

Introdução a linguagem Python

Objetivos do mini curso

- Conhecer a linguagem.
- Noção de programação utilizando Python.
- Aprender o básico.

Conteúdo

- Uma visão da linguagem: O modo python de programar
- O interpretador dinâmico
- Salvando e rodando programas em python
- Variáveis
- Manipulação de tipos
- Listas, tuplas e dicionários
- Condicionais
- Estruturas de repetição
- Funções
- Orientação a Objeto em python
- Classes e New Style Class
- Métodos Mágicos
- Properties
- Exceções
- Django - Introdução e noções básicas

Uma visão da linguagem: O modo python de programar

Hello World em java

- package hello_world;
- public class Main {
- public static void main(String[] args) {
- System.out.println("Hello World!");
- }
- }

Hello World em python

- print 'Hello World!'

Uma visão da linguagem: O modo python de programar

- Blocos por indentação
- Sintaxe limpa
- Amigável e de fácil aprendizado

Salvando e rodando programas em python

- Necessário ter o interpretador
- Salve o arquivo com a extensão py
- Na linha de comando digite python caminho_para_o_arquivo

Interpretador dinâmico

- `dir(objeto)` #lista todos os métodos do objeto
- `help(objeto.metodo)` #retorna o docstring do objeto, função, método, modulo, etc...

Manipulação de tipo

- Tipagem forte e dinâmica
- `a = 1` #tipo inteiro
- `a = "` #string
- `a = 1.0` #float
- `a = 10000000000000000000000000` #long
- `str(dado)` #converte para string
- `float(dado)` #converte para float
- `int (dado)` #converte para inteiro
- `long (dado)` #converte para long

Listas, tuplas e dicionários

- Listas - estrutura de dados, os elementos são colocados em sequência e para cada um deles é determinado um índice numérico, ex: `a = []`
- Tuplas - funciona da mesma forma que listas, mas são imutáveis, ex: `a = (1,2,3)`
- Dicionários - diferem pois seus índices não precisam ser numéricos, ex: `a {'teste' : 'teste_cont'}`

Condicionais

- Sintaxe if:
 - if (condição):
 - Processamento
 - elif (condição):
 - Processamento
 - else:
 - Processamento

Condicionais

- Não existe case, uma solução para situações onde esta estrutura seria indicada é fazer uma implementação com dicionários:
 - `dict_case = {'conda' : funca, 'condb' : funcb}`
 - `dict_case[var_cond]`

Estruturas de repetição

- Sintaxe for:
 - for (expressao):
 - Bloco
 - else:
 - Bloco

- Sintaxe while:
 - while (expressão):
 - Bloco
 - else:
 - Bloco

Funções

- Como definir funções:
 - `def nome_da_funcao (parametros):`
 - Corpo

Orientação a objeto no python

- Para o python, tudo é um objeto.
- Implementação Simples
- `class nome_da_classe:`
 - Métodos e/ou atributo

Classes e New style class

- New style class são parte de um esforço para unificar os tipos built-in
- `class nome_da_classe (object):`
 - Métodos e atributos
- Properties são uma das mais importantes features das new style class.
- A função `super(Classe, instancia)` também não irá funcionar em old-style.

Métodos mágicos

- Em python, qualquer método começando com `__` e terminado com `__` é considerado um método mágico.
 - `__init__(self)`: -> construtor

Properties

- Uma forma de prover encapsulamento sem a necessidade de atributos privados
- class A(object):
 - Def `__init__(self):`
 - `_legal = 'Com certeza'`
 - def `geta(self):`
 - `return self._legal`
 - def `seta(self, val):`
 - `self._legal = val`
 - `legal = property (geta, seta)`

Exceções

- Atos incomuns na execução de um programa
- Sintaxe:
- try:
 - Bloco
- except (exceção):
 - Tratamento da exceção
- raise objeto exceção ou raise mensagem-> levanta uma exceção

Django - Introdução e noções básicas

- Django é um framework para desenvolvimento web
- Encoraja desenvolvimento rápido e design pragmático e limpo
- Para iniciar uma aplicação é simples, depois de instalado e configurado digite num prompt de comando
 - `django-admin.py startproject meuprojeto`

Django - Introdução e noções básicas

- Criando uma app
 - Dentro da pasta do projeto digite
 - `python manage.py startapp minhaapp`
- Dentro do diretório da app o arquivo `models.py` controla os modelos da aplicação

Django - Introdução e noções básicas

- Para ativar sistema administrativo automatico edite o arquivo `urls.py` dentro da pasta do projeto, descomente as linhas
 - `from django.contrib import admin`
 - `admin.autodiscover()`
 - `(r'^admin/', include(admin.site.urls))`
- Na linha de comando digite `python manage.py syncdb` para criar as tabelas, tanto do sistema administrativo quanto do seu site

Django - Introdução e noções básicas

- Tornar dados da sua aplicação alteráveis pelo sistema adm.
 - Crie um arquivo chamado `admin.py` dentro da pasta da aplicação
 - Altere, adicionando as seguintes linhas
 - `from meuprojeto.minhaapp.models import Modelos`
 - `from django.contrib import admin`
 - `admin.site.register(Modelo)`

Bibliografia

- HETLAND, Magnus Lie. *Beginning Python From Novice to Professional*. Nova York: Apress, 2005.