

Especialização Desenvolvimento de Aplicações Web e Móveis Escaláveis

Turma 2021-2022

Big Data com Python

André Morais

andre.morais@luizalabs.com

09/2022



Why Choose Python for

**BIG
DATA**

Uni-FACEF
Centro Universitário Municipal de Franca



Especialização Desenvolvimento de Aplicações Web e Móveis Escaláveis

Módulo: BIG DATA COM PYTHON - Professor: André Morais

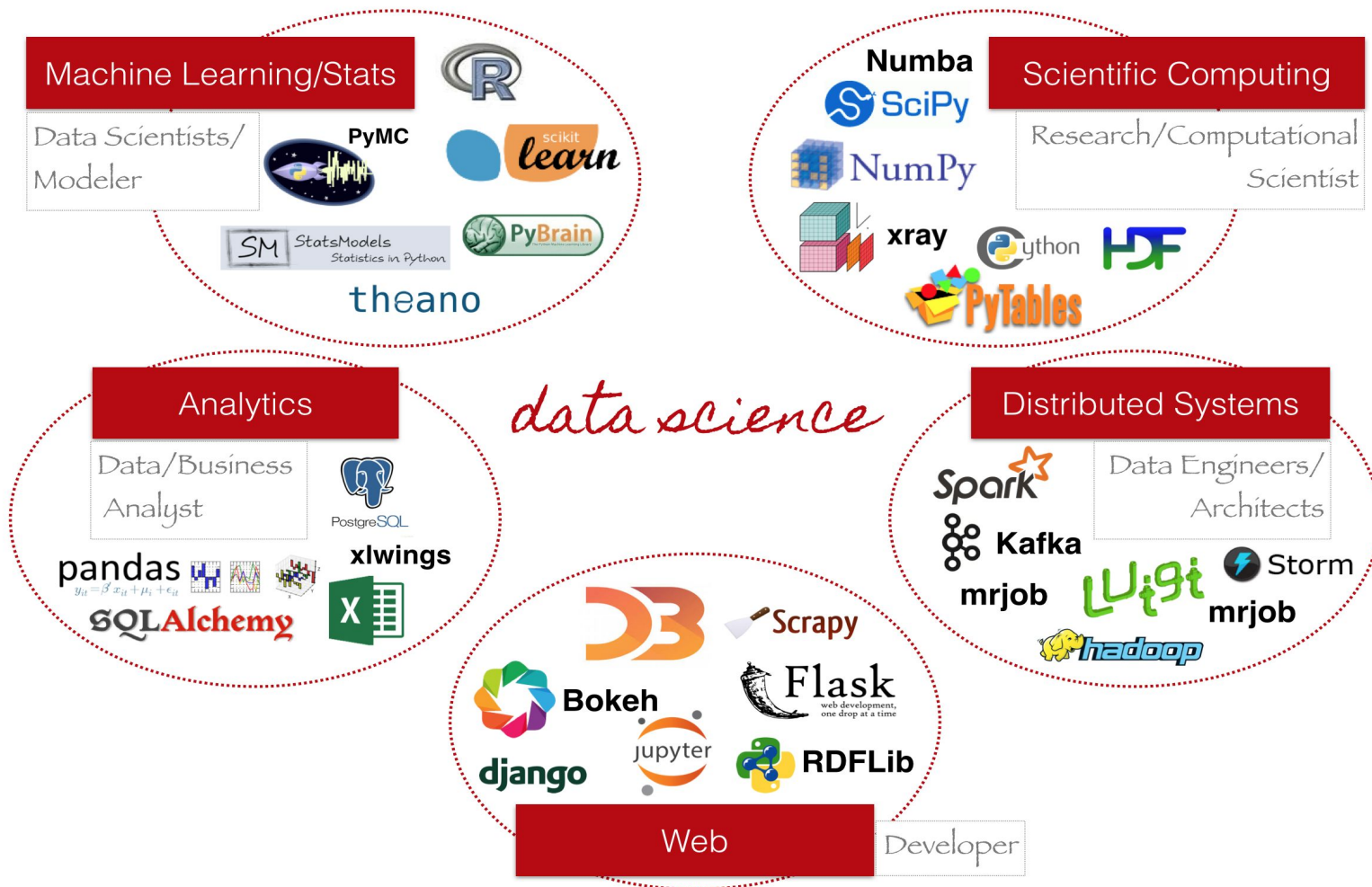
Turma 2019-2020

Python Por que é utilizado para Big Data?

- Linguagem de uso geral
- Infinitude de bibliotecas disponíveis - <https://pypi.org/>
(Processamento distribuído e computação científica)
- Escalabilidade e flexibilidade
- Simplicidade e fácil aprendizado
- Multiplataforma
- Comunidade grande e ativa
<https://python.org.br/>
- Ferramentas como Jupyter Notebook e Anaconda
- Convenções e melhores práticas - PEP8
<https://www.python.org/dev/peps/pep-0008/>



Python Ecossistema e bibliotecas



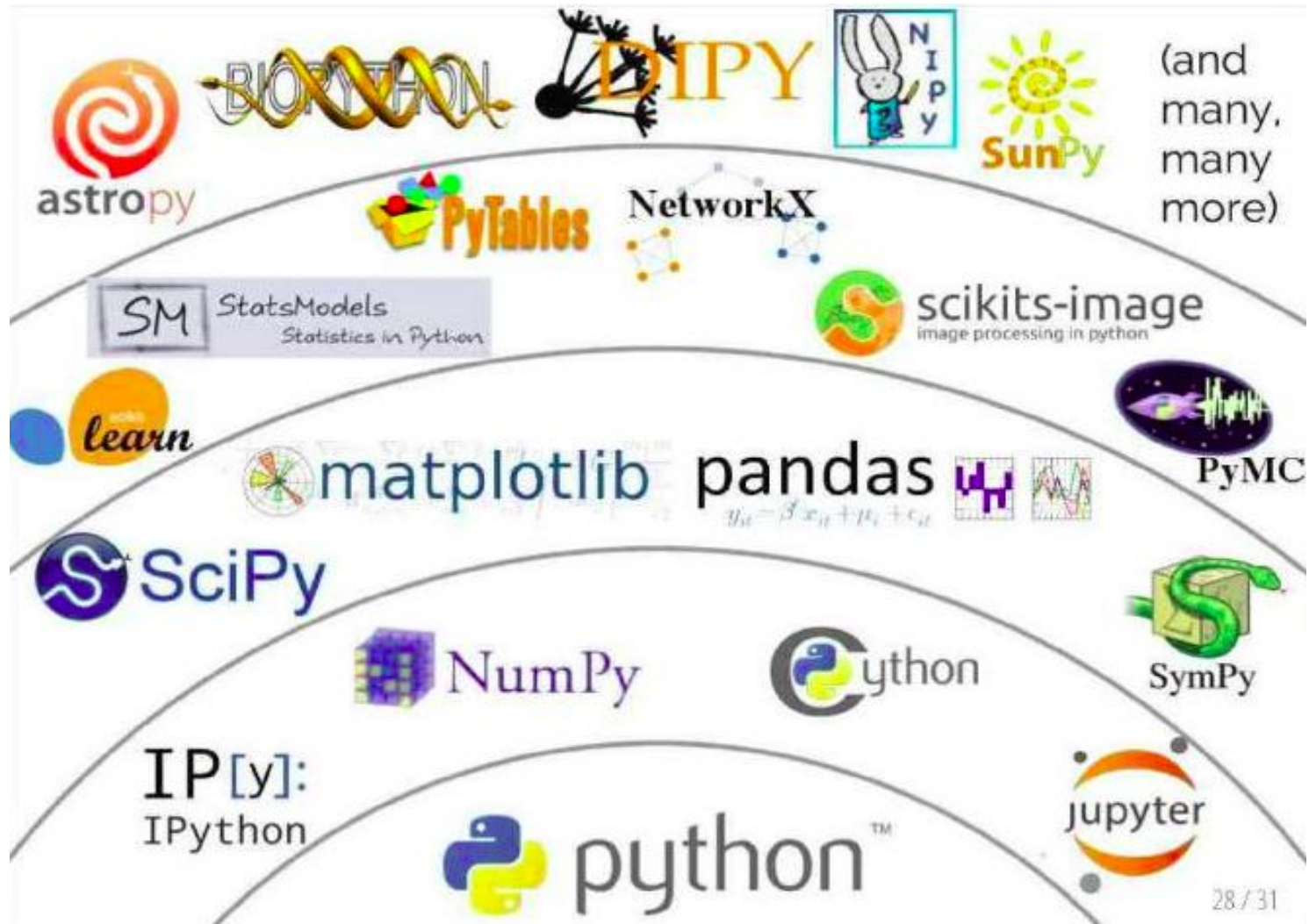
O Python possui uma infinidade de pacotes e bibliotecas de código aberto, que nos dá escalabilidade e flexibilidade para trabalhar uma Stack e Ecossistemas ricos para processamento Bigdata, Data Science e muito mais.

Python Frameworks para Computação Distribuída



Spark e Hadoop são frameworks para processamento paralelo e distribuído de grandes volumes de dados em clusters de computadores. Bastante utilizado pela Engenharia de Dados.

Python Uma infinidade de bibliotecas para Ciência de dados, Machine Learning e Inteligência Artificial



28 / 31

Curiosidade Algumas empresas que usam Python



Empresas no Brasil: <https://python.org.br/empresas/>

Python Ferramentas de Trabalho

O **Jupyter Notebook** é um shell interativo muito utilizado na exploração de dados, que permite unir código e texto.

Isso faz com que sejam uma poderosa ferramenta para desenvolver seus scripts e registrar seus experimentos com dados.



ANACONDA®

Anaconda é um pacote que oferece uma distribuição do Python, juntamente com várias bibliotecas para análise de dados, e ferramentas como Jupyter Notebook e a IDE Spyder.



O **Google Colaboratory** ou **Colab** permite escrever código Python no seu navegador, da mesma forma intuitiva do Jupyter Notebook

- Nenhuma configuração é necessária
- Roda em uma VM provisionada em um host Google
- Armazenamento no Google Drive
- Acesso gratuito a GPUs e TPUs (Tensor Processing Unit)
- Muitas bibliotecas e frameworks pré-instalados
- Facilidade de Compartilhamento
- Compatibilidade com os notebooks do Jupyter



Conhecendo o Jupyter Notebook e Google Colab

<https://docs.continuum.io/anaconda/packages/pkg-docs/>

<https://colab.research.google.com/>



Conhecendo a Linguagem Python

Python Um breve histórico

- Python foi criado em 1991, pelo holandês Guido van Rossum

[@gvanrossum](#)

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

- Desde de sua criação o Python tem essa essência do Open Source, da colaboração e da comunidade.
- Em 1991, Guido van Rossum enviou um e-mail para a internet com o código fonte da linguagem convidando as pessoas a discutirem e usarem o Python, e esse simples e-mail se propagou de forma incrível pela internet.
- Em 2020, Python libera entre as 10 melhores linguagens de programação segundo artigo do “OlhaDigital.com”:

[Python lidera ranking entre as 10 melhores linguagens de programacao em 2020](#)

- Documentação: <https://docs.python.org/pt-br/3/library/>

Python Tipos de dados básicos

- **Números:** int, long, float, complex
- **Booleanos:** bool (True, False)
- **Strings:** str e unicode
- **Listas e tuplas:** list, tuple
- **Dicionários:** dict
- **Conjuntos:** set, frozenset
- **None**

Python Variáveis

- Às variáveis são atribuídos objetos e valores;
- O operador de atribuição é “=”;
- Variáveis não “contém” os objetos em si, são apenas referências;
- Variáveis não têm tipo, mas sim os objetos a que se referem;
- Variáveis não são criadas automaticamente;
- Uma variável precisa ser iniciada antes de ser utilizada em uma expressão.

Python Palavras reservadas

Palavras reservadas definem as regras e a estrutura da linguagem e não podem ser usadas como nomes de variáveis.

- **and**
- **assert**
- **break**
- **class**
- **continue**
- **def**
- **del**
- **elif**
- **else**
- **except**
- **exec**
- **finally**
- **for**
- **from**
- **global**
- **if**
- **import**
- **in**
- **is**
- **lambda**
- **not**
- **or**
- **pass**
- **print**
- **raise**
- **return**
- **try**
- **while**
- **yield**

Python Comentários

- Usa-se o caracter especial **#** no início do comentário, e até o final da linha será ignorado pelo interpretador, exceto quando aparece em uma string;

```
# Exemplo de comentário de uma linha
```

- As “**doc string**” são utilizadas para comentários de várias linhas, geralmente para documentar o código. São representadas por 3 aspas simples ou duplas no início e final do texto

```
"""
```

```
Exemplo de comentário de várias  
linhas
```

```
"""
```

Python Indentação

- A indentação em Python é obrigatória
- Ela quem determina o escopo de uma função ou bloco
- Aumentar a legibilidade do código
- Espaços a mais ou desnecessários gera erro na aplicação
- Como melhores práticas utilize **4 espaços** ou **Tab**, seja conciso em qual for utilizar.

Python Blocos e comandos aceitos

- if / elif / else
- try / except / finally / else
- for / else
- while / else
- class - Definição de classes
- def - Definição de funções e métodos
- with

Python Operadores Aritméticos

OPERADOR	OPERAÇÃO	PRECEDÊNCIA	DESCRIÇÃO
+	Adição	0	Realiza a soma dos operandos. Ex: $a + b$
-	Subtração	0	Realiza a subtração dos operandos. Ex: total - desconto
/	Divisão	1	Realiza a divisão dos operandos. Ex: $12 / 2$
//	Divisão inteira	1	Retorna a parte inteira da divisão. Ex: $3 // 2$ (resulta em 1)
*	Multiplicação	1	Multiplica os operandos. Ex: $2 * 5$
**	Exponenciação	2	Eleva o operando a esquerda pelo operando a direita. Ex: $5 ** 2$ (cinco ao quadrado)
%	Módulo	2	Obtém o resto da divisão dos operandos. Ex: $7 \% 2$ (Resulta em 1)

As operações com menor precedência são executadas por último. Estas precedências podem ser alteradas fazendo uso do parêntese (), dando precedência para a expressão que estiver dentro do parêntese.

Python Operadores Relacionais

OPERADOR	COMPARAÇÃO	DESCRIÇÃO
==	Igualdade	Dois sinais de = compara se dois valores são idênticos. Ex: a == b
!=	Diferença	Compara se dois valores são diferentes. Ex: a != b
>	Maior	Compara se o primeiro valor é maior que segundo. Ex: a > b
<	Menor	Compara se o primeiro valor é menor que o segundo. Ex: a < b
>=	Maior igual	Compara se o primeiro valor é maior ou igual ao segundo valor. Ex: a >= b
<=	Menor igual	Compara se o primeiro valor é menor ou igual ao segundo. Ex: a <= b

As expressões relacionais sempre retornarão um valor lógico (**True** ou **False**)

Python Operadores Lógicos

OPERADOR	OPERAÇÃO	PRECEDÊNCIA	DESCRIÇÃO
or	Disjunção	1	A disjunção entre duas operações resultara verdadeiro se um dos valores for verdadeiro
and	Conjunção	2	A conjunção resultará em verdadeiro se, e somente se, todos os valores examinados forem verdadeiros
not	Negação	3	A negação inverte o valor lógico da variável examinada. Se o valor for verdadeiro ele se tornará falso, e vice-versa.

Python Funções Intrínsecas ou Built-in

Built-in Functions				
<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	



Mãos na massa, ou melhor no código \o/