Contents

[Antes de mais nada 3](#_Toc116163943)

[- Qual e a configuração recomendada para usar o Python: 4](#_Toc116163944)

[- Existe alguma recomendação para trabalhar com os dados então? 4](#_Toc116163945)

[- Existe alguma recomendação de organização de código? 4](#_Toc116163946)

[- Como eu procuro ajuda sobre o Python? 4](#_Toc116163947)

[1. O mundo de análise de dados 5](#_Toc116163948)

[O que está acontecendo? 5](#_Toc116163949)

[E o que é BIG DATA? 5](#_Toc116163950)

[E a ciência de dados? 5](#_Toc116163951)

[Novas tecnologias e ferramentas 6](#_Toc116163952)

[Onde eu me encaixo nisso? 10](#_Toc116163953)

[2. O que é uma linguagem de programação? 11](#_Toc116163954)

[Visão geral sobre linguagens de programação 11](#_Toc116163955)

[Exemplos e utilidades 12](#_Toc116163956)

[Compilada VS Interpretada 12](#_Toc116163957)

[Estruturadas, orientadas a objeto e funcionais 13](#_Toc116163958)

[Linguagens estruturadas 13](#_Toc116163959)

[Linguagens orientadas a objeto 13](#_Toc116163960)

[Linguagens funcionais 13](#_Toc116163961)

[ACL & HCL? Python? SQL? Fórmulas do Excel 14](#_Toc116163962)

[3. E por qual motivo Python (e SQL)? 16](#_Toc116163963)

[Vamos entender o ecossistema Python 16](#_Toc116163964)

[Adoção do Python ao redor do mundo (e do tempo) 17](#_Toc116163965)

[Passado, presente e futuro (Python, machine learning, R e Julia) 19](#_Toc116163966)

[4. Primeiros passos com o Python 20](#_Toc116163967)

[Baixando, instalando e configurando ambiente - Anaconda 20](#_Toc116163968)

[Download 21](#_Toc116163969)

[Instalando 22](#_Toc116163970)

[Configurando e primeiros passos 27](#_Toc116163971)

[Se ambientando com a IDE 34](#_Toc116163972)

[Variáveis, funções, objetos, chamadas e outros... 43](#_Toc116163973)

[Preparando o seu primeiro notebook para variáveis 43](#_Toc116163974)

[Preparando o seu primeiro notebook para laços de repetição e operadores lógicos 48](#_Toc116163975)

[Preparando o seu primeiro notebook para funções 54](#_Toc116163976)

[Trabalhando com tabelas dentro do Python 57](#_Toc116163977)

[Operações em tabelas 62](#_Toc116163978)

[Amostragem 64](#_Toc116163979)

[Joins e agregações 65](#_Toc116163980)

[Concat 65](#_Toc116163981)

[Minha primeira análise de dados 66](#_Toc116163982)

[Sobre o exercício: 66](#_Toc116163983)

[Suspeitas e Insights 67](#_Toc116163984)

# Antes de mais nada

O conteúdo dessa apostila e desse curso foi feito pensando em pessoas desenvolvedoras/analistas ACL que querem iniciar a sua vida de analise de dados com Python, e aqui precisamos entender que o ACL e o Python possuem alguma intersecção mas as comparações precisam ser feitas com cuidado;

Enquanto o Python e uma linguagem de programação aberta e você, caso queira, pode alterar absolutamente tudo o ACL possui uma serie de facilitadores para que qualquer um possa confeccionar analises automatizadas ponta a ponta.

Diagram

Description automatically generated

Essas são só algumas diferenças mais gritantes, mas e importante ressaltar que o ACL e uma linguagem de automatização enquanto o Python e uma linguagem de programação, o ACL só vai conseguir fazer aquilo que for pre programado pela Dilligent enquanto o Python e completamente aberto e você mesmo pode criar o que quiser dentro da linguagem. E isso traz ônus e bônus para as duas abordagens, fazer o que quiser e muito legal, mas começar tudo do zero exige um nível de conhecimento muito maior. (Mais sobre isso durante o conteúdo da apostila)

Você provavelmente já tem uma serie de observações que faz antes de começar uma análise de dados usando o ACL. Pense nessa sessão como uma guia de recomendações para iniciar as suas análises Python.

#### - Qual e a configuração recomendada para usar o Python:

Assim como no ACL quanto maior o volume de dados e maior a necessidade de velocidade de processamento, mais potente a maquina precisa ser. **Para apenas rodar o Python, e fazer analises simples recomendamos o seguinte setup:**  
Disco: 500gb (disco mecânico)  
RAM: 4gb  
CPU: Dual Core 2ghz CPU

**Para análises com ate 10 arquivos de 200mb, sugerimos o seguinte setup:**  
Disco: 500gb (disco mecânico)  
RAM: 8gb  
CPU: Dual Core 2.5ghz CPU

**Para análises mais complexas que isso, recomendamos o seguinte setup:**Disco: 2tb (preferível SSD)  
RAM: 16gb  
CPU: Quad Core 3ghz CPU  
*(Caso ainda sim você tenha problemas em rodar a sua análise, converse com um dos nossos especialistas para entender o que podemos fazer)*

Diferente do ACL, o Python trabalha principalmente com dados em memória RAM, por tanto quanto mais dados usarmos, mais memória ram precisamos ter.

#### - Existe alguma recomendação para trabalhar com os dados então?

No Python trabalhamos como objetos e variáveis, a regra de ouro e, se você não está mais usando alguma variável ou tabela, limpe a memória e delete esse objeto.

#### - Existe alguma recomendação de organização de código?

A comunidade Python criou um guia de estilo de código, o PEP8, lá podemos encontrar uma serie de procedimentos e padrões definidos sobre a melhor forma de como seguir com o seu código.  
[PEP 8 – Style Guide for Python Code | peps.python.org](https://peps.python.org/pep-0008/)

#### - Como eu procuro ajuda sobre o Python?

A comunidade Python não e administrada pela Dilligent, e uma comunidade aberta em que todos podem colaborar e se ajudar, por tanto a principal fonte de conteúdo são sites como [www.stackoverflow.com](http://www.stackoverflow.com) , [www.docspython.com](http://www.docspython.com) , [www.codeacademy.com](http://www.codeacademy.com) e outros.  
A regra de ouro aqui e, procure no google, provavelmente alguém já passou pelo mesmo problema que você e provavelmente já resolveu.

# O mundo de análise de dados

BLUFF: Análise de dados enquanto uma necessidade de mercado, já existe a anos e todos os profissionais que lidam com dados profissionalmente e academicamente, provavelmente já devem ter passado por várias ondas diferentes de ferramentas e conceitos. Essa sessão pretende explorar a onda em que estamos, tendencias de mercado e explorar onde nos encaixamos em tudo isso.

## O que está acontecendo?

Como de costume, a cada ano que se passa, novas ferramentas e conceitos são lançados no mercado, com a esperança de modificar a forma como analisamos dados e nos relacionamos com a forma que dados do mundo real são codificados em informações computacionais.

Ainda é muito comum encontrar pessoas que trabalharam nos famosos CPD’s (centro de processamento de dados), que tinha como objetivo armazenar, processar e reportar dados, isso tudo usando bancos de dados relacionais, excel e algumas linguagens de época. Alguns anos depois, tivemos uma outra grande onda, onde Ralph Kimball, apresentou seu conceito de Data Wharehouse’s e o mundo foi tomado pela rápida necessidade do que conhecemos hoje como B.I. (Bussiness Inteligence).  
Em meio a tantas revoluções, hoje vivemos um novo momento que culminou em dois grandes blocos de novidades, o **BIG DATA** e a **democratização da Ciência de Dados**.

### E o que é BIG DATA?

*Apesar de não ser o foco desse curso, é importante temos uma visão geral sobre o mundo de BIG DATA pois trabalhamos muito próximos a bases de dados e somos parceiros intrinsicamente ligados aos profissionais dessas áreas.*

Em linhas gerais, BIG DATA, é uma série de paradigmas, conceitos e técnicas aplicadas em ferramentas para que possamos trabalhar com; 1. Um altíssimo volume de dados (EX: petabytes) e 2. Conseguir lidar com dados desestruturados (Ex: Áudio, imagens, rotas, etc).

### E a ciência de dados?

Agora sim muito mais próximos do mundo de análise de dados, a ciência de dados é uma área que sempre existiu, só que hoje está muito mais democratizada e acessível.

Cientistas de dados, são pessoas especialistas em algoritmos e estatística. Usando essas duas áreas do conhecimento esses profissionais tem como objetivo automatizar decisões através da inteligência artificial.

E o que queremos denotar aqui é que a cada ciclo de novidades tecnológicas, o passado nunca é esquecido, essas evoluções sempre levam em consideração o que era bom antes e alinham com as apostas do futuro e do presente. Então tanto os movimentos de BIG DATA e de democratização da ciência de dados, tem forçado uma série de novidades para todos os profissionais que orbitam o mundo da analise de dados.

## Novas tecnologias e ferramentas

É importante começar essa sessão da apostila deixando bem claro que as tecnologias e ferramentas que veremos não são necessariamente novas, na verdade elas só estão sendo empregadas de uma nova forma ou ganhando cada vez mais espaço.

Aqui, temos uma lista de ferramentas/linguagens/tecnologias que tem ganhado muito espaço

|  |  |
| --- | --- |
| Nome: | Python |
| O que é? | Linguagem de programação interpretada, usada há muitos anos na academia. |
| Pontos positivos: | * Linguagem fácil de se aprender em comparação com outras * Funciona em qualquer sistema operacional * Facilmente incorporável e extensível * Completamente gratuita * Adoção em massa por companhias nacionais e internacionais * Comunidade ativa e apaixonada ❤ |
| Pontos negativos: | * Mais verbosa e complexa que o ACL * Linguagens interpretadas são menos velozes que compiladas (Tanto o Python quanto o ACL são interpretadas) |
| A importância veio de onde? | O Python surgiu como uma linguagem de programação para ensinar alunos de universidade a programar, logo a sua expectativa de mercado não era ser tão conhecida e usada como é hoje. Acontece que por estar muito forte dentro da academia, pesquisadores e cientistas acabaram usando o Python para tudo dentro do dia a dia e lentamente o Python saiu das universidades e se tornou uma linguagem utilizada no mundo todo. Apesar de lento, esse movimento aconteceu com muita facilidade uma vez que a linguagem teve muito tempo de maturidade dentro da academia além de ter vários algoritmos no estado da arte implementados logo de começo na sua base de código. |

|  |  |
| --- | --- |
| Nome: | SQL (Structured Query Language) |
| O que é? | Linguagem de marcação para manipulação de dados |
| Pontos positivos: | * Linguagem padrão em vários bancos de dados e softwares de processamento de dados * Implementação simples da teoria dos conjuntos em código * Estabelecida e conhecida no mercado a muitos anos * Tem muitos pontos de proximidade com o ACL (Scripting) que já conhecemos |
| Pontos negativos: | * Não é uma linguagem de programação com a qual você pode fazer tudo * Código fácil de se bagunçar |
| A importância veio de onde? | Na verdade, o SQL nunca perdeu a sua importância! Advindo da necessidade que várias fontes de dados diferentes tivessem uma forma em comum de se consultar, o SQL é padrão para o consumo e manipulação de dados e provavelmente você já deve ter tido algum contato com a linguagem.  Com um paradigma muito próximo ao ACL, o SQL é imprescindível para o analista de dados. |

|  |  |
| --- | --- |
| Nome: | Serviços de mensageria |
| O que é? | Uma forma de os sistemas conversarem entre si. Cada sistema se torna responsável por enviar as informações que produz para uma fila e responsável por consumir de outras filas de outros sistemas. |
| Pontos positivos: | * Baixa complexidade de implementação e manutenção * Facilidade de implementação * Suportado e recomendado para aplicações cloud based * A responsabilidade de cada time, reflete a responsabilidade de dados de cada sistema |
| Pontos negativos: | * Cada sistema tem que ser responsável por gerir sua própria fila * Cada sistema precisa garantir que consuma os dados de outras filas de forma consistente |
| Importância veio de onde? | Durante muitos anos trafegamos arquivos gigantes com muitos dados de uma vez, e isso resultava em uma série de pequenos problemas. Os serviços de mensageria se preocupam em trafegar poucas coisas, porém constantemente e distribuindo as responsabilidades de forma igual entre sistemas e times.  (Exemplos de serviços de mensageria: Kafka, RabitMq, Pub-Sub, Kynesis...) |

|  |  |
| --- | --- |
| Nome: | Bancos de dados não relacionais (No-SQL) |
| O que é? | Bancos de dados que abandonaram o conceito de entidade relação por formas diferentes de se tratar um dado. |
| Pontos positivos: | Essa sessão não se aplica, uma vez que são bancos que se aplicam a necessidades diferentes onde cada um tem o seu espaço |
| Pontos negativos: | Essa sessão não se aplica, uma vez que são bancos que se aplicam a necessidades diferentes onde cada um tem o seu espaço |
| A importância veio de onde? | Bancos de dados relacionais (EX: Oracle, SQL Server, Postgres...) tiveram e têm uma grande importância na evolução da tecnologia como conhecemos hoje, porém existem problemas de armazenamento e consulta de dados que esses bancos não conseguem mais resolver. Por isso abordagens diferentes foram empregadas, exemplos:   * Banco de dados de documentos (Ex: MongoDb, CosmosDb) 🡪 Os dados são modelados com o intuito de serem gravados em um único arquivo (normalmente Json). Provavelmente sua loja online usa um banco desse tipo para guardar o catálogo de produtos. * Bancos de dados chave valor (Ex: DynamoDb) 🡪 Aplicações de baixíssima latência precisam de consultas rápidas, talvez o seu hospital utilize esse tipo de aplicação para guardar prontuários de pacientes. * Colunares (Ex: HBase, Cassandra): 🡪 O perfeito oposto dos modelos relacionais que tínhamos no passado. Esse tipo de banco salva seus dados em linhas separadas e aguentam um tráfego de dados intenso. É bem provável que o seu aplicativo de delivery favorito utilize um desses para armazenar os pedidos que chegam aos montes em horários de pico. * Grafos (Ex: Neo4J) 🡪 Utilizando da teoria dos grafos, os dados são armazenados em nós que se conectam em outros nós através de arestas e cada um dos nós pode ter dados completamente diferente do outro. Sua rede social favorita utiliza esse tipo de banco de dados com toda certeza. |

Dentro desses, a cada dia são criadas novas tecnologias e paradigmas é sempre importante nos mantermos alertas com as novidades que surgem.

## Onde eu me encaixo nisso?

**A boa notícia disso tudo, é que usuários ACL não estão nada longe desse novo ciclo de tecnologias.**

Somados os esforços da **Diligent** e da **Quality**, a realidade de mercado **nunca passou despercebida,** não só o ACL Analytics tem passado por constantes melhorias e atualizações, como toda a plataforma Diligent funciona pensada em **acomodar análises legadas sem menosprezar quem quer aproveitar as novas tendências de mercado.**

**Esse curso é montado para desvendar o básico do Python sem esquecer que você tem um background em ACL. (Caso você queira aprender e focar apenas no Python, também é possível)**

Vamos começar dando alguns passos para trás para explicar alguns termos e paridades entre linguagens, passamos pelas motivações do ACL e do SQL, colocamos a mão na massa e fechamos o curso com uma **análise completa de folha de pagamento dentro do Python**.

# O que é uma linguagem de programação?

BLUFF: Para que possamos ter um entendimento mais profundo do mundo em que estamos entrando, é necessário estabeleçamos alguns conhecimentos básicos prévios. Se você já tem familiaridade com linguagens de programação o contexto aqui vai ser conhecido.

Para o slide, [The Most Popular Programming Languages - 1965/2021 - New Update - Statistics and Data](https://statisticsanddata.org/data/the-most-popular-programming-languages-1965-2021/)

## Visão geral sobre linguagens de programação

O termo **“linguagem de programação”** é utilizado de forma ampla dentro de quase qualquer contexto técnico hoje em dia, e para entender exatamente do que estamos falando é necessário que tenhamos alguma preparação prévia.

É difícil saber ao certo quantas linguagens de programação existem no mundo, o que podemos afirmar sem dúvida é que **existem linguagens que são mais utilizadas em determinados momentos da história**, variando de acordo com a necessidade e o que cada linguagem oferece. **Toda linguagem é boa em alguma coisa, mas nenhuma linguagem é boa em tudo, e é por isso que existem tantas.**

Aqui vai uma lista das linguagens mais populares em 2022:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Python | Java | JavaScript | C# | C |
| C++ | PHP | R | Objective-C | Swift |
| TypeScript | MatLab | Kotlin | Go (Golang) | VBA |
| Ruby | Scala | Visual Basic | Rust | Dart |
| Ada | Lua | Abap | Groovy | Perl |
| Cobol | Julia | Haskell | Objective Pascal | Elm |
| PowerShell | SQL | Clojure | Elixir | Pascal |
| LISP | Ballerina | FORTRAN | BASIC | Alice |
| COBOL | Speakeasy | Simula | Smalltalk | Prolog |
| Erlang | Ada | Eiffel | Rebol | Scratch |

Como você pode ver, são muitas linguagens e não, você não precisa se preocupar em aprender todas elas. Esse curso vai te ensinar Python que é provavelmente uma linguagem que ainda vai existir e ser suportada por muitas empresas por muitos anos. Nem mesmo grandes desenvolvedores conhecem muitas linguagens de programação, o importante é dominar os conceitos do mundo de programação.  
Mas se você quiser muito olhar para o futuro depois desse curso, recomendamos a linguagem Julia.

## Exemplos e utilidades

Lembrar de mostrar a foto do buraco negro

Aqui temos alguns exemplos para clarificar que **nenhuma linguagem boa para fazer tudo** e que existem nichos específicos para cada uma delas.

|  |  |
| --- | --- |
| JAVA, C#, C++, C, Objective Pascal, VBA | Criação de sistemas complexos de gestão corporativa (ERP), jogos iterativos, backend de sistemas, aplicativos diversos. |
| Objective-C, Swift, Java | Criação de aplicativos móveis para smartphones |
| JavaScript | Construção de frontend de aplicativos |
| **Python**, R, Matlab, Julia. | Massivamente utilizadas em análises de dados, das mais simples até as mais complexas. Papers científicos, modelos preditivos, aulas de matemática aplicada e até simulações de fluidos. Essas linguagens são muito conhecidas na academia, e recentemente tem tido muito protagonismo no mercado de trabalho. Essas linguagens também oferecem possibilidades de desenvolvimentos de outros tipos de sistemas, mas não são apropriadas para isso. |
| C, FORTRAN, COBOL, LUA, Rust | Manipulação de registros de sistema em baixíssimo nível, manipulação de memória de sistema em baixíssimo nível |
| SQL, **ACL**,Picalo, **Python**, R | Manipulação de dados em alto nível |
| Golang, Python, Rust, Java | Automatizações de processos de OPS em times de infra, |

Ainda existem as linguagens que chamamos de **linguagens de marcação, que é o caso do HTML**, que funcionam mais como uma **anotação do que deveria ser mostrado em lugares específicos** dentro do seu navegador. Esse tipo de linguagem não necessariamente instruções e especificas como **por exemplo “IF”**.

## Compilada VS Interpretada

Dentro de todas essas linguagens de programação, ainda sim podemos dividir elas em dois tipos as compiladas e as interpretadas.

* Compiladas: Quando você terminar um programa, é necessário que o computador traduza o seu código em linguagem de máquina
  + Ex: C++, C, Objective-C, Swift, Julia, FORTRAN
* Interpretada: O seu código é passado linha a linha por um interpretador de código que por sua vez conversa com o sistema operacional para executar o que é necessário.
  + Ex: **Python**, **ACL**, HCL, SQL

Existem várias motivações para que uma linguagem seja desenvolvida de uma forma ou de outra, mas em linhas gerais, aplicações de altíssima criticidade acabam sendo compiladas por uma questão de velocidade de resposta.

## Estruturadas, orientadas a objeto e funcionais

Ainda, independente de termos linguagens compiladas e interpretadas, existem 3 principais formas de estruturação de uma linguagem de programação. Sendo essas, a estruturada a orientada a objeto e a funcional.

### Linguagens estruturadas

As primeiras linguagens de programação que sugiram eram quase que uma conversa direta entre o programador e o hardware do computador. No passado, era comum que essas linguagens chegassem no nível de conseguir manipular a ausência ou presença de energia nos componentes internos do computador. Felizmente essa não e mais a realidade hoje, temos uma serie de camadas de abstração, que facilitam o desenvolvimento e manutenção de código.  
De qualquer forma, essas linguagens acabaram sendo encaixadas no paradigma de “linguagens estruturadas”, onde o desenvolvedor estrutura comandos com ênfase em sequência, decisão e, iteração. E e bom lembrar que não são só linguagens antigas que operam de forma estruturada.  
Exemplos de linguagens estruturadas: C, R, Basic, Fortran, Clipper, Pascal, Assembly, **ACL, Python...**

### Linguagens orientadas a objeto

Com a necessidade de organização de código e principalmente reutilização de código, um novo paradigma foi sugerido. Aqui temos uma fundamental diferença, onde encapsulamos código em um conceito chamado objeto. Um objeto pode conter tudo dentro dele, variáveis, tabelas, funções e até outros objetos.  
Durante muitos anos, o mercado fez um grande esforço para que tudo fosse orientado a objeto, o ponto e que existe uma medida de complexidade e performance quando pensamos em uma linguagem estruturada vs uma linguagem orientada a objeto, e por tanto um meio termo foi criado...  
Exemplos de linguagens orientadas a objeto: Java, C#, Objective-Pascal, VBA, **Python**...

### Linguagens funcionais

Podemos dizer que aqui temos um meio termo confortável entre os dois tipos de paradigma, que inclusive e seguido como padrão por várias linguagens de programação mais novas.  
Aqui temos uma forma de desenvolver o que precisamos com o nível de burocracia que quisermos, ou seja, caso queiramos estar mais próximo de linguagens estruturadas ou mais próximo de orientações a objeto e possível.  
Exemplos de linguagens funcionais: Go, Rusty, Julia, **Python**...

E sempre importante lembrar que não existe uma linguagem melhor ou pior, tudo depende do que queremos fazer e em qual contexto estamos trabalhando. No nosso contexto de analise de dados, e muito difícil que uma linguagem como o Java substitua o ACL ou o Python em algum futuro próximo.

Outro ponto interessante, e que o Python, consegue se encaixar perfeitamente em qualquer um dos paradigmas acima listados e essa e a primeira grande vantagem do Python, sua facilidade de se integrar em muitos cenários específicos.

## ACL & HCL? Python? SQL? Fórmulas do Excel

Esse capítulo tem como objetivo te mostrar as diferenças entre cada uma das linguagens que temos na nossa frente agora. A ideia e que esclareçamos possíveis diferenças que uma linguagem tem em cima da outra.

* ACL e HCL
  + Aqui temos um velho querido de auditores no mundo todo, o ACL, está a anos nos mercados de risco com o suporte da Dilligent e foi todo pensado para que seja uma linguagem simplificada.
  + O HCL e uma iniciativa da Dilligent de dar mais poder para analistas, além de se alinhar com as melhores práticas de mercado, a ideia aqui e trazer o Python para desenvolvedores ACL. Todo nosso curso vai ser em Python, mas o aprendizado vai ser quase que 100% traduzido para o HCL.
* Python
  + Desenvolvido no final dos anos 80/começo dos anos 90 na Holanda, o Python foi concebido para fazer interfaces com um sistema operacional muito específico da época. Hoje o Python e a principal linguagem de programação voltada para análise e manipulação de dados. Tido como um sucessor direto do R na academia, basicamente qualquer núcleo de análise de dados utiliza Python para alguma ou todas as atividades que exerce.
* SQL
  + Quando os bancos de dados começaram a se popularizar, foi tida a necessidade de que todos respondessem a comandos de forma padronizada e para isso foi-se proposto um padrão de linguagem que hoje conhecemos como SQL. Vale notar que apesar do SQL ser um padrão, cada banco de dados pode implementar comandos específicos dentro do seu contexto, e é por isso que muitas vezes algum comando mais avançado do Oracle não funciona no SQL-Server e vice-versa.
* Fórmulas do Excel
  + Comumente comparável ao ACL, ao Python e até ao SQL, todos que trabalharam com análises de volumetria média (mais de 1 milhão de linhas), sabe que o Excel não foi feito para trabalhar com dados em larga escala. É interessante lembrar que as versões mais novas do Excel te dão suporte para trabalhar com funções em Python, além de usar SQL para algumas funções especificas.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Excel (fórmulas) | SQL | ACL | HCL | Python |
| Tipo | Interpretada | Interpretada e compilada | Interpretada | Interpretada | Interpretada (Mas pode ser compilada) |
| Paradigma | Funcional | Estruturada | Estruturada | Estruturada, Orientada a objeto e funcional | Estruturada, Orientada a objeto e funcional |
| Gratuito | 🚫 | ✅ | 🚫 | 🚫 | ✅ |
| Windows | ✅ | ✅ | ✅ | ✅ | ✅ |
| Linux | 🚫 | ✅ | 🚫 | ⚠️ | ✅ |
| Mac | ✅ | ✅ | 🚫 | ⚠️ | ✅ |
| Open source | 🚫 | 🚫 | 🚫 | 🚫 | ✅ |
| Disco rígido para processamento | ✅ | ✅ | ✅ | ✅ | ✅ |
| RAM para processamento | 🚫 | 🚫 | 🚫 | ✅ | ✅ |
| GPU para processamento | 🚫 | 🚫 | 🚫 | ⚠️ | ✅ |
| Processamento batch | ⚠️ | ✅ | ✅ | ✅ | ✅ |
| Processamento inline | ⚠️ | ✅ | ⚠️ | ✅ | ✅ |
| Processamento assíncrono | 🚫 | ✅ | 🚫 | ⚠️ | ✅ |
| Portátil | 🚫 | ✅ | 🚫 | 🚫 | ✅ |
| Extensível | 🚫 | ⚠️ | 🚫 | 🚫 | ✅ |
| Incorporável | 🚫 | ✅ | 🚫 | 🚫 | ✅ |
| Gratuito | 🚫 | ⚠️ | 🚫 | 🚫 | ✅ |
| Comunidade | ✅ | ✅ | 🚫 | 🚫 | ✅ |

# E por qual motivo Python (e SQL)?

BLUFF: Aqui nos aprofundamos em entender por qual motivo esse curso foca no Python e como o mundo se relaciona com a linguagem. Também traçamos paralelos com o SQL pois essas duas linguagens existem juntas e se auxiliam em diversos momentos. A partir daqui, começa a sua jornada de imersão dentro do Python.

## Vamos entender o ecossistema Python

Interface gráfica do usuário

Descrição gerada automaticamente

Muito mais que uma linguagem de programação, hoje o Python é uma ferramenta utilizada em uma serie de casos e áreas de domínio específicos, que vão de desde criação de jogos, até mapeamento de astros no céu passando por computação cientifica de alto nível.   
É importante lembrar que a adoção em massa do Python por todas as grandes empresas de TI do mercado é uma realidade, por exemplo, a única outra linguagem que o Excel aceita na hora de se fazer funções, é o Python.

A Diligent, sempre atenta aos movimentos de mercado, também está adotando o Python. Esse movimento se iniciou no ACL Analytics 11, onde se era possível fazer chamadas externas a aplicativos Python e retornar o resultado diretamente em scripts e tabelas ACL.  
Abraçando e evoluindo ainda mais esse movimento, a Diligent, deu mais um passo nesse em encontro a essa mentalidade, o HCL e as funções Python embarcadas.

Esse curso não foca nas funções Python embarcadas na linguagem ACL.

O HCL é uma implementação otimizada do Python, feita pela Diligent. Então tudo que você conhece e aprender sobre Python vai ser 100% aplicável dentro do HCL, isso sem contar que as novas versões do ACL SERVIDOR te dão a possibilidade de agendar e executar scripts Python.  
Como o HCL é uma combinação do ACL que já conhecemos, e todo o poder que o Python oferece, aqui temos uma combinação poderosíssima, que te entrega o melhor dos dois mundos.

## Adoção do Python ao redor do mundo (e do tempo)

O objetivo aqui, é quantificar o quão grande é a popularidade do Python no mundo.

Não só dentro do universo de analise de dados, o Python é provavelmente uma das linguagens de programação mais famosas e utilizadas no mundo. Nesse primeiro gráfico, monitoramos as perguntas que são feitas na internet sobre linguagens de programação até o ano de 2018.

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

A plataforma monitorada é o stack overflow, eixo X é o ano e o eixo Y é o percentual de perguntas em relação ao todo.

Aqui, utilizando uma pesquisa feita pela RedMonk, podemos relacionar a popularidade das linguagens de programação no stack overflow, com a quantidade de projetos públicos cadastrados no git hub.

O Python se encontra no quadrante das linguagens mais populares.

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente  
Pesquisa feita em Jan-2022

É comum que autoridades em programação e sites especializados no assunto, apontem o Python como a linguagem que mais cresce no mercado.

Isso não quer dizer que o Python não possua problemas, mas quer dizer que você nunca estará sozinho e provavelmente qualquer problema que você encontre, alguém também já encontrou, resolveu e documentou em algum lugar.

## Passado, presente e futuro (Python, machine learning, R e Julia)

Como qualquer linguagem de programação, software ou processo, o Python não é perfeito.

Pelo fato de ser interpretado e ter sido concebido para outros propósitos que não eram necessariamente analise de dados, o Python tem problemas de performance quando falamos de dados realmente grandes (Petabytes, swap intenso de memória ou operações de calculo intensas).

E assim como a grande maioria dos estatísticos que hoje usam o Python hoje em dia no dia-a-dia e no passado usavam R, o futuro com certeza nos reserva uma mudança de linguagem novamente.

Não que você precise se preocupar com isso agora, o Python ainda tem uma grande vida útil pela frente, e vamos usa-lo bastante até que ele caia em desuso, mas é importante ficar antenado com o futuro.

Atualmente a maior aposta da comunidade de analise de dados, é que o Python vai ser substituído pela linguagem Julia. O Julia tem resolve todos os problemas de performance do Python, sem perder nenhuma das vantagens.

O panorama hoje é esse:

* Após um grande período onde as linguagens analíticas eram basicamente dominadas pelo R, ACL, MATLAB e Octave...
* O Python surgiu e está unificando o cenário de linguagens analíticas...
* Porém o Python possui alguns problemas de performance quando estamos lidando com problemas computacionalmente intensos e por tanto uma nova linguagem surge, a linguagem Julia, que parece ter tudo para ser a linguagem analítica mais usada no futuro.

Vale notar que tanto R, quanto Python e Julia, possuem ótimas bibliotecas para trabalhar com analise de dados e inteligência artificial e com o tipo de atividade que desempenhamos no mundo de auditoria e riscos qualquer uma das três linguagens funcionaria muito bem, o Python foi escolhido para esse curso pelo alinhamento de mercado e crescimento da utilização da linguagem.

# Primeiros passos com o Python

Esse capítulo foca em preparar o seu computador para ser completamente funcional com o Python.

* É imprescindível que você seja administrador local do seu computador, do contrário, teremos problemas de instalação
* Esse capítulo e curso, tem como premissa a utilização do Python no Windows. É possível utilizar outros sistemas operacionais, mas não é o foco desse curso.
* Para facilitar o aprendizado e diminuir o atrito com uma nova linguagem, utilizaremos um utilitário chamado Anaconda <https://www.anaconda.com/> .
  + Esse utilitário já possui todo o arcabouço de ferramentas e bibliotecas que precisamos para seguir com o nosso curso, assim evitamos perder tempo com instalações de arquivos menores e configurações de ambiente.
  + Caso você queira instalar o Python puro, existe um adendo no fim desse material com um passo-a-passo, mas os exemplos no curso foram feitos pensados em um computador com o utilitário Anaconda.

Os requisitos para essa instalação são:

* Sistema operacional: Windows 8.1/Windows 10/Windows 11
* Disco: 5GB disponível
* Ram: 8GB
* Processador: Dual core 1.8 Ghz ou superior
* Permissão de administrador local
* Navegador baseado no projeto Chromium ou Firefox
  + Chrome, Firefox, PaleMoon, Edge, etc.

## Baixando, instalando e configurando ambiente - Anaconda

Como esse pode ser o seu primeiro contato com o Python, vamos optar por utilizar um utilitário que configura tudo que você precisa com o mínimo de trabalhos extras e dores de cabeça.

* Quando você se sentir mais confortável com a linguagem e com o ambiente, é altamente recomendado que você utilize a linguagem sem nenhum utilitário. Se precisar de ajuda, nosso serviço de suporte ao cliente pode ter uma dica ou outra.

Para inicia;

* Navegue até o site anaconda: <https://www.anaconda.com/>

### Download

! A partir desse momento no curso, vamos começar a ver muitos passos em inglês !

O site , normalmente te entrega a última versão do instalador do Anaconda já adaptado para a versão do seu sistema operacional:  
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

De qualquer forma é sempre importante ter certeza, por tanto verifique a plataforma selecionada é o Windows.

Esse curso foi feito com a versão 2022.05 64 bits do Anaconda, caso queira utilizar exatamente a mesma versão, o site disponibiliza todos os instaladores de todas as versões que já foram lançadas na opção “Get aditional Installers”.

OBS: Esses links podem mudar de acordo com a interface do site que pode ser atualizada com o tempo, mas no geral, essa são funcionalidades que não costumam desaparecer.

Print da versão utilizada:



Como administrador do sistema, basta dar dois cliques no instalador.

### Instalando

O processo é relativamente simples;

1. Texto

   Descrição gerada automaticamente  
   Avance a instalação.
2. Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente  
   Leia com atenção o contrato de licença e caso esteja de acordo, aceite-o.
3. Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente  
   Recomendamos a instalação para todos os usuários da máquina, selecione a opção “All Users (requires admin privileges)”
4. Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente  
   Recomendamos a instalação na raiz do seu HD principal. Altere qualquer valor que venha preenchido para “C:\Anaconda3”.
5. Interface gráfica do usuário, Texto, Aplicativo

   Descrição gerada automaticamente  
   Caso seja perguntado se você quer registrar o Anaconda como a versão padrão do Python no Windows, confirme que sim.
6. Interface gráfica do usuário, Texto

   Descrição gerada automaticamente  
   Aguarde a instalação ser finalizada, ela pode demorar um pouco para finalizar (estamos instalando quase 4GB de informações).
7. Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente  
   Após a mensagem “Completed” aparecer, avance a instalação.
8. Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente  
   Avance a instalação.
9. Interface gráfica do usuário, Texto

   Descrição gerada automaticamente  
   Desmarque as opções de tutorial e finalize a instalação clicando no botão “Finish”.
10. Tela de celular com fundo preto

    Descrição gerada automaticamente  
    Para garantir que a instalação funcionou de acordo, verifique se a pasta do Anaconda3 foi criada no seu menu iniciar.

### Configurando e primeiros passos

Para iniciar, aperta a tecla da bandeira no seu teclado (ou procure aplicativos da forma que preferir), e digite Anaconda. O aplicativo que queremos iniciar é o Anaconda Navigator:  
Interface gráfica do usuário, Texto, Aplicativo, Teams

Descrição gerada automaticamente

Clique no software.

Algumas telas do prompt de comando podem aparecer por alguns instantes e você verá o mesmo ícone do software projetado no seu desktop por alguns instantes:

Uma imagem contendo Ícone

Descrição gerada automaticamente

Ignore qualquer aviso de upgrade de versão por enquanto.

A tela que você vai encontrar deve ser algo parecido com essa:  
Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Cada uma dessas caixinhas é um software diferente que é incluído com o Anaconda, todos são gratuitos e estão conectados com o universo de analise de dados de alguma forma.  
(Dependendo da versão do Anaconda esses softwares podem mudar, mas todos eles estão disponíveis gratuitamente separadamente nos sites dos seus respectivos desenvolvedores.)

Baseado na imagem da instalação:

* **CMD.exe Prompt** = Prompt de comando do Windows com o Python já habilitado
* **DataLore** = Aplicação da JetBrains que serve para ajudar a criar analises de dados.
* **IBM Watson Studio Cloud** = Ferramenta da IBM que ajuda na iteração com o Watson.
* **JupyterLab** = Versão mais poderosa e evolução do Jupyter Notebook, ambiente de analise de dados poderosíssima que aceita comandos no estilo “inLine”. (É aqui que nosso curso vai se focar).
* **Notebook** = Famoso Jupuyter Notebook, ambiente de desenvolvimento de análises que aceita comandos do tipo “inLine”.
* **Powershell Prompt:** Versão mais poderosa do prompt de comando, suportada pela Microsoft;
* **Qt Console:** Console habilitado para trabalhar com Python de forma “inLine”.
* **Spyder:** Ambiente de desenvolvimento Python, possui alguma semelhança com o ACL Desktop, mas é capaz de ser muito mais extensível e poderoso.
* **Glueviz:** Software de exploração de dados.
* **Orange 3:** Software baseado em componentes para data mining.
* **Pycharm Professional:** Ambiente de desenvolvimento Python utilizado pelas maiores empresas do mercado para desenvolvimento de aplicações profissionais com Python. Não é gratuito para grandes empresas, apenas usuários individuais com fins de estudo.
  + **Microsoft Visual Code:** Caso tenha se interessado pelo Pycharm, o Microsoft Visual Code é uma versão baseada em componentes menos burocrática.
* **RStudio:** Ficou curioso para entender como é o R? O Rstudio é o ambiente mais comum para desenvolvimento de scripts R.

Ainda, existem outras opções dentro do Anaconda Navigator, vamos explorar cada uma delas:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

A opção de home, é onde temos todos os softwares que são instalados por padrão dentro do Anaconda Navigator:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* De acordo com a versão do Anaconda navigator, novos softwares podem acabar sendo adicionados aqui.
  + Alguns softwares podem acabar sendo retirados também, embora seja muito mais raro.
* No geral, todos os softwares aqui listados têm como objetivo serem gratuitos, mas em alguns casos específicos podemos ter softwares que tem uma parte paga;
  + O foco do time Anaconda, é que os softwares sejam gratuitos principalmente para estudantes e curiosos e embora a maioria dos softwares sejam totalmente gratuitos, vamos ter alguns casos como o IBM Watson Studio Cloud, que possuem níveis de gratuidade;
* O Anaconda navigator sempre inicia na opção de home;
* Todos os softwares possuem uma engrenagem no seu nome, essa engrenagem permite que você atualize ou desinstale aplicações especificas.

A opção “Environments” é onde administramos o nosso ambiente python, todos os pacotes e aplicações que ali estão instalados. Também é possível a criação de outros ambientes para aplicações especificas, bem como o backup e importação de outros ambientes:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

* A função de backup de ambientes pode ser muito interessante caso você queira ter um mesmo ambiente padronizado entre todos os membros do seu time.
* É aqui que faremos a instalação de novos pacotes caso queiramos.
  + Podemos atualizar pacotes específicos também.
* É importante ter em mente que existem milhares de pacotes python, um utilitário como o Anaconda facilita muito encontrar e instalar esses pacotes.

A opção “Learning” traz documentações e treinamentos completamente gratuitos sobre tudo que gira em torno do universo Anaconda:

Uma imagem contendo Calendário

Descrição gerada automaticamente

* O conteúdo aqui, também pode mudar de acordo com o tempo, a comunidade sempre atualiza esses pacotes e procura atualizar as documentações com bastante frequência.
* Algumas das documentações aqui podem acabar tendo um nível técnico bem elevado.
* Todas as leituras são gratuitas.

A opção “Community” tem o link para fóruns muito interessantes que orbitam em volta do mundo de analise de dados:

Linha do tempo

Descrição gerada automaticamente

* Os links também podem variar de acordo com a versão do Anaconda.
* Todos os fóruns têm centenas de pessoas colaborando ativamente todos os dias, vale bastante conhecer.
* Todos os fóruns são gratuitos.

## Se ambientando com a IDE

Mas o que é uma IDE?

A sigla significa “integrated development environment” que em uma tradução livre seria algo como “Ambiente de desenvolvimento integrado”.  
E aqui começa a nossa primeira grande diferença prática entre Python e ACL.

O ACL é uma linguagem de programação e o ambiente de desenvolvimento do ACL também se chama ACL, e tudo é uma coisa só.  
O Python é uma linguagem de programação e tem muitos ambientes de desenvolvimento, o python inclusive pode ser programado sem nenhum ambiente de desenvolvimento se assim você quiser.

Nosso curso vai focar em uma IDE chamada “Jupyter Lab” que é um ambiente que roda no seu navegador de internet e aceita códigos Python no estilo “inLine” que significa que lá vamos digitar códigos e testar conforme vamos construindo nossa análise sem ter a necessidade de escrever todo o código e só depois testar.

Não se preocupe com isso por enquanto, mas aqui temos algumas IDE’s que suportam o Python:

* Pycharm
  + Professional e Community
* Spyder
* Jupyter Notebook
  + Toda a família Jupyter e extensões do mesmo
* Microsoft VScode
* Microsoft Visual Studio
* Sublime
* Notepad++
* IDLE
* Atom
* Eclipse + PyDev
* Thonny
* Wing
* VI / VIM
* GNU Emacs
* ...

Mas a nossa IDE para esse curso vai ser o **Jupyter Lab**.

No seu anaconda Navigator você vai encontrar o ícone do Jupyter Lab e vai clicar no botão “Launch”:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

O seu navegador vai abrir com uma tela que deve ser parecida com essa:  
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Repare que o endereço da página é algo como <http://localhost:8888/lab>, e isso acontece pois você está rodando um servidor Jupyter dentro da sua máquina.

A estrutura de pastas na esquerda, representa a estrutura de pastas do seu usuário dentro do computador que você está usando. Essa estrutura normalmente é C:\Users\<Seu usuário>, mas isso pode ser um pouco diferente dependendo da instalação padrão do Windows na sua companhia ou computador pessoal.

Agora que estamos aqui, tudo que precisamos para começar é a criação de uma pasta de análises que é onde vamos fazer todo nosso curso. Para tal, clique com o botão direito em uma região em branco embaixo da árvore de pastas e clique em “New Folder”:  
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Sugerimos que o nome da pasta seja “Diligent Python”, para facilitar o acompanhamento junto com o material aqui descrito.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Dê um duplo clique para entrar na pasta.

Dentro dos materiais do curso, temos 3 arquivos que vamos utilizar:

Texto

Descrição gerada automaticamente

Selecione os 3 e mova para dentro da pasta que acabamos de criar

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

O seu resultado deve ser parecido com o que temos na imagem acima.

Isso feito clique no botão abaixo do título notebook, escrito Python 3 (ipykernel):

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Você acabou de criar seu primeiro notebook e é aqui que grande parte do nosso curso vai acontecer.

Repare que na arvore de pastas que você tem um novo arquivo, “Untitled.ipynb”:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Clique com o botão direito e renomeie ele para “Meu primeiro notebook.ipynb”:  
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

O notebook funciona com células, e você já pode ver uma logo de cara dentro do seu notebook. Cada célula pode ter um ou mais comandos python:  
Tabela

Descrição gerada automaticamente com confiança baixa

Para começar vamos escrever nosso primeiro comando Python, tente escrever print(“Seu nome”) e apertar CTRL + Enter:  
Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Você acabou de executar o seu primeiro comando Python!

1. Como dito, uma célula pode ter mais de um comando, tente escrever outros comandos em outras linhas e execute (CTRL + Enter):  
   Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente
2. Agora, vamos criar uma nova célula e executar comandos em células diferentes. Para isso basta clicar no botão “+” na parte de cima do seu notebook:  
   Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente
3. Digite outro comando dentro da nova célula e execute novamente. Você pode imprimir frases com espaço também:  
   Interface gráfica do usuário, Texto, Aplicativo

   Descrição gerada automaticamente
4. As células também podem servir como células de documentação, crie uma nova célula (como no passo 2), clique nos colchetes no começo da célula, depois no menu na parte superior do notebook procure o menu “dropdown” que está escrito “code”, e mude para “markdown”:  
   Interface gráfica do usuário, Texto, Aplicativo

   Descrição gerada automaticamente  
   Repare que os colchetes da célula sumiram. Tente digitar o seguinte comando “<h1> Meu primeiro titulo em markdown” sem as aspas e execute:  
   Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente  
   Todo markdown pode ser alterado, basta clicar duas vezes nele e alterar o código. O Markdown é uma linguagem própria que apesar de simples, serve para documentar nossos projetos e processos.
5. Tente chegar em um resultado parecido sozinho:  
   Interface gráfica do usuário, Texto, Aplicativo, Email

   Descrição gerada automaticamente

Aqui tem um guia rápido de todas as funções que o notebook possui, tente se ambientar um pouco com a IDE.

* Shift + Enter = Executa a ou as células selecionadas. Caso não exista uma célula embaixo, adiciona uma.
* Ctrl + Enter = Executa a ou as células selecionadas.
* Ctrl + S = Salva seu trabalho
* Ctrl + Shift + S = Salva o seu trabalho te dando mais opções de onde
* Ctrl + F = Encontra algo nas células
* Ctrl + Z = Desfaz a deleção ou adição de células
* DD = (Apertar a tecla D duas vezes) Deleta uma célula

## Variáveis, funções, objetos, chamadas e outros...

Vamos colocar a mão na massa, nessa sessão começamos a codificar propriamente em python.

Para começar temos que estabelecer alguns conceitos simples:

* Sintaxe = É a forma que uma linguagem de programação estrutura seus comandos. A sintaxe do Python é diferente da do ACL, e é por isso que os comandos não são os mesmos.
* Variável = É um espaço em memória nomeado onde guardamos uma informação. Qualquer informação;
* Funções = Chamamos de função, todo aquele código que é guardado de forma estática e recebe parâmetros para fazer cálculos imutáveis em sua maioria das vezes. Pode conter uma ou mais variáveis;
* Objetos = Quando fazemos cópias de grandes pedaços de códigos com partes especificas. Pode conter funções, variáveis e outros;
* Chamadas = É quando executamos algum comando, ou função ou qualquer coisa que dependa de algum código ser executado;

### Preparando o seu primeiro notebook para variáveis

Para iniciar esse passo do curso, dentro do JupyterLab, crie um novo notebook com o nome “01\_Primeiros passos”:  
Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

É de bom costume, iniciar todo notebook com um titulo e uma descrição do que faremos. Utilize o que você aprendeu de documentação, adicione um titulo ( <h1> ) em uma célula e na outra uma descrição criativa sobre aprender os primeiros passos em Python.   
(Caso precise pular linha na sua descrição, use a TAG <BR>)

Caso precise de inspiração, o notebook oficial do curso ficou assim:  
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Em uma nova célula, digite o seguinte comando, e execute:  
Interface gráfica do usuário, Aplicativo, Site

Descrição gerada automaticamente

Você acabou de guardar a sua primeira variável. E dentro dela você guardou o valor “Python Diligent”

Dentro do python para exibir qualquer coisa, basta você reescrever o nome da variável, ou colocar ela em um comando print(). Veja os exemplos:  
Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
(Não se esqueça de executar as células depois de digitar o comando)

Em uma nova célula, digite os seguintes comandos:  
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Linha a linha o que fizemos:

* variável\_02 = Estamos apenas recebendo um novo valor em texto
* variável\_03 = Estamos recebendo um novo tipo de valor, um numero!
  + Percebe que aqui não temos as aspas? Números dentro do python são informados de forma livre
* variável\_04 = Aqui já começamos a fazer operações com variáveis. Essa variável está recebendo a anterior duas vezes.
* variável\_05 = Aqui também estamos fazendo uma operação matemática, só que entre duas frases.
  + O python tem um operador de textos excelente que nos permite fazer uma serie de operações bem legais.
  + É só importante notar que dependendo da sua situação, as operações também podem ficar complexas.

Execute o seguinte comando logo após:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

O python, também permite que comentemos nossas linhas de código com o #:

Texto

Descrição gerada automaticamente

E você também pode agrupar comandos em células e executa-los de uma vez só:

Texto

Descrição gerada automaticamente

Uma das linhas de base de criação da linguagem, foi que ela fosse organizada e de fácil leitura, mas não existe linguagem que seja limpa e organizada caso não colaboremos. Com os novos aprendizados que você teve, reorganize seu notebook para que ele fique limpo e fácil de se entender.  
Utilize comentários, tags de documentação, adicione células, remova células e agrupe da forma que você achar melhor e mais organizado.  
Se precisar de inspiração, o notebook oficial do curso ficou assim:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Você também pode fazer print das suas variáveis de forma linear:

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

Você também pode colocar outros textos dentro do seu print:

Texto

Descrição gerada automaticamente

Veja que no ultimo exemplo vimos duas técnicas novas:

* \n dentro de uma string, serve para pular uma linha
* Você pode pular linhas dentro de comandos Python, desde que as coisas continuem fazendo sentido dentro de um caminho lógico.

A criação e manutenção de variáveis dentro do Python é muito simples, vamos ver como funcionam operações matemáticas simples.

Vamos entrar em um novo tópico, e é importantíssimo que o seu notebook esteja organizado e limpo, faça questão de ter o seu trabalho bem documentado e descrito começando por um título, veja o meu como ficou:  
Texto

Descrição gerada automaticamente  
(Utilizei a tag <H2> no titulo)

Os criadores do Python, decidiram que as operações matemáticas deveriam ser exatamente como são na vida real. Por tanto as operações são muito simples e coezas. Veja os seguintes exemplos  
Texto

Descrição gerada automaticamente

O Python segue a ordem algébrica na hora de resolver operadores e isso quer dizer que; multiplicações e divisões são sempre realizadas antes de somas e subtrações. Utilize o “()” caso precise indicar orde:

Interface gráfica do usuário, Texto, Aplicativo

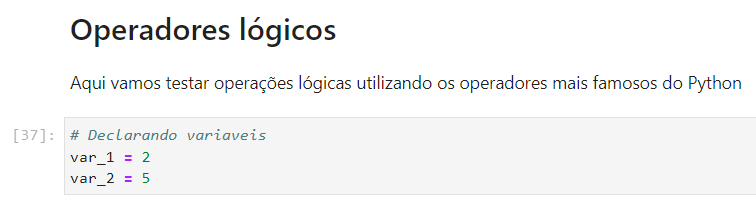
Descrição gerada automaticamente

### Preparando o seu primeiro notebook para laços de repetição e operadores lógicos

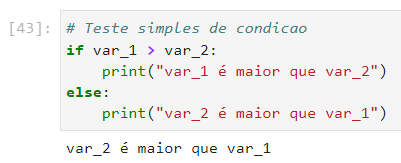
Assim como a grande maioria das linguagens de programação, o Python possui laços de repetição, os mais famosos são:

* IF = Testa condições;
* FOR = Itera sobre algum contável até chegar no seu fim;
* WHILE = Executa uma série de códigos até que uma condição deixe de ser verdade

(Não se esqueça de documentar e explicar o seu notebook)

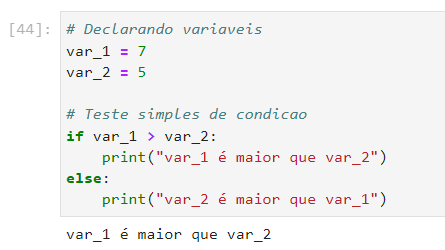
Vamos começar com o comando if utilizando algumas variáveis. Primeiro declare duas variáveis:  


Vamos testar uma condição simples:



Aqui é importante reparar que, ao contrário do ACL, a endentação do python é muito importante. O Python só entende que um comando vai ser executado dentro de um espaço de condição, caso ele tenha o tamanho de “1 TAB” e esteja imediatamente embaixo do comando de IF.

Tente alterar as variáveis para ver que a saída vai ser diferente:



Resumindo, o IF precisa ser escrito da seguinte forma:

* Inicia-se com IF;
* Condição com uma resolução binária;
  + Ou isso, ou aquilo;
* Dois pontos;
* Comandos que vão ser executados caso a condição seja verdadeira;
  + Esses comandos precisam ter 1 TAB de distancia e estar imediatamente abaixo do IF com a condição;
  + Podemos ter quantos comandos quisermos;
* De volta na mesma posição, o else;
* Dois pontos;
  + Comandos que vão ser executados caso a condição original seja falsa;
  + Podemos ter quantos comandos quisermos;

Veja o exemplo:

Texto

Descrição gerada automaticamente

O Python sabe que não tem mais comandos para serem executados quando não existirem mais comandos com a distância de um TAB

Você também pode encadear IF’s para ter respostas mais complexas quando você precisar. Algo como:

Texto

Descrição gerada automaticamente com confiança média

O exemplo utiliza a palavra TAB para exemplificar que devemos ter a distancia de 1 TAB, mas no código você não pode colocar o TAB e sim os espaços.

O for também é um comando muito interessante, que permite iterar sobre qualquer coisa que seja contável dentro do Python. E isso pode ser o numero de linhas/colunas de uma tabela, a quantidade de arquivos a se processar ou simplesmente um numero que vai funcionar como um contador.

Para tal, declare uma variável que vai receber a quantidade de vezes que vamos executar uma série de comandos:

Uma imagem contendo Texto

Descrição gerada automaticamente

Vejamos como um for simples funciona:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Aqui temos uma série de conceitos novos.

* Você reparou que não declaramos nenhuma variável “i” em lugar nenhum? O python entende que você precisa de uma variável ali e cria ela de acordo com a necessidade.
* Temos um novo comando, o range(). O range é responsável por retornar o tamanho de iterações que uma variável tem, no nosso caso, a variável contagem é uma iteração de números inteiros. Para ser mais exato uma iteração de 5 vezes um numero inteiro.
* E depois usamos um conhecido nosso, o print, que só é responsável por exibir o valor de “i” a cada interação.

Resumindo, nosso comando está fazendo :

* Na primeira linha
  + Crie uma variável
  + Que vai ser incrementada o mesmo tanto de vezes que o tamanho total da variável fim
* Na segunda linha
  + Exiba o valor de “i” a cada iteração

(Vale notar que a regra dos 1 espaços de TAB também vale aqui)

Vamos tentar um comando mais complexo.

Texto

Descrição gerada automaticamente

Você reparou que apesar da variável fim ter valor 5, nosso laço termina no 4?  
Isso está certo e acontece porque o computador entende que o primeiro número é o 0.  
Se iterarmos 5 vezes a partir do 0, o resultado vai ser 4

Resumindo, o FOR precisa ser escrito da seguinte forma:

* Inicia-se com for;
* Condição de iteração;
  + Não necessariamente uma contagem numérica, existem várias formas de iterar no python;
  + No nosso caso usamos o range() que nos retorna a quantidade de posições que uma variável tem;
* Dois pontos;
* Comandos que vão ser executados caso a condição seja verdadeira;
  + Esses comandos precisam ter 1 TAB de distância e estar imediatamente abaixo do IF com a condição;
  + Podemos ter quantos comandos quisermos;

Você também pode misturar operadores de repetição:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Aqui estamos testando com uma condição if, se a posição de ‘i’ é menor ou maior que 3 e exibindo textos diferentes de acordo com o resultado.

O while é o operador de repetição que vai funcionar para sempre até existir uma condição de parada.

Vamos declarar uma variável de parada para entender o funcionamento do while:



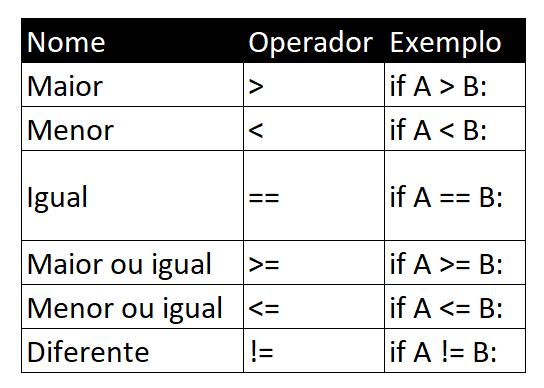
Como declarar um while():

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

No geral o resultado parece muito com o do for, certo? E ele é mesmo. Pense no for como uma forma mais automatizada de ser fazer um while. Embora o while tenha condições especificas de uso e por isso os dois existem dentro da linguagem.

Os operadores lógicos, são os caracteres que usamos para criar condições.

Veja a tabela:  


* Repare que quando queremos verificar se algo é igual, usamos dois sinais de igual;
  + Só um sinal de igual, significa que queremos atribuir um valor a uma variável, por isso usamos duas;

### Preparando o seu primeiro notebook para funções

Agora que você já conhece o básico sobre variáveis, vamos aprender o básico sobre funções.  
(A partir desse momento, assumimos que você já sabe criar novas células e que todo código vai ser documentado e explicado)

Uma função dentro do python precisa obedecer a seguinte ordem:

* O comando def
* Nome da função
* Entre parênteses quais parâmetros ela recebe
* Dois pontos

Algo como:



Todos os comandos que aprendemos até agora podem ser usados dentro de uma função e tudo que vai acontecer dentro da função precisa estar indentado com um tab de distância:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Tente escrever e executar essa função e executar a célula, você verá que o python não vai executar nada. O que está acontecendo na verdade é que o Python entendeu, leu e guardou essa função para ser usada. A partir de agora, essa função existe dentro do contexto desse notebook.  
Agora execute a sua função da seguinte maneira:

Texto

Descrição gerada automaticamente com confiança média

Qualquer função que você criar vai passar a existir enquanto seu console conda estiver funcionando. Se você desligar o Python, ou o computador, você precisará “ensinar” novamente o python essa função. Isso não é um problema, uma vez que podemos deixar essas funções previamente criadas.

Agora vamos tentar criar uma função usando os conceitos que aprendemos de laços de repetição:

Diagrama

Descrição gerada automaticamente com confiança baixa  
Execute essa célula e veja que nada acontece, exatamente como a outra função que fizemos.

Execute a função que você criou:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Um outro conceito que precisamos aprender é o retorno de funções. Em muitos casos é importante que nossas funções retornem valores e não só os imprimam. Para isso existe o comando return, onde retornamos valores para uma variável que precisa estar esperando uma resposta de uma função:

Texto

Descrição gerada automaticamente

A diferença parece sutil, mas ao invés de usarmos o comando print e perder aquela informação, estamos retornando para que o usuário possa utiliza-la lugares que faça sentido.

E a utilização é bem simples:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Imagine que tendo esse tipo de funcionalidade, podemos encadear funções e fazer soluções bem complexas e completas:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Como desafio, tente fazer uma função que:

1. Faz uma contagem de X a Y;
2. Sempre que o número for par, exibe na tela;
3. Sempre que o número for ímpar mostra um traço;
4. Retorna para uma variável True se o último número for par, se o número for ímpar retorna False;

Esse exercício também é bom para você exercitar o poder da comunidade Python!

Como recomendação

* Procure pelo operador de “resto de uma divisão” para descobrir o que é um número par ou ímpar;
* Também procure pelo tipo de variável Boolean, pode ajudar com a parte do “True e False”;

O notebook oficial do curso tem essa função pronta e explicada.

## Trabalhando com tabelas dentro do Python

Para iniciar esse passo do curso, dentro do JupyterLab, crie um novo notebook com o nome “02\_Trabalhando\_com\_tabelas”:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Não se esqueça de documentar o seu trabalho com as tags de markdown:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Para iniciarmos esse capítulo é importante deixar claro a importância da comunidade Python. O problema de se trabalhar com os objetos do tipo tabela é algo que todos os analistas de dados já passaram ou precisam passar.  
Alguém, que em algum momento passou por esse problema o resolveu e expôs isso para a comunidade, muitas outras pessoas usaram o trabalho e iteraram em cima disso, dessa iteração que acontece até hoje, nasceu o pacote (conjunto de funções, objetos e classes) “Pandas” que é o que usaremos para trabalhar com tabelas:

[Pandas - Python Data Analysis Library (pydata.org)](https://pandas.pydata.org/)

Ícone

Descrição gerada automaticamente com confiança média

Para começarmos a usar o Pandas, você primeiro precisa avisar o Python que vamos usa-lo:

Texto

Descrição gerada automaticamente com confiança média

Repare que importamos o pacote pandas e damos um apelido para ele, nesse caso “pd”, esse é o apelido mais frequente usado na comunidade.

Para iniciarmos a manipulação da tabelas, vamos fazer da forma mais simples possível primeiro e depois vamos deixando mais complexo conforme progredimos.

Para iniciar, vamos importar o arquivo “Departamento.txt”:



Veja que o pandas é relativamente esperto na hora de importar um arquivo, mas isso não quer dizer que ele não tenha vários parâmetros que conseguimos usar para fazer importações mais complexas e complicadas. Outro ponto é que nem sempre o Pandas faz tudo como deveria, é sempre bom fazer tudo com bastante cuidado.

Para ver o conteúdo dessa tabela, basta usar o comando “.head()”:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Como você pode ver o pandas achou que a primeira linha do arquivo era o cabeçalho. Isso está errado, para os comandos básicos do pandas podemos seguir assim, mas isso reforça a necessidade de sempre olharmos os dados que estamos trabalhando.

Para vermos o final da tabela, utilizamos o comando “.tail()”:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Se quisermos ver registros específicos da tabela, podemos utilizar o “.iloc”:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

(Nesse caso forcei mostrar os registros de 1 a 3, ou seja o primeiro e o segundo)

Podemos visualizar apenas registros de uma coluna especifica:

Texto

Descrição gerada automaticamente

Vamos iniciar fazendo o import de todas as tabelas que temos no nosso projeto de forma correta:

Texto

Descrição gerada automaticamente

Repare que aqui utilizamos alguns conceitos novos:

* Parâmetros do read\_csv, onde indicamos que não temos cabeçalho de dados e damos nomes das colunas
* Utilizamos uma nova função do pandas, o read\_excel, que serve para ler arquivos XLS e XLSX.
  + Caso você tenha planilhas com muitas funções e formatações, podemos ter problemas.

Verifique se os dados foram importados com sucesso, utilize o comando “.head()” para ter uma ideia.

Tabela

Descrição gerada automaticamenteInterface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Parar criar campos calculados simples:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Agora que você já entendeu a dinâmica básica com tabelas Python, vamos começar a trabalhar com comandos mais complexos.

Para continuar vamos fazer a importação de todos os arquivos que temos:

Graphical user interface, text, application

Description automatically generated

Não se esqueça de verificar a consistência dos dados com o comando .head(), para cada um dos arquivos importados.

### Operações em tabelas

Primeiramente vamos trabalhar com um conceito novo, para evitar alterar algum dado sensível nas tabelas que importamos, vamos criar uma copia do dataframe.  
Table

Description automatically generated

Não se esqueça de verificar a consistência dos dados que você copiou.

Mais uma vez vamos fazer algumas operações de campo calculado:

Text

Description automatically generated with low confidence

No nosso notebook, fizemos várias verificações de consistência, veja o exemplo no notebook: 02\_trabalhando\_com\_tabelas.ipynb

No notebook utilizamos um exemplo prático de groupby também:

A picture containing text

Description automatically generated

Também criamos uma rotina para salvar resultados:

Graphical user interface, text, application, email

Description automatically generated

### Amostragem

O Python oferece uma quantidade enorme de formas estatisticas de se fazer amostragem. Aqui vamos focar nas mais simples e reprodutíveis, o comando dataframe.**sample**()

Table

Description automatically generated

Aqui podemos ver como utilizar amostragem com reposição e sem:  
Text

Description automatically generated  
E aqui também podemos ver como usar uma semente de reprodutibilidade para reproduzir experimentos:  
Graphical user interface, text

Description automatically generated

### Joins e agregações

Em qualquer rotina de análise de dados, fazer agregações de tabelas e necessário e corriqueiro. Aqui no python, podemos fazer uma agregação através dos comandos join e merge. No nosso curso, vamos estudar o comando merge.

Texto

Descrição gerada automaticamente

Veja o notebook para referência, lá temos exemplos prontos para usar com as tabelas que usamos durante o curso da apostila.

### Concat

O comando **dataframe.concat()** e utilizado para fazer a operação de **append** dentro do Python. Assim como qualquer outro comando, existem muitas opções de parâmetros para o comando concat(), mas a regra de ouro e garantir que as duas tabelas que queremos usar tenham exatamente o mesmo formato.



Veja o notebook do curso para entender na pratica um exemplo.

## Minha primeira análise de dados

Esta sessão tenta criar uma situação que é passada no dia a dia do analista de dados, vamos utilizar o Python para fazer um auditoria de dados.

Sugerimos que você construa sua análise através dos seguintes níveis de maturidade:

* Planejamento -> Onde entendemos os requisitos, e verificamos o que é possível e o que não e;
* Acesso aos dados -> Onde verificamos os dados disponíveis,
* Análise de integridade/consistência -> Onde verificamos se os dados disponíveis são de fato bons o suficiente para trabalharmos em cima;
* Análise -> Onde desenvolvemos a análise em si;
* Apresentação -> Que e onde emitimos nosso relatório de auditoria;

### Sobre o exercício:

As seguintes informações de contexto sobre a nossa empresa e o contexto da nossa analise foram passadas:

1. Nosso time de TI, nos informou que todos os arquivos que recebemos estao na pasta '.../Data' mas que eles nao foram validados;
2. Existem 3 tabelas (departamento, funcionarios e folha), que consistem todo o universo de dados que possuimos para essa analise;
3. Entendemos que seria muito interessante entender outras metricas relacionadas aos dados que temos, conseguimos trazer algum insight?
4. Nosso time de TI, tambem nos informou que e esperado que o nosso resultado seja salvo na pasta '.../Resultados';
5. A analise nesse notebook precisa ficar limpa e concisa para que qualquer um no futuro possa entender o que foi feito;

### Suspeitas e Insights

Essa auditoria foi encomendada com as seguintes suspeitas:

* Existem funcionários recebendo pagamentos dobrados (no mesmo mês, sendo pagos duas vezes);
* Existem funcionários que não estão sendo pagos devidamente (pulando meses de pagamento);
* Existem funcionários sem valor registrado no pagamento;
* Não se sabe se as contas de salário bruto e salario liquido batem;

Conversando com o time de auditoria, identificamos alguns pontos que seriam interessantes para investigarmos:

* Funcionários que ganham muito acima ou abaixo da média;
* Departamentos com salários médios muito acima ou abaixo da média;
* O gasto mensal da empresa com folha de pagamento;
* Média de salário por cargo;
* Existe algum departamento sem funcionários?

Não deixe de planejar sua análise:

**Para realizarmos essa análise precisamos nos certificar que as tabelas existem e podem ser consumidas:**  
-> Va até a pasta Data e verifique se os arquivos que estão lá, podem ser usados e estão em formatos que você conhece.

**É importante verificar se a tabela resultados existe:**  
-> Veja se você já tem uma pasta chamada Resultados, se não, crie uma.

**Também é importante logar quando, por quem e onde esse trabalho está sendo realizado:**  
-> Não esqueça de documentar essas informações no notebook, elas vão ser muito importante para o futuro.

**Defina o objetivo final da sua análise de forma clara e concisa:**  
-> O que você quer resolver? Não deixe de escrever e documentar no notebook da análise.

**Defina um cabeçalho para análise:**  
-> Parte da documentação, e deixar claro o objetivo da análise, quem fez e até especificações técnicas do computador que rodou a análise.

Utilize o notebook 03\_Estruturando\_uma\_análise – Exercício para confeccionar sua primeira análise. E esperado que você consiga chegar até o final da analise com os comandos que foram aprendidos até aqui, mas também e esperado que você exercite a sua capacidade de procurar alternativas junto a comunidade Python na internet.

Tente utilizar comando gráficos ou até análises estatísticas diversas.

Algumas bibliotecas que podem ser interessantes, e podem te ajudar a encontrar resultados interessantes são:

* seaborn: Biblioteca gráfica com uma serie de gráficos pré-prontos que ajudam bastante na análise de dados -> <https://seaborn.pydata.org/>
* Plotly: Mais uma biblioteca gráfica, mais poderosa que a anterior, porém mais trabalhosa de se usar -> <https://plotly.com/>
* Scikit-learn: A biblioteca mais popular de machine learning e testes estatísticos. Ainda e muito cedo para utilizarmos algoritmos de aprendizado de máquina, mas já podemos aproveitar algumas coisas aqui -> <https://scikit-learn.org/stable/>