrair padrões e estruturas genéricas que permitiam o desenvolvimento mais ágil de aplicações web. Em 21 de julho de 2005, o framework Django foi lançado. A ferramenta teve seu nome inspirado no guitarrista de jazz *Django Reinhardt*.

Atualmente a ferramenta é bastante estável, várias empresas utilizam-a em seus projetos como, o Mozilla em seu site, a rede social de ideias Pinterest, o Instagram em aplicativo, e a Globo em seus sites, globo.com e globoesporte.com o maior site de esportes da América Latina, que recebe mais de 20 milhões de

visitantes por mês. O framework utiliza todos os princípios e metodologias da linguagem Python como DRY, RAD e KISS.

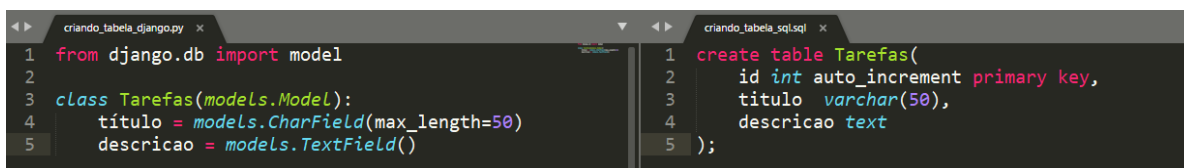
O Django abstrai a parte mais repetitiva no desenvolvimento web como, autenticações, manuseio de banco de dados, segurança padrão, desenvolvimento de formulários, e algumas outras características, e traz módulos prontos para realizar essas atividade com mais agilidade, praticidade e menos codificação. Dessa forma o programador desenvolve grandes aplicações em um tempo significativamente menor.

CARACTERÍSTICAS DO DJANGO

ORM

Ao iniciar um projeto Django, a ferramenta já fornece para o analista toda a estrutura básica de arquivos necessários para o desenvolvimento. Nesta estrutura de arquivos é disponibilizado um arquivo para o modelo (models.py), outro para o controle (views.py) um para administração da base de dados (admin.py), outro para a lógica da aplicação (app.py), além de arquivos de configurações para o gerenciamentos do projeto (settings.py). Mesmo com esta estrutura de arquivos o desenvolvedor ainda está livre para criar quantos arquivos ele desejar. O Django possui diversas API's (Application Programming Interface), e uma delas permite uma maneira diferente de manipulação da base de dados, a API chamada ORM (Object-Relational Mapping - Mapeamento objeto-relacional). Esta API permite que o analista de sistemas manipule todas a operações em banco de dados por códigos python sem a necessidade de scripts sql. Ao criar uma classe com seus atributos no arquivo (models.py) a API irá espelhar um SQL através desta classe, e criará uma tabela no banco de dados, dispensando o uso de códigos SQL's.

ORM do Django X SQL. – Exemplo 10



```
criando_tabela_django.py x      criando_tabela_sqlsql x
1 from django.db import model
2
3 class Tarefas(models.Model):
4     titulo = models.CharField(max_length=50)
5     descricao = models.TextField()

1 create table Tarefas(
2     id int auto_increment primary key,
3     titulo varchar(50),
4     descricao text
5 );
```

Fonte: Igor Rodrigues Sousa Silva.

Neste caso será criada uma tabela chamada tarefas com três campos (titulo,descricao,id). Automaticamente, o django cria um campo id inteiro como auto incremento para todas as tabelas, sem a necessidade codificar isto, (Caso o desenvolvedor deseje não utilizar este campo criado de forma automática, ele poderá apagá-lo, alterá-lo ou excluí-lo da forma que lhe convém). Também é possível que o desenvolvedor trabalhe com o sql propriamente dito.

Além de criação de tabelas, também é possível realizar diversos tipos de operações na base de dados, como inserção, atualização, busca, entre outras ações. Abaixo está um exemplo de um pequeno CRUD(Create, Read, Update, Delete) na tabela tarefas.

CRUD com django x SQL. – Exemplo 11

<code>Table.objects.create(title='teste',desc='teste')</code>	<code>Insert into table values('teste','teste');</code>
<code>Table.objects.all()</code>	<code>Select * from table;</code>
<code>Table.objects.filter(id=1).update(title='teste2')</code>	<code>Update table set title='teste2' where id=1;</code>
<code>Table.objects.filter(id=2).delete()</code>	<code>Delete from table where id=2;</code>

Fonte: Igor Rodrigues Sousa Silva.

O framework também fornece tabelas criadas automaticamente, dessa forma, não é necessário que o desenvolvedor as crie, porém, a ferramenta permite que ele as altere ou as exclua a qualquer momento. Automaticamente o Django cria uma tabela de usuários, uma tabela de controle de permissões, uma tabela de log, uma para controle de sessões, permissões de usuário, tipo de conteúdo, grupos, permissões de grupos, grupos de usuários, e das migrações do sistema (modificações no banco). Cada uma possui sua finalidade, podendo ser usadas em aplicações de pequeno ou grande porte.

Além do ORM, o Django possui a API admin, que permite que o desenvolvedor administre seu banco de dados sem uso de ferramentas externas, podendo fazer diversas operações ou permitir que o próprio usuário do sistema administre seu banco. Por meio desta API, ele controla a exibição de seus dados, buscas, filtragens, inserções, entre outras ações, tudo isso no próprio sistema em que está sendo desenvolvido.

TEMPLATES

O framework Django possui um sistema de templates. Este sistema permite que desenvolvedor crie templates dinâmicos e mais enxutos através de tags de template, e filtros. Os templates Django possuem sua própria sintaxe, para separar a linguagem de template da linguagem de servidor. Estas tags também oferecem todo o poder de estruturas condicionais, laços de repetição, inclusões e até mesmo heranças, tornando assim o html mais resumido e dinâmico.

Tags de templates. – Exemplo 12

Estruturas	Tags de Template
<i>If</i>	<pre>{% if condicao %} {% endif %}</pre>
<i>For</i>	<pre>{% for L in lista%} {% endfor %}</pre>
<i>Valor de uma variável</i>	<pre>{{ variavel }}</pre>
<i>Herança</i>	<pre>{% extends 'template.html' %}</pre>
<i>Blocos de código</i>	<pre>{% block conteudo%} {% endblock %}</pre>
<i>Inclusão de código</i>	<pre>{% include 'template.html' %}</pre>
<i>Filtros</i>	<pre>{{variavel filtro }}</pre>

Fonte: Igor Rodrigues Sousa Silva.

FORMULÁRIOS

Como formulários são características de todo sistema que processe dados de entrada, o Django possui um sistema de formulários onde, formulários html podem ser desenvolvidos diretamente do código Python no servidor. Dessa forma o template html se torna mais limpo e mais rápido para ser desenvolvido, trazendo também mais facilidade no momento da manutenção do sistema.

Formulários Django x Formulários HTML. – Exemplo 13

```

formulário_django_server.py
1 from django import forms
2
3 class FormVendas(forms.Form):
4     nome = forms.CharField(max_length=50)
5     data = forms.DateField()
6     email = forms.EmailField()
7     produto = forms.CharField(max_length=50)

formulário_django_front.html
1 <form>
2     {{ FormVendas }}
3 </form>

```

```

1 <form>
2     {%csrf_token%}
3     <label>Nome</label>
4     <input type="text" id="id_nome" required/>
5     <label>Email</label>
6     <input type="email" id="id_email" required/>
7     <label>Data</label>
8     <input type="date" id="id_data" required/>
9     <label>Produto</label>
10    <input type="text" id="id_produto" required/>
11 </form>

```

Fonte: Igor Rodrigues Sousa Silva.

Outro ponto importante que o framework Django ressalta é que, a maioria dos sistemas, campos em tabelas existentes na base de dados são exatamente os mesmos campos contidos nos formulários. Logo, para agilizar este processo, o Django possui uma classe chamada ModelForm. Para utilizar formulários baseado em modelos, cria-se uma classe que herda de ModelForm e tem como atributo o modelo que será espelhado em um formulário, e os campos a serem exibidos.

Formulários baseado em modelos (ModelForm). – Exemplo 14

```

model_form.py
1 from django.forms import ModelForm
2 from app.models import Produto
3
4 class FormProduto(ModelForm):
5
6     class Meta:
7         model = Produto
8         fields = "__all__"

model_form_front.html
1 <form>
2     {{ FormProduto }}
3 </form>

```

```

1 <form>
2     {%csrf_token%}
3     <label>Nome</label>
4     <input type="text" id="id_nome" required/>
5     <label>Descricao</label>
6     <input type="text" id="id_descricao" required/>
7     <label>Quantidade</label>
8     <input type="text" id="id_quantidade" required/>
9     <label>Preço</label>
10    <input type="number" id="id_preco" required/>
11 </form>

```

Fonte: Igor Rodrigues Sousa Silva.

O Django já vem com vários formulários pré definidos, bastando apenas importá-los, como formulários de criação de usuários, de login, Alteração de senha, entre outros.

SEGURANÇA

A segurança depende muito de como o sistema está sendo implementado, junto a ferramentas extras para a proteção de seus dados. O Django por padrão já fornece ferramentas e técnicas, para um desenvolvimento mais seguro. Dentre elas

será citado de modo geral três proteções contra os ataques mais comuns em sites e aplicações web.

PROTEÇÃO CONTRA FALSIFICAÇÃO DE SOLICITAÇÃO DE SITE CRUZADO (CSRF)

Os ataques CSRF(Cross-Site Request Forgery- Falsificação de solicitações entre sites) permite que o usuário mal intencionado, execute ações usando as credenciais de outro usuário, sem que ele saiba. Para se proteger contra ataques desta natureza, o framework fornece um token de autenticação de usuário, que é modificado a cada requisição feita pelo mesmo, fazendo assim com que o usuário seja sempre identificado antes de qualquer tipo de requisição, caso contrário, o servidor retornará um erro HTTP 403 (Proibido), dizendo que o usuário está proibido de realizar requisições.

SQL INJECT

Os ataques conhecidos como SQL inject, (injeção de sql) permitem que o usuário mal intencionado, consiga injetar scripts sql na aplicação, podendo resultar em perda de dados e vazamento de informações. Com a ORM do django (citada acima), estes ataques diminuem drasticamente devido a troca de consultas sql pela codificação da própria API ORM. Ao entrar consultas vindas da ORM o resultado é tratado através do driver do banco de dados, dessa forma não é possível que entre consultas sql cruas dentro do sistema, o que diminui os riscos destes ataques.

VALIDAÇÃO DO HOST NO CABEÇALHO DA REQUISIÇÃO

O Django utiliza o host de cabeçalho para construir urls. Os valores dos hosts são validados para evitar ataques no tipo CSS(Cross Site Scripting) onde outros sites emitem requisição ao seu próprio site. Através desta validação o framework configura os hosts permitidos, e com isso, apenas urls permitidas podem emitir requisições, bloqueando qualquer outro tipo de solicitação externa.

PROTEÇÃO CONTRA O CLICKJACKING

Clickjacking é o tipo de ataque em que um site malicioso envolve outro site dentro de um frame. Logo ao realizar determinada ação nesse site malicioso, sem que usuário perceba, ele estará efetuando uma ação em outro site. O Django

contém a proteção contra o clickjacking através do navegador, onde ele impede que seja processado sites em frames.

Além destes sistemas o django possui diversos outros que defendem o usuário contra vários ataques (Todos estes sistemas de proteção podem ser configurados, otimizados ou desabilitados). Portanto, usando o framework Django não é necessário ter um conhecimento aprofundado na área de segurança, pois, por padrão, a ferramenta implanta diversos métodos de proteção.

Mesmo o desenvolvedor sendo leigo na segurança da informação, ele será capaz de desenvolver um sistema protegido contra os demais tipos de ataques. Vale ressaltar que, toda segurança contém suas limitações, como todo sistema, e que a maior segurança provém do conhecimento de quem está implementando o software.

CONSIDERAÇÕES FINAIS

Como pretendido, o objetivo deste artigo foi fornecer uma breve apresentação introdutória sobre a linguagem de programação Python, debatendo algumas de suas características gerais. Este documento visa transmitir ao leitor uma base inicial, para que o mesmo possa responder a seguinte pergunta “Por que usar Python?”.

Para alcançar o objetivo deste artigo, foi se apresentando diversos conceitos e explicações, desde implementações básicas da linguagem e seu funcionamento interno, até o uso de alguns frameworks.

Foi citado vários exemplos práticos e suas descrições de um modo didático. Junto a parte prática, também fora fornecido diversas comparações, fazendo com que, o usuário veja as diferenças entre linguagens de programação, e possa captar através das comparações, as vantagens de uma tecnologia para outra. Portanto através deste artigo o leitor possui informações que permitem que o mesmo responda outra pergunta “Já sabe o porquê de usar Python?”.

REFERÊNCIAS

Tim Petters. **Zen do Python**. Disponível em <<https://www.python.org/dev/peps/pep-0020/>>. Acessado em 31/08/2017.

Security in Django. Disponível em <<https://docs.djangoproject.com/en/1.11/topics/security/>>. Acessado em 31/08/2017.

Python Brasil. **Funcionamento do Garbage Collector**. Disponível em <<https://wiki.python.org.br/FuncionamentoGarbageCollector>>. Acessado em 21/08/2017.

Victor Pantoja. **Performace utilizando Django: durma tranquilo**. Disponível em <<https://www.tiespecialistas.com.br/2014/01/performance-utilizando-django-durma-tranquilo/>>. Acessado em 25/08/2017.

Knupp jaff. **Everything I know about Python**. Disponível em: <<https://jeffknupp.com/blog/2013/02/14/drastically-improve-your-python-understanding-pythons-execution-model/>> Acessado em 23/08/2017.

Beazley,David e Jones k Brian. **Python CookBook**. 3º edição. 1005 Gravenstein Highway North, Sebastopol. Editora O'Reilly Media. 2013-05-08.145.

Romano Fabrizio. **Learning Python**. 5º edição. BIRMINGHAM – MUMBAI. Editora Packt Publishing. 2015

Guido Van Hossum. The fate of reduce() in Python 3000. Disponível em <<http://www.artima.com/weblogs/viewpost.jsp?thread=98196>> acessado em 02/10/2017.

Paul Barry. Code Reuse: Functions and Modules. Disponível em <https://www.safaribooksonline.com/library/view/head-first-python/9781491919521/ch04.html>. Acessado em 02/10/2017.