



Apresentação

Thaís Vergani

thais.vergani@cakeerp.com

Cake ERP

<http://cakeerp.com/>

Tecnologias

Python 2.7

- Kivy (PDV)

- Pyramid

- Django

Javascript

- jQuery

- Angular JS

HTML

CSS

MySQL

SQLite

Sintaxe

- ▶ Identações
- ▶ Boa Legibilidade
- ▶ Uso de caracteres reduzido

- ▶ PEPs - Python Enhancement Proposals (propostas de aprimoramento)
 - ▶ <https://www.python.org/dev/peps/>

Java

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

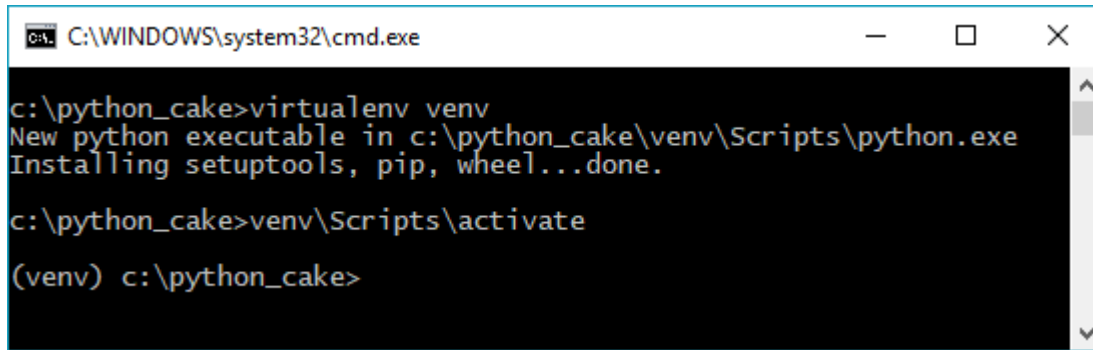
Python

```
print('Hello World!')
```

Instalando o Python

<https://www.python.org/>

Virtualenv



```
C:\WINDOWS\system32\cmd.exe

c:\python_cake>virtualenv venv
New python executable in c:\python_cake\venv\Scripts\python.exe
Installing setuptools, pip, wheel...done.

c:\python_cake>venv\Scripts\activate
(venv) c:\python_cake>
```


Bibliotecas

PIP - sistema de gerenciamento dos pacotes

- pip freeze

- pip install

- pip - help (mostra os comandos)

- easy_install

IPython

Variáveis

- ▶ Atribuição
- ▶ Reatribuição

Operadores

<	estritamente menos que
<=	menos ou igual que
>	estritamente maior que
>=	maior ou igual que
==	igual
!=	diferente
is	identidade do objeto
is not	não identidade do objeto

+	adição
-	subtração
*	multiplicação
/	divisão
//	divisão inteira
**	exponenciação
%	resto de divisão

or	lógico ou
and	lógico e
not	lógico de negação

Variáveis

- ▶ Atribuição Aumentada
 - ▶ -=, *=, /=, //=, %= e **=.
- ▶ Convenção para nomes

If

- ▶ If simples
- ▶ Operador ternário

Exercícios

Boolean

- ▶ Use if

Strings

<https://docs.python.org/2/library/string.html>

<https://docs.python.org/2.7/library/stdtypes.html#string-methods>

▶ Métodos

- ▶ in
- ▶ .upper()
- ▶ .lower()
- ▶ .capitalize()
- ▶ .count()
- ▶ format()

▶ # Estilo de formatação ruim
`print('hello ' + name + '!')`

Estilo de formatação antiga
`print('Hello %s!' % name)`

Estilo de formatação nova
`print('Hello {}'.format(name))`

Listas

<https://docs.python.org/2/library/string.html>

- ▶ Uma lista é uma coleção ordenada
- ▶ `append(novo_elemento)`
- ▶ `extend(segunda_lista)`
- ▶ `insert(posicao, novo_elemento)`
- ▶ `remove(valor_do_elemento)`
- ▶ `pop(posicao)`
- ▶ `index(valor_do_elemento)`
- ▶ `count(valor_do_elemento)`
- ▶ `sort(cmp=None, key=None, reverse=False)`
- ▶ `reverse()`
- ▶ `len(lista)`
- ▶ `del`
- ▶ list comprehension
- ▶ <https://pythonhelp.wordpress.com/2013/06/26/brincando-com-listas/>

Slicing

- ▶ `s[1]`
- ▶ Índice negativo: `s[-1]`
- ▶ Fatias:
 - ▶ Sintaxe básica é `s[inicio:final]`: `s[1:4]`
 - ▶ Fatiando no início da string: `s[:5]`
 - ▶ Fatiando até o final da string: `s[3:]`
 - ▶ Índice negativo também pode ser usado em fatias: `s[-3:-1]`

Estruturas de Laço

<https://docs.python.org/2.7/tutorial/datastructures.html#looping-techniques>

- ▶ For
 - ▶ in
- ▶ While
- ▶ Continue
- ▶ Break
- ▶ Enumerate
- ▶ Range
- ▶ Zip
- ▶ Sorted

Exercícios

Tuplas

<https://docs.python.org/2/tutorial/datastructures.html#tuples-and-sequences>

- ▶ Imutáveis
- ▶ Métodos
 - ▶ `index()`
 - ▶ `count()`
- ▶ *Sequence Unpacking*

Dicionários

<https://docs.python.org/2.7/library/stdtypes.html#mapping-types-dict>

- ▶ `copy()`
- ▶ `get()`
- ▶ `del()`
- ▶ `keys()`
- ▶ `clear()`
- ▶ `has_key()`
- ▶ `items()`
- ▶ dict comprehension
 - ▶ `d = dict((key, value) for (key, value) in iterable)`
 - ▶ `d = {key: value for (key, value) in iterable}`

Tipos

<https://docs.python.org/2/library/types.html>

- ▶ Tipagem dinâmica forte
- ▶ `a = b = c = 1`

Exercícios

Funções e métodos

- ▶ Comparação com java
 - ▶ Dois pontos (:) indica o início de um bloco
 - ▶ As linhas seguintes são endentadas
 - ▶ Declaração de função não especifica o tipo de retorno
 - ▶ Todas as funções retornam um valor (None se não for especificado)
- ▶ Passagem de parâmetros
 - ▶ Tipos de dados parametrizados não são especificados
- ▶ Parâmetros default

Built in Functions

<https://docs.python.org/3/library/functions.html>

Built in functions são implementadas em C dentro do interpretador do Python.
Demais trechos de código devem ser interpretados.

- ▶ Len()
- ▶ Lambda()
 - ▶ `g = lambda x: x**2`
- ▶ Filter()
- ▶ Map()
- ▶ Reduce()
- ▶ Max() e Min()
- ▶ dict()

Datetime

```
1  from datetime import date
2  now = date.today()
3  now.strftime("%m-%d-%y. %d %b %Y is a %A on the %d day of %B.")
4
5  birthday = date(1996, 9, 23)
6  age = now - birthday
7  print age.days
8
```

```
dt = datetime.strptime("21/11/06 16:30", "%d/%m/%y %H:%M")
weekday()
```

Random

<https://docs.python.org/2.7/library/random.html>

```
from random import randint  
randint(0, 9)
```

```
random.sample('abcdefghijk', 5)
```

Exercícios

Tratamento de exceções

SyntaxError, TypeError, NameError

Exercícios



Orientação a Objetos

Java

```
public class Gerente {  
    private String nome;  
    private String cpf;  
    private double salario;  
    private int senha;  
    private int numeroDeFuncionariosGerenciados;  
  
    public boolean autentica(int senha) {  
        if (this.senha == senha) {  
            System.out.println("Acesso Permitido!");  
            return true;  
        } else {  
            System.out.println("Acesso Negado!");  
            return false;  
        }  
    }  
}
```

Python

```
class Gerente(object):  
    def __init__(self, nome, cpf, salario, senha):  
        self.nome = nome  
        self.cpf = cpf  
        self.salario = salario  
        self.senha = senha  
        self.numeroDeFuncionariosGerenciados = 0  
  
    def autentica(self, senha):  
        if self.senha == senha:  
            print "Acesso Permitido!"  
            return True  
        else:  
            print "Acesso Negado!"  
            return False
```

Classes

- ▶ Declaração de classes
- ▶ Construtores (init)

```
class MinhaClasse:  
    """Um exemplo simples de classe"""  
    i = 12345  
    def f(self):  
        return 'olá, mundo'
```

Instâncias

- ▶ Atributos
- ▶ Métodos

Métodos

- ▶ Funções definidas dentro da classe
- ▶ O primeiro parâmetro de um método faz referência ao objeto, e por convenção se utiliza “*self*”

`__init__`

- ▶ Método construtor.
- ▶ É chamado automaticamente quando a classe é instanciada.

Métodos de Classe e Estáticos



Herança

- ▶ *object* é a classe base padrão

- ▶ sintaxe:

```
class DerivedClassName(BaseClassName):
```

```
...
```

```
class C(B):
```

```
    def method(self, arg):
```

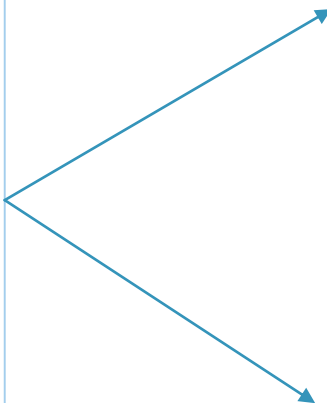
```
        super(C, self).method(arg)
```


super()

```
class Base(object):  
    def __init__(self):  
        print 'Construindo a classe Base'  
  
class Derivada(Base):  
    def __init__(self):  
        print 'Construindo a classe Derivada'
```

```
x = Derivada()
```

- ▶ o construtor de Base não foi chamado em nenhum momento



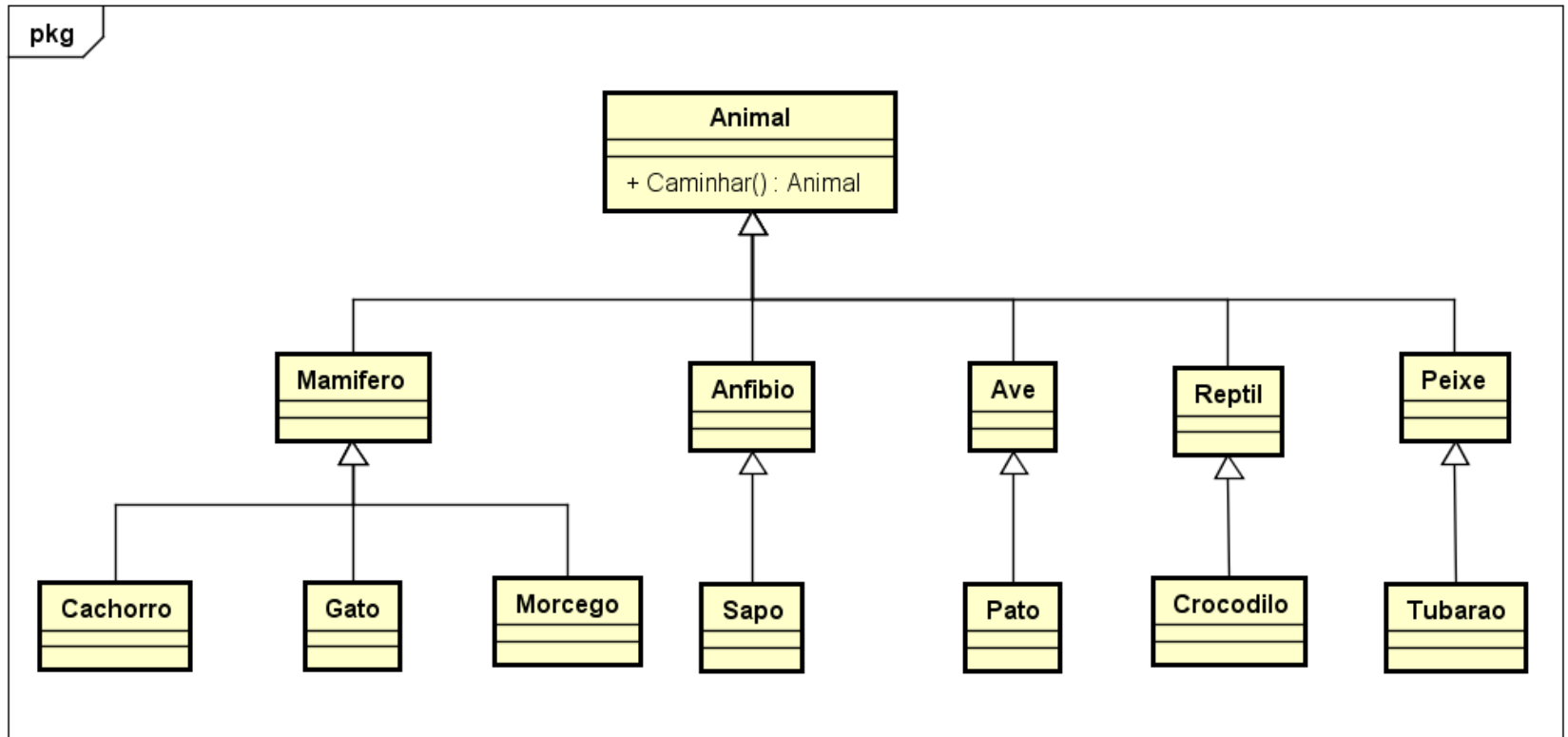
```
class Derivada(Base):  
    def __init__(self):  
        Base.__init__(self)
```

```
class Derivada(Base):  
    def __init__(self):  
        super(Derivada, self).__init__()
```

Herança

- ▶ Herança múltipla
- ▶ `isinstance()`
- ▶ `issubclass()`

Exercício



Exercício

- ▶ Veículo
 - ▶ Carro
 - ▶ Avião
 - ▶ Bicicleta
- ▶ atributos:
 - ▶ modelo
 - ▶ velocidade
 - ▶ combustível
 - ▶ portas
 - ▶ peso
 - ▶ consumo
- ▶ funções:
 - ▶ abastecer
 - ▶ andar

Args e Kwargs

- ▶ `'a teste'.replace('a ', 'o ')`
- ▶ `args = ['a ', 'o ']`
- ▶ `'a teste'.replace(args)`
- ▶ `'a teste'.replace(*args)`

[illegible]

Frameworks

- ▶ Django
- ▶ Pyramid
- ▶ Flask
- ▶ Kivy

Criando um projeto em Django

- ▶ `django-admin startproject cake_project`
- ▶ `python manage.py runserver`
- ▶ `python manage.py startapp products`

Database

MySQL

SQLite

Migrações de banco

- ▶ `python manage.py makemigrations products`
- ▶ `python manage.py migrate`

Database API

<https://docs.djangoproject.com/en/2.0/ref/models/queriesets/#>

- ▶ `python manage.py shell`
- ▶ `a = Product(name='tenis', price_sell=10)`
- ▶ `a.save()`
- ▶ `Product.objects.filter(id=1)`
 - ▶ <https://docs.djangoproject.com/en/2.0/ref/models/queriesets/#id4>
 - ▶ `Product.objects.get(name__contains='bolo')`
 - ▶ `Product.objects.filter(id__in=[1, 3, 4])`

Admin

- ▶ `python manage.py createsuperuser`

IDE

Pycharm

Notepad ++

Sublime

Atom

HTML

- ▶ linguagem de marcação

```
<!DOCTYPE html>
```

```
<html>
```

```
<title>HTML!!</title>
```

```
<body>
```

```
<h1>This is a heading</h1>
```

```
<p>This is a paragraph.</p>
```

```
</body>
```

```
</html>
```

CSS

```
body {  
    background-color: lightblue;  
}  
h1 {  
    color: white;  
    text-align: center;  
}  
p {  
    font-family: verdana;  
    font-size: 20px;  
}
```

CSS

```
body {  
    background-color: lightblue;  
}  
h1 {  
    color: white;  
    text-align: center;  
}  
p {  
    font-family: verdana;  
    font-size: 20px;  
}
```



CSS

- ▶ Bootstrap <https://getbootstrap.com/>

JS

```
<script>
```

```
function myFunction() {  
    var x = document.getElementById("demo");  
    x.style.fontSize = "25px";  
    x.style.color = "red";  
}
```

```
</script>
```

```
<button onclick="myFunction()">Click Me!</button>
```

JS

- ▶ Bibliotecas
- ▶ Frameworks
- ▶ <https://www.javascripting.com/>

GIT

GIT - sistema de controle de versão de arquivos

GIT

1. `git config --global user.name "Thais Vergani"`
 - ▶ `git config user.name`
2. `git config --global user.email "thais.vergani1@gmail.com"`
3. `git init`: inicializa o repositório no diretório atual
 - ▶ status dos arquivos
 - ▶ untracked
 - ▶ unmodified
 - ▶ modified
 - ▶ staged

GIT

- ▶ `git status`
- ▶ `git add (nome_do_arquivo)`
 - ▶ `git add -A` stages All
- ▶ `git commit -m "mensagem do seu commit"`
- ▶ `git log`
 - ▶ `git log --graph`
- ▶ `git show (hash do commit)`
- ▶ `git diff`
 - ▶ `git diff --name-only`

GIT HUB

- ▶ GitHub é uma plataforma de hospedagem de código-fonte com controle de versão usando o Git.
- ▶ criar repositório
 - ▶ `git remote add origin https://github.com/thaisvergani/arquivos_curso.git`
 - ▶ `ssh`
 - ▶ protocolo para autenticação
 - ▶ chave
- ▶ `git remote`
 - ▶ `git remote -v`
- ▶ `git push -u origin master`
 - ▶ origin: pra onde vai
 - ▶ master: de onde vem
- ▶ `git clone`

GIT

- ▶ Branch
- ▶ `git branch`
- ▶ `git checkout (nome_do_branch)`
- ▶ `git branch -D (nome_do_branch)`
- ▶ `git merge (nome_do_branch)`
- ▶ `git rebase (nome_do_branch)`

Olimpíada de Inverno 2018

Os Jogos Olímpicos de Inverno são realizados a cada quatro anos. A última edição aconteceu do dia 9 ao dia 25 de fevereiro deste ano, e foi sediada na Coreia do Sul.

O Comitê Olímpico Internacional - COI, que é a organização responsável pelas Olimpíadas, precisa de um software para controlar a quantidade de medalhas em tempo real, e disponibilizar esta informação para o público.

O Comitê deve cadastrar os países participantes e a quantidade de medalhas.

O ranking dos países deve ser exibido com base no número de medalhas de ouro, prata e bronze, respectivamente.

- ▶ Diagrama ER
- ▶ Protótipos de tela

Desafio

- ▶ campo minado
- ▶ labirinto