



targettrust⁺
treinamento e tecnologia

Valli (tiago vallinoto)

Especialista de TI - Sicredi

Produtos, Agilidade e Dados

Atuo em movimentos de transformação
digital e mudança de cultura
~20 anos em tech / ~5 dados

Como falar comigo:

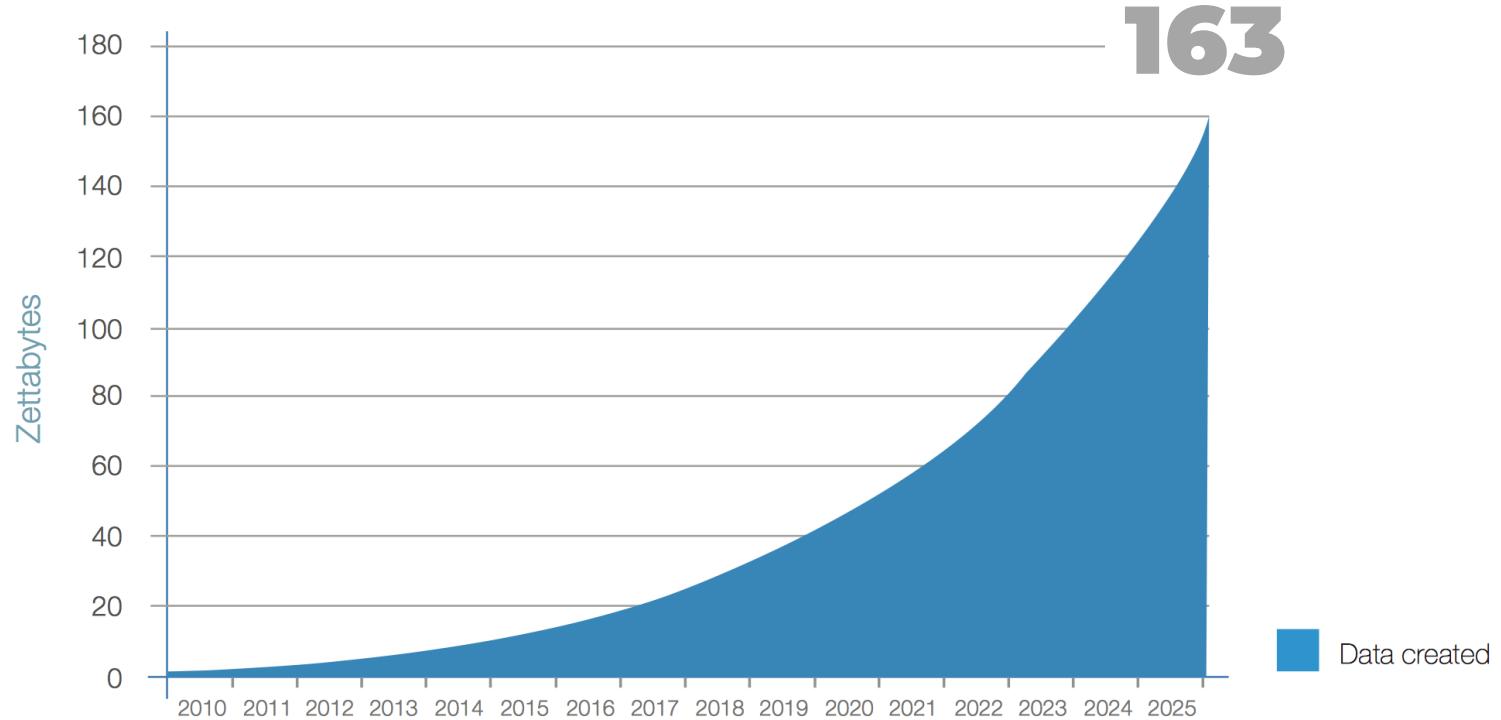
- tiago@targettrust.com.br
- linkedin.com/in/vallinoto
- <https://vallinoto.com>





O mercado hoje

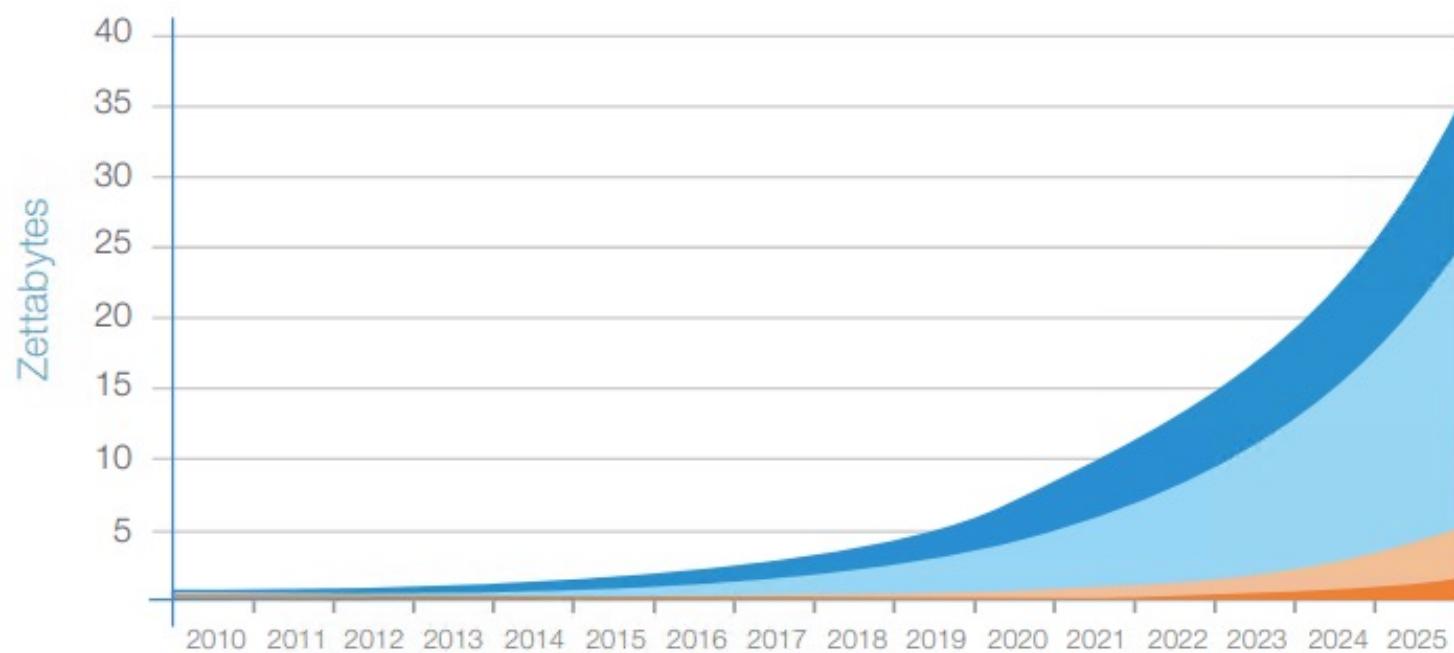
Estamos gerando cada vez + dados



Source: IDC's Data Age 2025 study, sponsored by Seagate, April 2017

<https://www.import.io/wp-content/uploads/2017/04/Seagate-WP-DataAge2025-March-2017.pdf>

Estamos gerando cada vez + dados



Source: IDC's Data Age 2025 study, sponsored by Seagate, April 2017

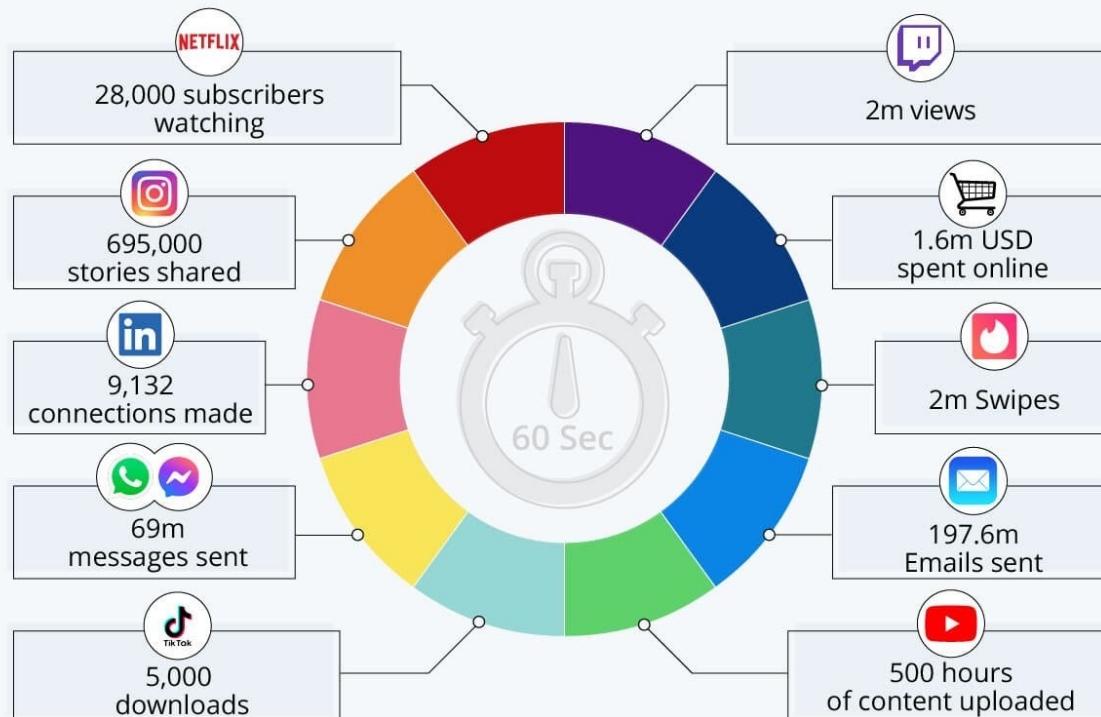
O IDC estima que até o final de 2025, apenas 15% dos dados na esfera de dados global será marcada e apenas um quinto disso será realmente ser analisado.

- Useful if Tagged
- Tagged
- Analyzed
- Touched by Cognitive Analytics

Estamos gerando cada vez + dados

A Minute on the Internet in 2021

Estimated amount of data created
on the internet in one minute



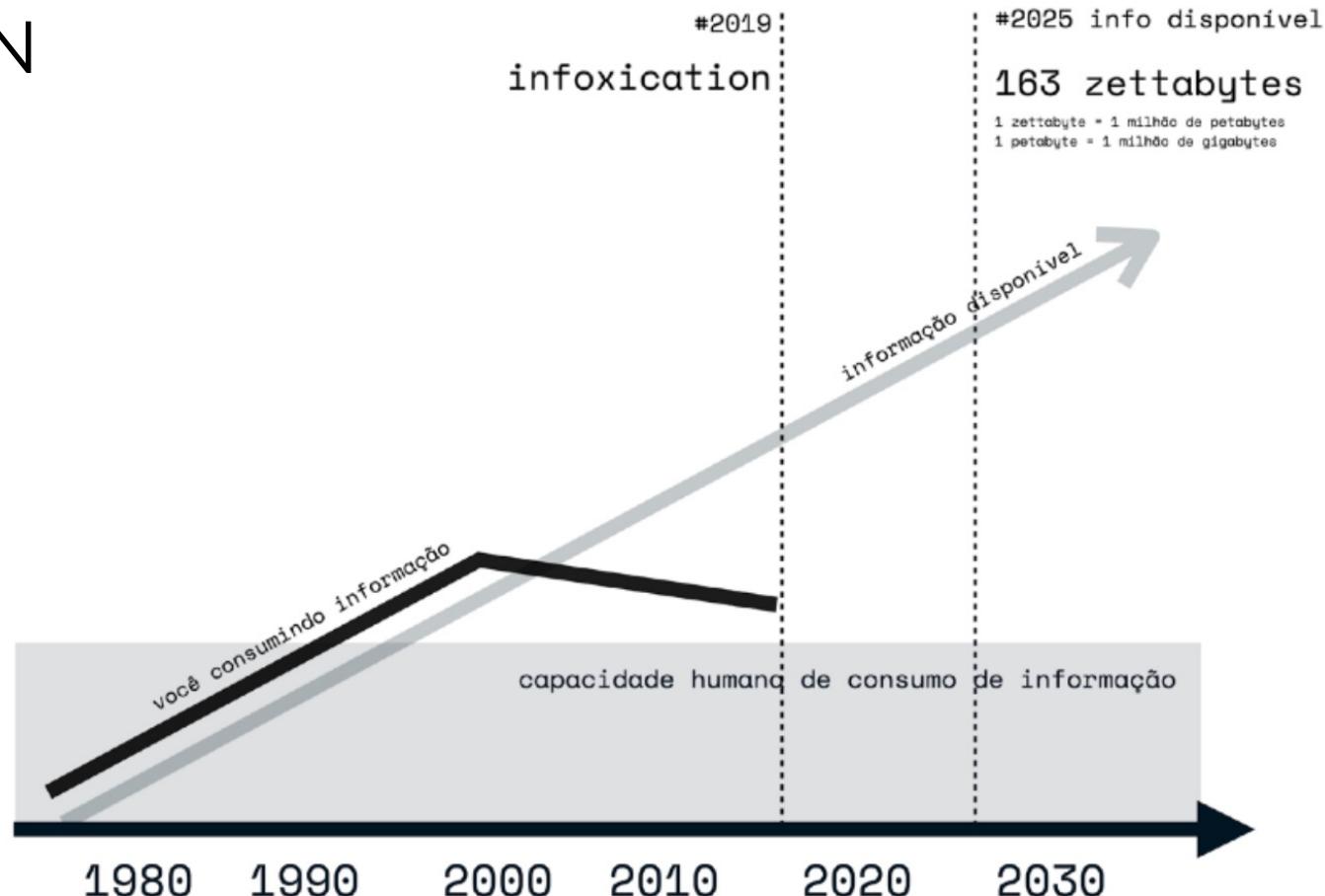
Source: Lori Lewis via AllAccess



E nossa capacidade cognitiva é limitada

<https://www.cappra.institute/2019>

INFOXICATION



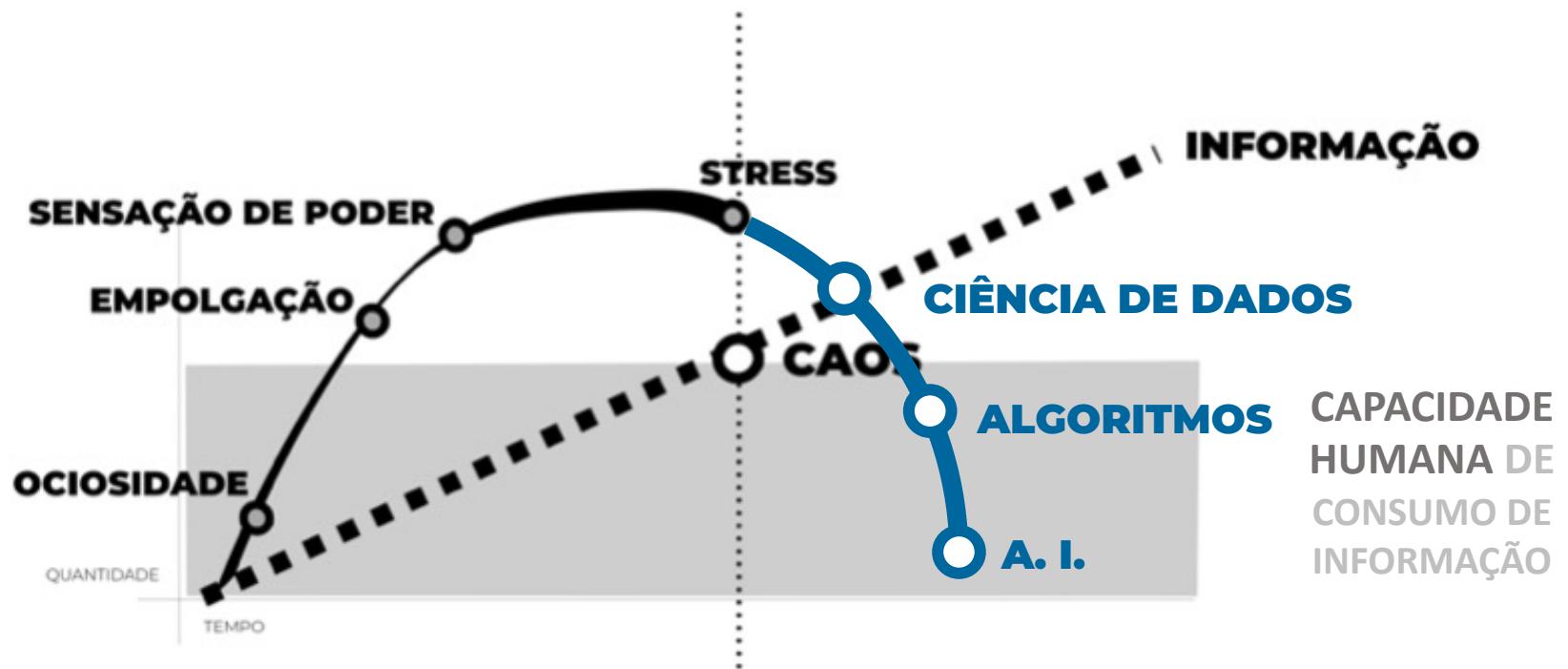
>> infográfico relação da quantidade de informação disponível com a capacidade humana de consumo de informação

“

Chegamos aos nossos limites físicos e biológicos como seres humanos. Não conseguimos mais consumir, processar e analisar a quantidade de informação que temos à nossa disposição.

Ricardo Capra

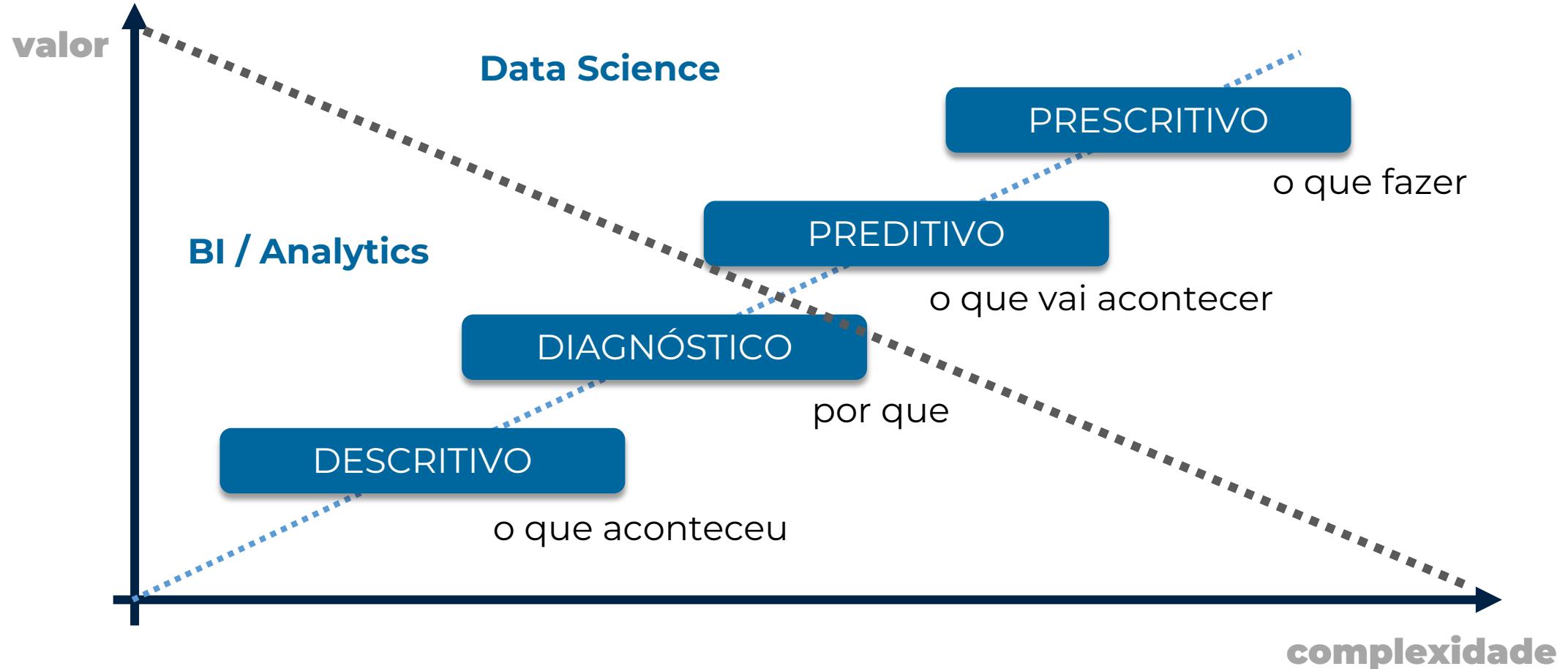
E nossa capacidade cognitiva é limitada



A geração do valor é no topo



A complexidade cresce também





DATA

Data Scientist: The Sexiest Job of the 21st Century

by Thomas H. Davenport and D.J. Patil

FROM THE OCTOBER 2012 ISSUE

[SUMMARY](#) [SAVE](#) [SHARE](#) [COMMENT](#) [TEXT SIZE](#) [PRINT](#) [\\$8.95 BUY COPIES](#)

When Jonathan Goldman arrived for work in June 2006 at LinkedIn, the business networking site, the place still felt like a start-up. The company had just under 8 million accounts, and the number was growing quickly as existing members invited their friends and colleagues to join. But users weren't seeking out

g1

GLOBO REPÓRTER

Globo Repórter revela quais são as profissões do futuro

Nesta sexta-feira (28), você vai conhecer a rotina de um superprofissional: o homem das mil habilidades.

Por Globo Repórter

25/06/2019 18h43 · Atualizado há 2 anos



fora o hype



O PROGRAMA ▾ NOTÍCIAS ▾ VÍDEOS ▾ FALE COM A PRODUÇÃO ▾ PODCAST REDES SOCIAIS

Carreira em ciência de dados promete salários de R\$ 22 mil

Pouco conhecida no Brasil, a profissão é promissora e muito procurada pelas empresas. Saiba mais sobre o trabalho desenvolvido por quem escolhe atuar nessa área

FALA BRASIL | Do R7

16/08/2018 - 01H00 (ATUALIZADO EM 16/08/2018 - 09H44)

COMPARTILHE: [f](#) [t](#) [w](#) [...](#)

ciência de dados | Brasil | Pesq.

Vagas | Último mês | Nível de experiência | Empre.

Ciência de dados em: Brasil 1.165 resultados

Gerente de Ciência de Dados
Dasa
Pinheiros, Espírito Santo, Brasil
2 ex-funcionários da empresa trabalham aqui
Promovida • 17 candidaturas

Especialista em Ciência de Dados - IT Data Analytics
BTG Pactual
São Paulo, São Paulo, Brasil
9 ex-funcionários da empresa trabalham aqui
Promovida

Analista de Dados III - Marketing Insights
Grupo Boticário
Brasil (Remoto)
1 conexão trabalha aqui
Há 3 horas

Data Visualization Analyst Senior
CIT
Brasil (Remoto)
2 conexões trabalham aqui
Há 3 semanas

Engenheiro de dados
Keep Simple
Brasil (Remoto)
Recrutando agora
Há 4 horas • Candidatura simplificada

Engenheiro(a) de Dados PL
Geofusion
Brasil (Remoto)
Recrutando agora
Há 5 horas

analista de dados | Brasil | Pesq.

Vagas | Último mês | Nível de experiência | Empre.

Analista de dados em: Brasil 3.673 resultados

Analista Engenharia de Dados PL/SR
Accenture Brasil
Rio de Janeiro, Rio de Janeiro, Brasil
1 conexão trabalha aqui
Promovida • 7 candidaturas

Analista de Dados Sênior
PicPay
São Paulo, São Paulo, Brasil
1 conexão trabalha aqui
Promovida

Analista de Dados (Data Engineer) - Asset Management
BTG Pactual
São Paulo, São Paulo, Brasil
10 ex-alunos trabalham aqui
Promovida

Analista de Dados (Foco em Modelagem) | BANCO DE TALENTOS
Stone
Rio de Janeiro, Rio de Janeiro, Brasil
23 ex-funcionários da empresa trabalham aqui
Promovida

Analista de dados
LUMA Ensino Individualizado
Vitória, Espírito Santo, Brasil (Remoto)
Seu perfil corresponde a esta vaga
Há 1 hora

Analista de Dados
Business Partners Consulting
São Paulo, Brasil (Presencial)
Recrutando agora

engenheiro de dados | Brasil | Pesq.

Vagas | Último mês | Nível de experiência | Empre.

Engenheiro de dados em: Brasil 1.306 resultados

Engenheiro(a) de Dados Sênior - Big Data
EY
Recife, Pernambuco, Brasil
1 conexão trabalha aqui
Promovida • 3 candidaturas

Engenheiro de Dados - Risco
Stone
Brasil (Remoto)
23 ex-funcionários da empresa trabalham aqui
Promovida • Candidatura simplificada

Engenheiro de Dados- Hadoop
Accenture Brasil
Rio de Janeiro, Rio de Janeiro, Brasil
1 conexão trabalha aqui
Promovida • 2 candidaturas

Engenheiro de Dados Sênior
Cartão Elo
São Paulo, Brasil (Híbrido)
2 ex-alunos trabalham aqui
Promovida • 5 candidaturas

Engenheiro de dados
GAVB
Canoas, Rio Grande do Sul, Brasil (Remoto)
7 conexões trabalham aqui
Há 1 dia • 18 candidaturas

Engenheiro de dados
Hrsoul
Brasil (Remoto)
Recrutando agora
Há 5 minutos • Candidatura simplificada

analytics | Brasil | Pesq.

Vagas | Último mês | Nível de experiência | Empre.

Analytics em: Brasil 5.519 resultados

Especialista em Engenharia de Dados - IT Data Analytics
BTG Pactual
São Paulo, São Paulo, Brasil
10 ex-alunos trabalham aqui
Promovida

Insight and Analytics Global Lead, People with Disabilities
Google
São Paulo, São Paulo, Brasil
Recrutando agora
Promovida • 4 candidaturas

Marketplace Analytics Operations Manager, Marketplace Central Ops LatAm
Uber
Osasco, São Paulo, Brasil
1 conexão trabalha aqui
Promovida • 18 candidaturas

Business Analyst - Marketing Analytics
Toptal
Porto Alegre, Rio Grande do Sul, Brasil (Remoto)
6 ex-alunos trabalham aqui
Promovida • 14 candidaturas

Gerente de Dados
Business Partners Consulting
São Paulo, Brasil (Presencial)
Recrutando agora
Há 41 minutos

Analista de Performance e BI
JB3 Investimentos
Florianópolis, Santa Catarina, Brasil (Presencial)

Top Trends in Data and Analytics For 2021



Accelerating Change

- 1 Smarter, Responsible, Scalable AI
- 2 Composable Data and Analytics
- 3 Data Fabric is The Foundation
- 4 From Big to Small and Wide Data



Operationalizing Business Value

- 5 XOps
- 6 Engineering Decision Intelligence
- 7 D&A as a Core Business Function



Distributed Everything

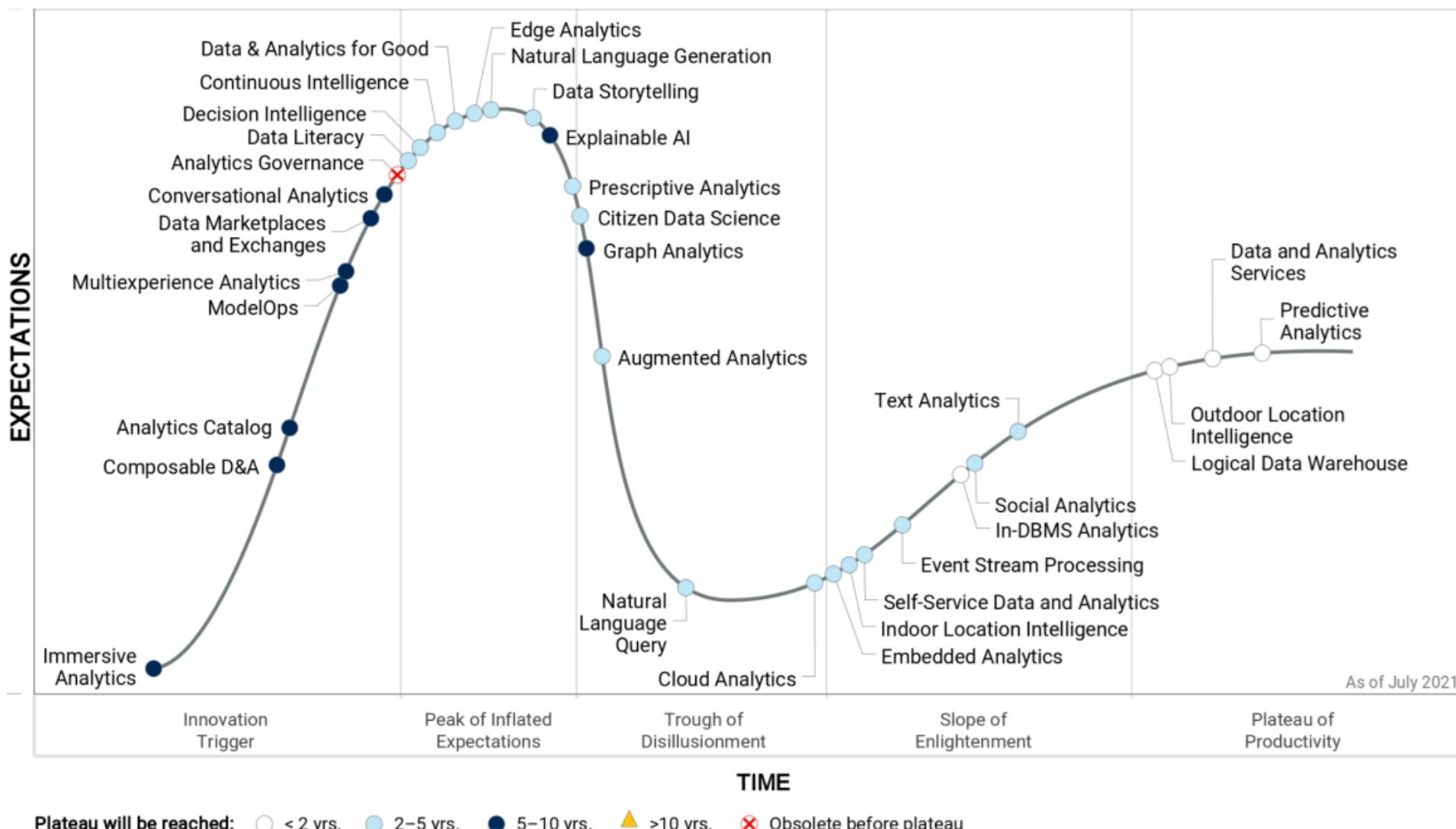
- 8 Graph Relates Everything
- 9 The Rise of the Augmented Consumer
- 10 D&A At the Edge

Source: Gartner

729348_C

Gartner

Hype Cycle for Analytics and Business Intelligence, 2021



Source: Gartner (July 2021)

747544



Papéis envolvendo
dados

Papéis envolvidos

- Quando falamos sobre trabalhar com dados, geralmente, falamos de atribuições de mais de um papel.
- Unicórnios, no contexto de dados, não existem :)



analista de negócios



analista de dados



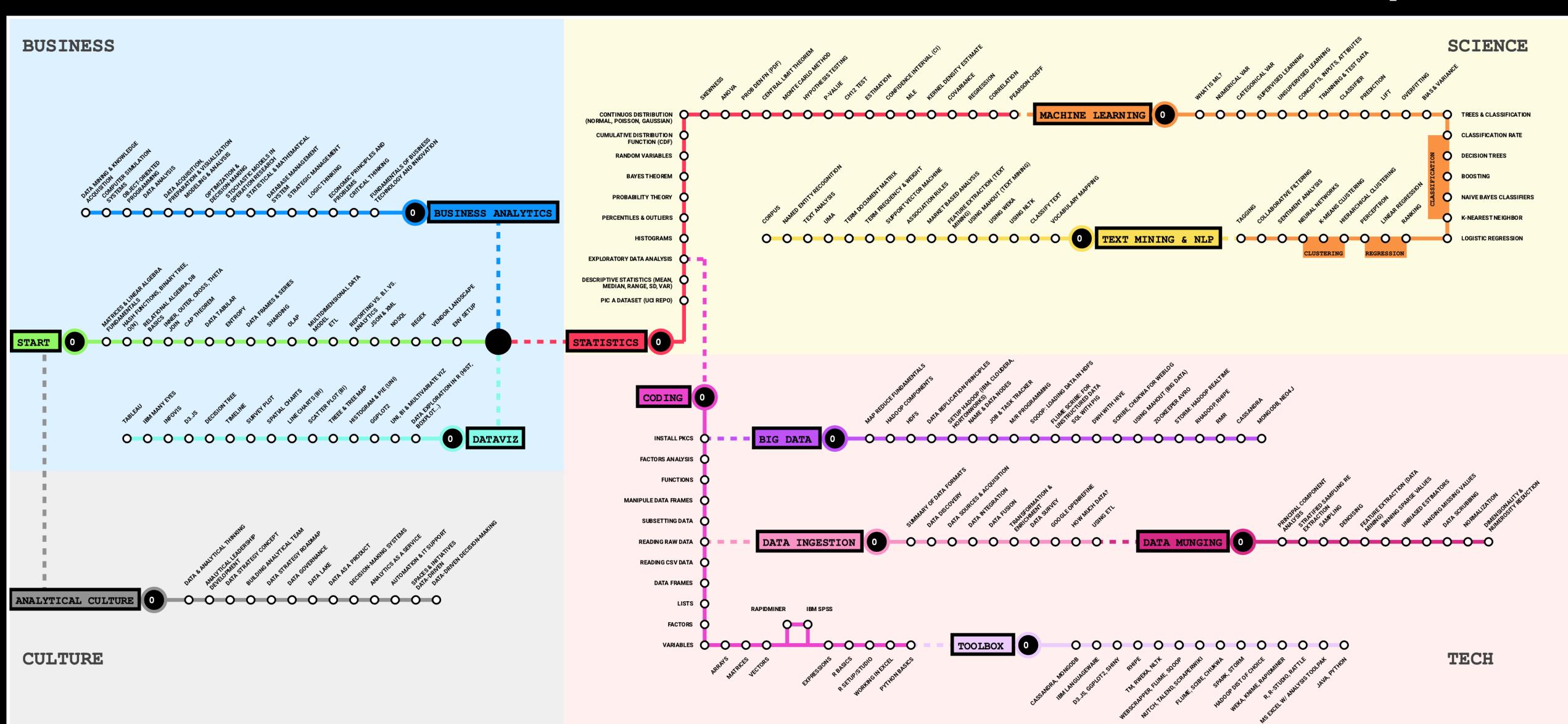
engenheiro de dados



cientista de dados



**administrador
de banco de dados**



Analista de negócios



- Embora existam algumas semelhanças entre um analista de dados e um analista de negócios, a principal diferença entre as duas funções é o que eles fazem com os dados.
- Um analista de negócios está mais perto da empresa e é especialista em interpretar os dados provenientes da visualização.
- Muitas vezes, as funções do analista de dados e o analista de negócios podem ser a responsabilidade de uma só pessoa.

Analista de dados



- Foco em maximizar o valor dos ativos de dados deles por meio de ferramentas de visualização e relatórios.
- Responsáveis pela criação de perfil, limpeza e transformação de dados. Inclui a criação de modelos de dados escalonáveis e eficientes e a habilitação e implementação de funcionalidades analíticas avançadas em relatórios para análise. Identifica os requisitos apropriados e necessários de dados e de relatórios, depois tem a tarefa de transformar dados brutos em insights relevantes e significativos.
- Os analistas de dados trabalham com engenheiros de dados para determinar e localizar fontes de dados apropriadas que atendam aos requisitos de stakeholder.

Engenheiro de dados



- Os engenheiros de dados provisionam e configuram tecnologias de plataforma de dados locais e na nuvem. Eles gerenciam e protegem o fluxo de dados estruturados e não estruturados de várias fontes. Os engenheiros de dados também garantem que os serviços de dados se integrem de maneira segura e direta uns aos outros.
- As principais responsabilidades de engenheiros de dados incluem o uso de serviços de dados locais e na nuvem e ferramentas para ingestão, saída e transformação de dados de várias fontes.
- Engenheiros de dados agregam um valor imenso a projetos de *business intelligence* e ciência de dados, ao reunir, estruturar e ingerir dados.

Cientistas de dados



- Os cientistas de dados realizam a análise avançada para extrair valor dos dados. Seu trabalho pode variar de análise descritiva a análise preditiva. A análise descritiva avalia os dados por meio de um processo conhecido como EDA (análise exploratória de dados).
- A análise preditiva é usada no machine learning para aplicar técnicas de modelagem que possam detectar anomalias ou padrões. Essas análises são partes importantes dos modelos de previsão.
- As análises descritiva e preditiva são apenas aspectos parciais do trabalho dos cientistas de dados. Alguns cientistas de dados podem trabalhar na área de aprendizado profundo, realizando experimentos iterativos para resolver um problema de dados complexo com o uso de algoritmos personalizados.
- A maior parte do trabalho em um projeto de ciência de dados é gasto em estruturação de dados e engenharia de recursos.
- À primeira vista, pode parecer que um cientista de dados e um analista de dados fazem trabalhos completamente diferentes, mas não é verdade.

Administrador de banco de dados



- Um administrador de banco de dados implementa e gerencia os aspectos operacionais de soluções de plataformas de dados híbridas e nativas de nuvem
- Um administrador de dados é responsável pela disponibilidade geral e pelo desempenho e otimizações consistentes das soluções de banco de dados.
- Um administrador de banco de dados monitora e gerencia a integridade geral de um banco de dados e o hardware no qual ele reside, enquanto um engenheiro de dados está envolvido no processo de estruturação de dados, em outras palavras, ingestão, transformação, validação e limpeza de dados para atender às necessidades e requisitos empresariais.
- O administrador de banco de dados também é responsável por gerenciar a segurança geral dos dados, concedendo e restringindo o acesso e os privilégios dos usuários aos dados, conforme determinado pelas necessidades e requisitos empresariais.

E como a **targettrust** pode ajudar?



Formação **Data Science**

A Formação Data Science é para você que deseja se especializar na Manipulação e Extração de Dados! Composta por 3 cursos essenciais (96h) , é ideal para quem busca desenvolver uma nova Skill e quer aprender na prática a criar aplicações que se destaquem por sua inteligência artificial. Nossa formação simula todos os desafios reais enfrentados por um cientista de dados. Por isso prepare-se para muito conhecimento e dedicação!



Formação **Data Analytics**

Quem deve fazer essa Jornada: pessoas que queiram entrar ou evoluir nas carreiras de Análise de Dados, Ciência de Dados, Engenharia de Dados, além de Analistas de Negócios, Analistas de Marketing, P.Os, P.Ms, que queiram apurar dados, gerar métricas, fazer análises e demonstrações mais assertivas do seu trabalho e resultados. Enfim, qualquer pessoa que queira conhecer a cultura Data Driven!

E como a **targettrust** pode ajudar?

Lóg. Programação e Ferramentas p/ trabalhar com Dados

Duração: 27h

Através da resolução de problemas, você aprenderá lógica de programação juntamente com a sintaxe da linguagem Python. Além de desenvolver o raciocínio lógico aplicado à resolução de problemas em nível computacional, capacitar-se a trabalhar com estruturas de dados identificando a aplicação destas na solução de problemas reais e entender as boas práticas de programação e organização de código.



**Formação
Data
Science**

Extração e Manipulação de Dados com Python

Duração: 30h

Aqui, você vai ser preparado para organizar, preparar e entender os dados manipulando-os com Python. Compreender os fundamentos da ciência de dados e da aprendizagem de máquina. Conhecer o processo de coleta e análise de dados aplicando métodos e ferramentas para responder a questões úteis à tomada de decisão. Apresentar os conceitos envolvidos na concepção de técnicas de visualização de informações, trabalhando com as principais bibliotecas de manipulação e visualização de informações.

Técnicas de Machine Learning aplicadas no Mercado

Duração: 39h

Ao final deste módulo você será um cientista de dados que além de organizar, preparar e entender, vai manipular dados com Python, compreender os fundamentos da ciência de dados e da aprendizagem de máquina, aplicando algoritmos de Machine Learning na prática e criando aplicações que se destaquem por sua inteligência artificial! Tomada de decisões com base na análise prévia de dados que, disponíveis em grande volume, conseguem detectar tendências de comportamento ao acompanhar padrões registrados.

E como a **targettrust** pode ajudar?

Análise e Visualização de Dados com SQL e Power BI

Duração: 18h

Torne-se um Analista de Dados capaz de organizar e preparar os dados para análise, descobrir padrões e insights, obter e expor conclusões significativas, comunicar claramente suas descobertas através de visualizações usando o SQL para acessar dados e o Power BI.



**Formação
Data Analytics**

Data Stories - contextualizando dados

Duração: 12h

Aprenda a construir histórias que combinam dados confiáveis e análises, dando significado aos dados, fornecendo visão, contexto e explicação, tornando a análise instantaneamente mais relevante, interessante e compreendida, gerando insights de dados de forma eficiente e eficaz.

Bora estudar?



Lógica de
Programação e
Ferramentas para
trabalhar com Dados

Quem são vcs?

- Nome / Empresa
- Experiência com o tema
- Expectativas com o treinamento



Houston, we have a problem



problema

substantivo masculino

1. assunto controverso, que pode ser objeto de pesquisas científicas ou discussões acadêmicas.
"o p. do descobrimento do Brasil"
2. questão social que traz transtornos e que exige grande esforço e determinação para ser
solucionado.
"o p. da seca no Nordeste brasileiro"
3. obstáculo, dificuldade que desafia a capacidade de solucionar de alguém.
"os moradores listaram os p. mais graves do bairro"
4. situação difícil; conflito emocional.
"ela vive cheia de problemas"
5. distúrbio, disfunção orgânica.
"p. digestivo"
6. pessoa, coisa ou situação incômoda, preocupante, fora de controle etc.
"essa menina é um p."
7. **MATEMÁTICA**
tarefa de calcular uma ou várias quantidades desconhecidas (*incógnitas*) relacionadas a outras
conhecidas (*dados*).

Origem

○ ETIM lat. *problēma, ātis* 'id.', do gr. *próblerma, atos* 'o que se tem diante de si; obstáculo; questão'

Questão DIFÍCIL, delicada,
suscetível de DIVERSAS
SOLUÇÕES

- Trocar a resistência de
um chuveiro
- Trocar o pneu do carro
- Definir onde almoçar
- Entregar um trabalho

Houston, we have a problem

E quando nos deparamos com um PROBLEMA?
Buscamos uma SOLUÇÃO!

Mas COMO?
Elaborando um PROCEDIMENTO

Ou seja, uma LISTA DE PASSOS

Houston, we have a problem

Procedimento para TROCAR RESISTÊNCIA DE UM CHUVEIRO:

1. Adquirir uma resistência nova
2. Abrir o chuveiro
3. Troca a resistência
4. Fechar o chuveiro
5. Testar o chuveiro

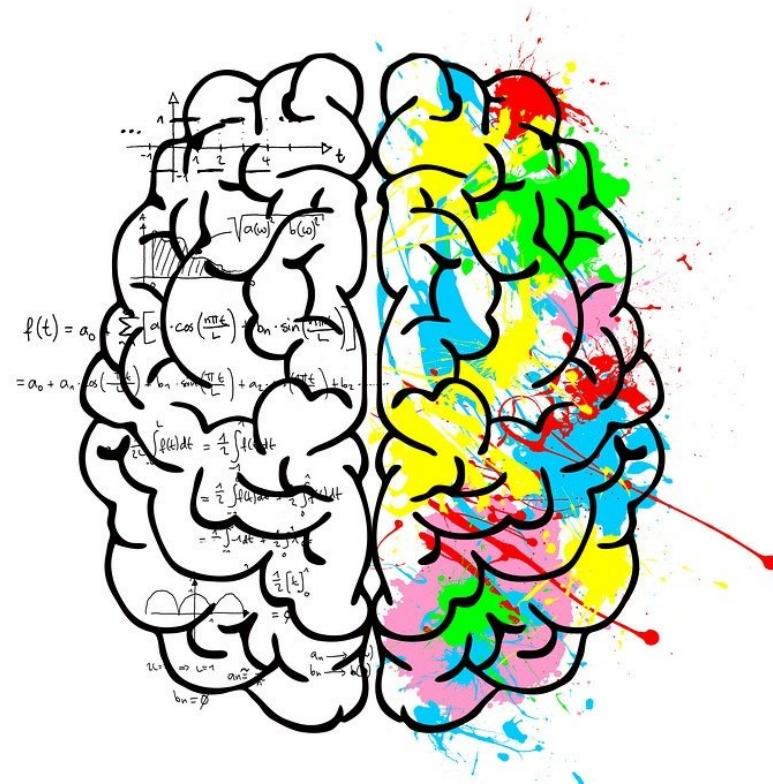
Tem certa **lógica**, né?

E o que é Lógica?

Ramo da FILOSOFIA e MATEMÁTICA
que tem como objetivo definir se
RACIOCÍNIOS são VÁLIDOS ou não.

Primordial para SOLUÇÃO DE
PROBLEMAS, pois permite alcançar
EFICIÊNCIA e EFICÁCIA

Imagine tentar trocar a resistência
de um chuveiro sem abri-lo ou sem
desliga-lo.



Algoritmos

SEQUÊNCIA DE PASSOS BEM DEFINIDA que deve ser seguida para realização de uma tarefa ou SOLUÇÃO DE UM PROBLEMA.

Para tanto, utilizamos a LÓGICA para LISTAR PASSOS que resultam na SOLUÇÃO de um determinado PROBLEMA

Exemplos:

- Resolver uma equação de segundo grau
- Somar vetores
- Preparar uma bebida



Exercício

Escreva um algoritmo em linguagem natural para calcular a média de três valores.



5 min

Algoritmo: MÉDIA TRÊS VALORES

1. Adicione o **VALOR1** ao **VALOR2**
2. Armazene o resultado da soma em **SOMA1**
3. Adicione o **VALOR3** ao **SOMA1**
4. Armazene o resultado da soma em **SOMA2**
5. Divida **SOMA2** por 3
6. Armazene o resultado da divisão em **MEDIA**

Fim_Algoritmo

Lógica de Programação

Estudo da estrutura da linguagem das máquinas

entrada

Forma de comunicação com computadores

Passos necessários para que os computadores precisam executar para resolver determinado problema ou atingir determinado objetivo

processamento

Essa comunicação é através de algoritmos (passos finitos e organizados)

saída

Lógica de Programação

- Variáveis e Constantes
 - Espaço reservado para guardar um tipo de dado. Um de cada vez.
Podendo ser numérico, caractere, alfanumérico ou lógico.
- Operadores
 - Trata a forma como comparamos, avaliamos, incrementamos ou decrementamos dados
- Estruturas de Controle (se)
- Estruturas de Repetição



Exercício

Escreva um algoritmo em linguagem natural para fazer um purê de batata com o que existe na geladeira seguindo a receita que vc tem.



5 min



Python

—

Linguagem Python

- Criada em 1991
 - Guido van Rossum no Instituto de Pesquisa Nacional para Matemática e Ciências da Computação (Países Baixos)
- Python é uma linguagem de programação de alto nível, versátil
- Projetada para ser legível, de fácil manutenção e com suporte avançado a mecanismos de reutilização de software.
- Filosofia por trás: esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade.



Linguagem Python

Principais motivos listados por quem desenvolve em Python:

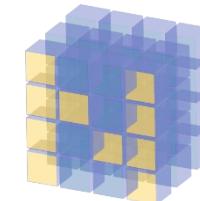
- Fácil aprendizado;
- Simples de programar;
- Sintaxe intuitiva;
- Open Source;
- Modularização;
- Multiplataforma;
- Grande quantidade de bibliotecas disponíveis;
- Grande comunidade de usuários;
- Documentação extensa;
- Número de oportunidades no mercado de trabalho.

Linguagem Python

Principais motivos listados por quem desenvolve em Python:

- Fácil aprendizado;
- Simples de programar;
- Sintaxe intuitiva;
- Open Source;
- Modularização;
- Multiplataforma;
- Grande quantidade de bibliotecas disponíveis;
- Grande comunidade de usuários;
- Documentação extensa;
- Número de oportunidades no mercado de trabalho.

Poder x Usabilidade x Comunidade



NumPy



Folium



Pandas



TensorFlow



Keras

Google Colab

- Serviço em nuvem, gratuito;
- Similar ao Jupyter Notebook (comum entre o pessoal que trabalha com Data Science);
- Suporta comandos bash;
- Vasta variedade de bibliotecas instaladas;
- Integração com Drive e Git;
- Vc só precisa de uma conta no Google :)

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

The screenshot shows the Google Colaboratory landing page. At the top, there's a navigation bar with the Colaboratory logo, the text "Olá, este é o Colaboratory", and links for Arquivo, Editar, Ver, Inserir, Ambiente de execução, Ferramentas, and Ajuda. To the right of the navigation bar are buttons for Compartilhar, settings, and a purple profile icon. Below the navigation bar is a search bar and a sidebar titled "Índice" which includes links for Primeiros passos, Ciência de dados, Machine learning, Mais recursos, Exemplos de machine learning, and Seção. The main content area features a "Mais recursos" section with a list of links: Experimentos com TPUs, Divulgação de pesquisas em IA, Criação de tutoriais, and examples of machine learning. It also includes sections for "Como trabalhar com Notebooks no Colab" and "Como trabalhar com dados".

Olá, este é o Colaboratory

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

Compartilhar

Índice

+ Código + Texto Copiar para o Drive

Conectar ^

Primeiros passos

Ciência de dados

Machine learning

Mais recursos

Exemplos de machine learning

Seção

Experimentos com TPUs

Divulgação de pesquisas em IA

Criação de tutoriais

Para ver notebooks do Colab que demonstram aplicações de machine learning, consulte os [exemplos de machine learning](#) abaixo.

Mais recursos

Como trabalhar com Notebooks no Colab

- [Visão geral do Colaboratory](#)
- [Guia sobre Markdown](#)
- [Importar bibliotecas e instalar dependências](#)
- [Salvar e carregar notebooks no GitHub](#)
- [Formulários interativos](#)
- [Widgets interativos](#)
- [TensorFlow 2 no Colab](#)

Como trabalhar com dados

Variáveis e Constantes

Uma variável é uma posição de memória capaz de armazenar um valor

Uma constante é uma variável com valores fixos

Tipo de Dado	Descrição
str	Caracteres
int	Número inteiro
float	Número real
complex	Números complexos
bool	Booleano (Verdadeiro ou Falso)

Existem outros tipos de dado com representatividade e precisão maior, sendo que, cada tipo possui um intervalo de valores aceitos.

Importante: Variáveis só existem em tempo de execução!

```
# CONSTANTES
PI = 3.14159265359
DIAS_DA_SEMANA = 7

# VARIÁVEIS
alunos = 13
nota = 9.5
nome = 'Matheus'

print(type(PI))
print(type(DIAS_DA_SEMANA))
print(type(alunos))
print(type(nota))
print(type(nome))
```

*Conheça os padrões de nomes de variáveis utilizados em diferentes linguagens

<https://medium.com/better-programming/string-case-styles-camel-pascal-snake-and-kebab-case-981407998841>

Operadores

A linguagem Python oferece uma gama variada de operadores

- Operadores aritméticos
- Operadores de atribuição
- Operadores relacionais
- Operadores lógicos

Operadores Aritméticos

Operadores matemáticos usuais:

Operador	Nome	Ação que executa
+	Adição	Soma dois valores
-	Subtração	Subtrai dois valores
*	Multiplicação	Multiplica dois valores
**	Potenciação	Potência entre dois valores
/	Divisão	Divide dois valores(resultado é um real)
//	Divisão Inteira	Divide dois valores (resultado é um inteiro)
%	Módulo	Retorna o resto da divisão de um valor por outro

```
adicao = 1 + 4  
subtracao = 5 - 3  
multiplicacao = 2 * 5  
potenciacao = 2 ** 3  
divisao_real = 5 / 2  
divisao_inteira = 5 // 2  
modulo = 10 % 3
```

```
print(adicao)  
print(subtracao)  
print(multiplicacao)  
print(potenciacao)  
print(divisao_real)  
print(divisao_inteira)  
print(modulo)
```



targettrust⁺
treinamento e tecnologia

Operadores de Atribuição

Uma atribuição é uma expressão cujo valor resultante é atribuído à uma variável

Operador	Exemplo	Ação que executa
=	idade = 10	Atribui um valor à uma variável
+=	i += 2	i = i + 2
-=	x -= i + 1	x = x - (i + 1)
*=	salario *= 1.1	salario = salario * 1.1
/=	soma /= 2	soma = soma / 2
//=	soma // 2	soma = soma // 2

Operadores Relacionais

Relações que uma variável tem com outra:

Operador	Nome	Ação que executa
>	maior	$A > B$ (verdadeiro se A for maior que B)
<	menor	$A < B$ (verdadeiro se A for menor que B)
\geq	maior ou igual	$A \geq B$ (verdadeiro se A for maior ou igual a B)
\leq	menor ou igual	$A \leq B$ (verdadeiro se A for menor ou igual a B)
$=$	igual	$A == B$ (verdadeiro se A for igual a B)
\neq	diferente	$A != B$ (verdadeiro se A for diferente de B)

Operadores Lógicos

Operadores utilizados para tomada de decisão:

Operador	Nome	Exemplo
and	E	$A > B \text{ and } A > C$
or	OU	$\text{hora} < 0 \text{ or } \text{hora} > 23$
not	Não	not valido

A	B	not A	A and B	A or B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Funções Built-in (nativas)

O Python implementa por padrão diversas funções, inclusive de entrada e saída:

Leitura do teclado

- `input()`

Escrita na tela

- `print()`

Retorna o tipo do dado

- `type()`

Leitura e Escrita do Teclado

```
nome = input('Digite o seu nome: ')
sobrenome = input('Digite o seu
sobrenome: ')

print(nome + ' ' + sobrenome)
```



Prática

Escreva um algoritmo em Python que calcula a média de 2 números quaisquer



5 min



Prática

Escreva um algoritmo em Python que calcula o salário líquido de um funcionário. Para tanto, pergunte ao mesmo:

- a) Quanto ele ganha por hora
- b) Quantas horas ele trabalha por dia
- c) Quantos dias ele trabalhou no mês



10 min

Condições

Quando o salário líquido de uma pessoa é calculado, existem faixas de imposto que serão descontadas. Se nosso salário bruto passa de um certo valor, então uma determinada taxa de imposto será aplicada a ele.

Em Python, o comando if irá anteceder uma proposição lógica, e caso a mesma for verdadeira, as linhas identadas abaixo serão executadas. Python não possui switch/case.

Por exemplo,

Se chover à tarde, fecharei as janelas.

Se chover à tarde ou ventar forte, fecharei as janelas.

Se chover fraco e ventar forte, fecharei as janelas.

if proposição 1:

código que executa caso
proposição 1 seja verdadeira

elif proposição 2:

código que executa caso
proposição 1 não seja verdadeira
e proposição 2 seja verdadeira

else:

código que executa caso
todas proposições anteriores
sejam falsas

O que esse código faz?

```
print('Digite dois números:')
```

```
N1 = input()
```

```
N2 = input()
```

```
if N1 == 2 * N2:
```

```
    print('N1 é o dobro de N2')
```

```
elif N2 == 2 * N1:
```

```
    print('N2 é o dobro de N1')
```

```
else:
```

```
    print('Não é o dobro')
```

Tem algum problema?

```
print('Digite dois números:')
```

```
N1 = input()
```

```
N2 = input()
```

```
if N1 == 2 * N2:
```

```
    print('N1 é o dobro de N2')
```

```
elif N2 == 2 * N1:
```

```
    print('N2 é o dobro de N1')
```

```
else:
```

```
    print('Não é o dobro')
```

```
print('Digite dois números:')

N1 = input()
N2 = input()

if N1 == 2 * N2:
    print('N1 é o dobro de N2')
elif N2 == 2 * N1:
    print('N2 é o dobro de N1')
else:
    print('Não é o dobro')
```

Qual o problema?

O comando `input` retorna uma *string*.

Precisamos converter para o tipo que queremos trabalhar.



Prática

Escreva um algoritmo em Python que testa se um número é maior que 10



10 min



E o imposto?



Escreva um algoritmo em Python que calcula o salário líquido de um funcionário. Para tanto, pergunte ao mesmo:

- a) Quanto ele ganha por hora
- b) Quantas horas ele trabalha por dia
- c) Quantos dias ele trabalhou no mês



15 min



Prática

Escreva um algoritmo em Python que lê 3 valores e imprime o maior deles



10 min



Prática

Escreva um algoritmo em Python que lê três valores quaisquer e testa se existe um par de valores em que um é o dobro do outro. Caso seja verdade imprime os dois valores, caso contrario imprime “Não encontrei”



10 min

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br



targettrust⁺
treinamento e tecnologia

Repetição

Às vezes é necessário repetir um conjunto de operações. Como por exemplo, calcular o salário líquido de 100 funcionários a partir de seu salário bruto. Ou calcular os números primos entre um intervalo como por exemplo 2 a 50.

Em Python, os comandos `for` e `while` repetem operações. O comando `do-while` não existe no Python.

Range e Reversed

O comando range é o gerador que nos permite delimitar a quantidade de vezes que queremos repetir o nosso código. Ele possui três parâmetros:

- start (por padrão ele é 0)
- end (não possui padrão, deve ser informado sempre)
- step (por padrão ele é 1)

Exemplo:

range(5) -> 0 1 2 3 4

range(3, 8) -> 3 4 5 6 7

range(0, 11, 2) -> 0 2 4 6 8 10

range(9, 5, -1) -> 9 8 7 6

reversed(range(5)) -> 4 3 2 1 0

reversed(range(3, 8)) -> 7 6 5 4 3

reversed(range(0, 11, 2)) -> 10 8 6 4 2 0

reversed(range(9, 5, -1)) -> 6 7 8 9

Comando de repetição FOR

```
print('Número pares entre 1 e 50: ')
for numero in range(1, 50):
    if numero % 2 == 0:
        print(numero, end=' ')
```

```
print('Digite um intervalo (inicio, fim)')
inicio = int(input('Inicio: '))
fim = int(input('Fim: '))

for numero in range(inicio, fim + 1):
    print(str(numero), end=' ')
```

Comando de Repetição - While

O laço de repetição while irá repetir o seu código enquanto a proposição lógica for verdadeira!

Exemplos:

- Enquanto a hora digitada for inválida pergunte para o usuário a hora.
- Enquanto o valor digitado for positivo calcule a raiz quadrada

Comando de repetição WHILE

```
print('Número impares entre 1 e 50: ')
numero = 1
```

```
while numero <= 50:
    if numero % 2 == 1:
        print(numero, end=' ')
    numero = numero + 1
```



Exercício

Escreva um algoritmo em Python que conta de 10 a 20 de 2 em 2, e imprime os valores contados



5 min



Exercício

Escreva um algoritmo em Python que exibe na tela todos números primos entre 2 e 50



10 min



Exercício

Escreva um algoritmo em Python que solicite um intervalo de anos e diga todos bissextos nesse intervalo.



10 min

Strings

Uma string em Python é a representação de uma ou mais palavras

- nome_completo = 'Matheus da Silva Serpa'
- mensagem = "Eu gostaria de um copo d'água"

Em Python, é possível percorrer uma string utilizando um laço for

```
for letra in nome:  
    print(letra)
```

Em Python não existe diferença entre um caractere (char) e um conjunto de caracteres (string)

Strings - Operações

Algumas operações aritméticas podem ser utilizadas com strings

- nome = 'Matheus'
- sobrenome = 'Serpa'

Concatenação de duas ou mais strings

- nome_completo = nome + sobrenome
- nome_completo = nome + ' ' + sobrenome

Multiplicação de Strings

- traco = '-'
- separador = traco * 80

Tamanho de Strings

Dadas algumas strings

- nome = 'Matheus'
- sobrenome = 'Serpa'

Tamanho da string

- len(nome)
- len(nome) / len(sobrenome)
- len('Segunda-feira')

Métodos com Strings

Mais utilizadas

- count, find, format, join, lower, replace, split, strip
- Outras operações
- <https://docs.python.org/pt-br/3.8/library/stdtypes.html#string-methods>



Prática

Implemente um programa em Python que recebe como entrada uma *string* digitada pelo usuário, e que gera uma nova *string* de saída substituindo todas as vogais (maiúsculas ou minúsculas) pelo caractere _. Como saída o programa deve imprimir a *string* gerada.

Veja o exemplo:

```
str = "Eu quero aprender Python"  
Saída = "__ q__r__pr_nd_r Pyth_n"
```



10 min

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br



targettrust⁺
treinamento e tecnologia



Prática Extra

Implemente um programa em Python que recebe como entrada duas *strings* digitadas pelo usuário (o programa só aceita *Strings* de mesmo tamanho), e como resultado imprime as duas strings alternadamente. Ou seja, deve-se imprimir o 1º char de str1, depois o 1º char de str2, e assim por diante.

Veja o exemplo:

```
str1 = "joão"    str2 = "1234"  
Saída = "j1o2a3o4"
```



10 min

Formatação de Strings

O método `format` nos auxilia na hora de formatar uma ou mais strings.

O uso mais comum é no comando `print`, quando queremos formatar a saída para um arquivo ou para a tela do usuário.

Exemplos:

- `print('MEDIA = {:.5f}'.format(MEDIA))`
- `print("Essa é a aula de número {} do curso de Lógica com Python".format(4))`
- `usuario_cidade = "{} nasceu em {} e mora em {}"`
- `print(usuario_cidade.format("Matheus", "Alegrete", "Porto Alegre"))`



Prática

Leia uma **string** grande (por exemplo o Hino Nacional Brasileiro). Após, responda as seguintes perguntas utilizando os métodos da **string**. Utilize o **format** em todos seus **print**.

- a) Qual o tamanho da **string**?
- b) Em qual posição da **string** aparece a primeira letra **a**?
- b) Em qual posição da **string** aparece a última letra **e**?
- d) Quantos espaços temos na **string**?



15 min

Comandos in/not in

Podemos verificar se um elemento está na lista ou não está utilizando esse operador

'Alegre' in 'Porto Alegre'

5 in [1, 2, 3, 4, 6]

5 not in [1, 2, 3, 4, 6]

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br



targettrust⁺
treinamento e tecnologia

Listas

Declaração de uma variável / constante

num1 = 5

num2 = 8

num3 = 6

num4 = 1

E se eu preciso armazenar 1000 números?

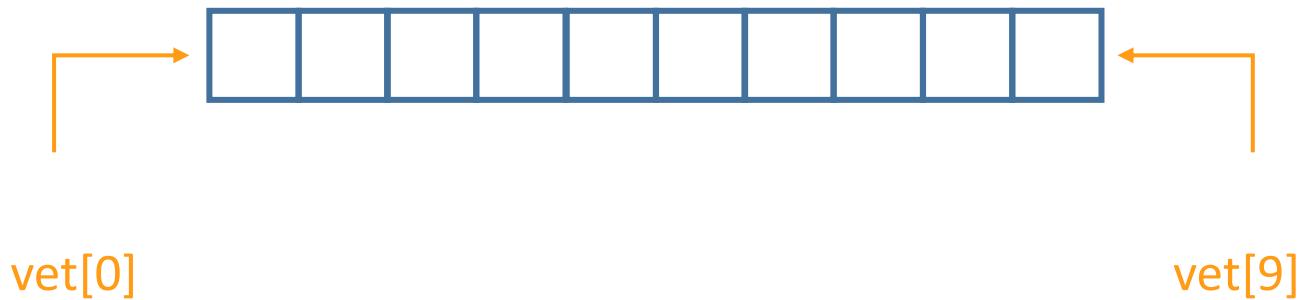
- Declaro 1000 variáveis?
- Não, usamos listas!

Declaração de listas

- num = []

Listas

Representação em 1 dimensão



Listas em Python são utilizadas para organizar e agrupar dados.
Os dados podem ser homogêneos ou heterogêneos!

- `notas_prova_1 = [7.1, 8.5, 7.5, 4.0, 3.0]`
- `minha_lista = [5, 3.5, 'Matheus', True]`

Listas - Slice

Também chamado de fatiamento

pessoas[2:5]

pessoas[3:7]

pessoas[:7]

pessoas[7:]

Listas - Exemplos

```
meses = [Jan','Fev','Mar','Abr','Mai','  
Jun','Jul','Ago','Set','Out','Nov','Dez']
```

```
for mes in meses:  
    print(mes)
```

Listas - Exemplos

```
meses = [Jan','Fev','Mar','Abr','Mai','  
Jun','Jul','Ago','Set','Out','Nov','Dez']
```

```
print(meses[0])  
print(meses[len(meses) - 1])
```

```
print(meses[-2])  
print(meses[-1])
```



Prática

Implemente um programa em Python que recebe como entrada uma *string* digitada pelo usuário e retorna a *string* invertida.

Veja o exemplo:

Entrada = "existem várias maneiras de inverter uma string"

Saída = "gnirts amu retrevni ed sarienam sairáv metsixe"



Listas - Métodos

Mais utilizadas

- append, insert, remove, pop, index, count, sort, sorted, join, copy

Outras operações

- <https://docs.python.org/pt-br/3.9/tutorial/datastructures.html>

Dicionários

Tipo de dado que armazena mapeamentos chave / valor

nome_idade["Matheus"] = 25

nome_idade["João"] = 65

nome_idade["Maria"] = 12

Dicionários

Tipo de dado que armazena mapeamentos chave / valor

nome_idade["Matheus"] = 25

nome_idade["João"] = 65

nome_idade["Maria"] = 12

O que mais é possível fazer?

- alunos["Matheus"]["nota"] = 9.5
- alunos["Matheus"]["faltas"] = 4
- pedidos["Matheus"]["25/06/2020"].append("Mouse Cobra M711")
- pedidos["Matheus"]["25/06/2020"].append("Mousepad Taurus (930x300mm)")

Dicionários - Exemplos

```
siglas = {"TT": "TargetTrust", "FB":  
"Facebook", "RS": "Rio Grande do  
Sul"}
```

```
print(siglas["TT"])  
print(siglas["FB"])  
print(siglas["RN"])
```

Retorna o elemento, se não existir, um erro ocorre.

```
notas = {"Bruna": [8.3, 6.6], "Gabriel": [0.9, 6.3], "Lucas": [8  
.0, 8.2]}
```

```
print(notas.get("Bruna", "Aluno não encontrado."))  
print(notas.get("Lucas", "Aluno não encontrado."))  
print(notas.get("Matheus", "Aluno não encontrado."))
```

```
notas["Matheus"] = [9.5, 8.3]
```

```
print(notas.get("Matheus", "Aluno não encontrado."))
```

```
notas = {"Bruna": 7.4, "Gabriel": 3.4, "Lucas": 8.1}
```

```
print('chaves: {}\\n'.format(notas.keys()))
```

```
print('for in keys')
```

```
for chave in notas.keys():
    print('\\t' + chave)
```

```
print()
```

Percorre, ou seja, itera sobre as chaves do dicionário.

```
notas = {"Bruna": 7.4, "Gabriel": 3.4, "Lucas": 8.1}

print('valores: {}\\n'.format(notas.values()))

print('for in values')
for chave in notas.values():
    print('\\t' + str(chave))
print()
```

Percorre, ou seja, itera sobre os valores de todas chaves do dicionário.

```
notas = {"Bruna": 7.4, "Gabriel": 3.4, "Lucas": 8.1}
```

```
print('items: {}\\n'.format(notas.items()))
```

```
print('for in items')
for chave, valor in notas.items():
    print('\\t{} : {}'.format(chave, valor))
print()
```

Percorre, ou seja, itera
sobre todos pares chave,
valor

Sets

Para resolver determinados problemas precisamos rapidamente remover os elementos duplicados.

Exemplo:

```
produtos = ['Monitores', 'Monitores', 'Teclado', 'Mouse', 'Mousepad', 'Pen Drive', 'Pen Drive']
print(produtos)
```

```
produtos_sem_repeticao = set(produtos)
print(produtos_sem_repeticao)
```

Diversas funções e métodos para listas funcionam com sets.

Por exemplo: in, not in, add, pop, len

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br



targettrust⁺
treinamento e tecnologia

```
notas = {"Gabriel": 3.4, "Bruna": 9.4, "Lucas": 8.1}

print('original: {}'.format(notas))

notas_por_nome = {}
for aluno, nota in sorted(notas.items(), key = lambda x: x[0]):
    notas_por_nome[aluno] = nota
print('por nome: {}'.format(notas_por_nome))

notas_por_nota = {}
for aluno, nota in sorted(notas.items(), key = lambda x: x[1]):
    notas_por_nota[aluno] = nota
print('por nota: {}'.format(notas_por_nota))
```

Ordenação de
Dicionários

Tuplas

Utilizado para armazenar informações relacionadas.

Exemplo:

```
coordenadas = (-30.051127,-51.2308707)
print("Latitude: {}".format(coordenadas[0]))
print("Longitude: {}".format(coordenadas[1]))
```

As vezes queremos fazer o *unpacking*, pois é desejável trabalhar com variáveis

```
latitude, longitude = coordenadas
print("Latitude: {}".format(latitude))
print("Longitude: {}".format(longitude))
```

Bibliotecas

Python possui uma biblioteca padrão, que oferece uma série de recursos.

Fornecem soluções padronizadas para muitos problemas que ocorrem em programação cotidiana.

Como usar?

- `import <biblioteca>`
- `from <biblioteca> import <função>`

Onde encontrar:

- <https://pypi.org/>

Funções

E quando o programa fica grande?

- Modularidade (Dividir o programa em partes)

Operações específicas

- Média, maior, transformada de Laplace, etc

Vantagens

- Facilita entendimento
- Facilita manutenção
- Permite reuso (bibliotecas)

Funções

Exemplos de funções conhecidas implementadas em bibliotecas

- input, print, range, math.sqrt, str, int, etc.

Declaração de funções em Python

- def nome_função(parametro1, parametro2, ...)

Chamada de funções em Python

- minha_var = minha_função(var1, var2, ...)

Funções

Por enquanto, só usamos funções que vêm com o Python, mas também é possível acrescentar novas funções. Uma definição de função especifica o nome de uma nova função e a sequência de instruções que são executadas quando a função é chamada.

```
def nome_funcao(parâmetros):
    código
    código
    código

    return algo
```

Passagem de Parâmetros

Vantagens

- Facilita o entendimento
- Diminui os erros na programação

Exemplo

- Função para calcular o desconto do INSS, envia o salário bruto e retorna o líquido.
- Função para calcular a média dos valores de uma lista, envia a lista e retorna a média.
- Dessa maneira é possível para diferentes listas de dados calcular a média.

```
def media(x, y, z):
    return (x + y + z) / 3

print('Digite três números: ')
v1 = float(input())
v2 = float(input())
v3 = float(input())

resultado = media(v1, v2, v3)

print('A média é ' + str(round(resultado, 1)))
```

Lambda λ

É uma maneira de criar funções anônimas

Funções que não tem nome!

Muito útil para realizar uma operação que não vai ser repetida muitas vezes a ponto de se criar uma função e também quando outras funções ou métodos solicitam funções como parâmetro.

Lambda λ

```
def soma(x, y):  
    return x + y  
  
subtracao = lambda x, y: x - y  
  
print(soma(5, 3))  
print(subtracao(5, 3))
```



Prática

Leia uma **string** grande (por exemplo o Hino Nacional Brasileiro). Após, responda as seguintes perguntas.

- a) Quais são as **cinco palavras** que mais aparecem no texto?
- b) Quais são as **cinco palavras** que menos aparecem no texto?



15 min

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br



targettrust⁺
treinamento e tecnologia

Funções e Lambda λ

Recapitulando ...

```
def valida_login(usuario, senha):
    if(usuario == 'matheus' and senha == '12345'):
        return True
    else:
        return False

user = input('Digite o usuário: ')
password = input('Digite a senha: ')

if valida_login(user, password):
    print('Acesso permitido!')
else:
    print('Acesso negado!')
```

```
def mostra_menu():
    print('-----')
    print('1 - Cadastro de clientes')
    print('2 - Cadastro de vendas')
    print('3 - Sair')
    print('-----')

opcao = 0

while opcao != 3:
    mostra_menu()
    opcao = int(input('Digite uma opção: '))

    print('Saindo...')
```

Lambda λ

```
def soma(x, y):
    return x + y

subtracao = lambda x, y: x - y

print(soma(5, 3))
print(subtracao(5, 3))
```



Prática

Dado um conjunto de dados, como eliminar os outliers? Ou seja, os pontos fora da curva.

- a) Calcular a média
- b) Calcular o desvio padrão
- c) Criar uma nova lista apenas com os dados dentro do intervalo [media – desvio, media + desvio]



15 min



Prática

Transforme os exercícios a seguir em funções:

1. Escreva um algoritmo em Python que lê 3 valores e retorna o maior deles.
2. Escreva um algoritmo em que recebe o salário, hora trabalhadas e qtd dias e retorne o salário líquido.



15 min

Filter / Map / Reduce / Zip

- A função *filter(função, lista)* oferece uma maneira conveniente de filtrar todos os elementos de um iterável para o qual a função retorne True.
- A função *filter(função(), l)* precisa de uma função como seu primeiro argumento. A função precisa retornar um valor booleano (True ou False).
- Esta função será aplicada a cada elemento do iterable. Somente se a função retornar True, o elemento do iterable será incluído no resultado.

Filter / Map / Reduce / Zip

- *map ()* é uma função que leva em dois argumentos: uma função e uma seqüência iterable. Na forma:

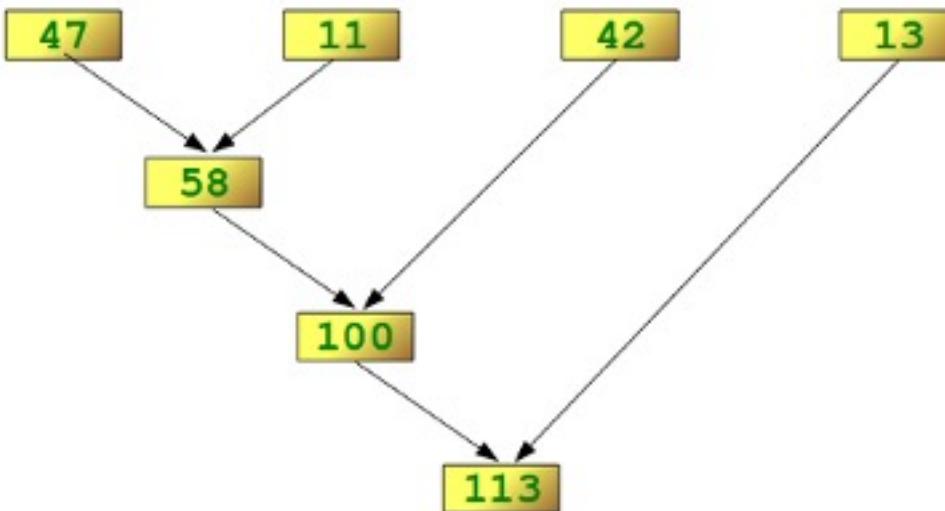
map(função, sequência)

- O primeiro argumento é o nome de uma função e a segunda uma seqüência (por exemplo, uma lista).
- *map()* aplica a função a todos os elementos da seqüência. Ele retorna uma nova lista com os elementos alterados por função.
- *map()* pode ser aplicado a mais de um iterable. Os iteráveis devem ter o mesmo comprimento.
- Por exemplo, se estamos trabalhando com duas listas-*map()* aplicará sua função lambda aos elementos das listas de argumentos, ou seja, aplica-se primeiro aos elementos com o índice 0, e depois aos elementos com o 1º índice até o que o índice N seja alcançado.

Filter / Map / Reduce / Zip

- A função `reduce(funcção, sequência)` aplica continuamente a função à sequência. Em seguida, ele retorna um único valor.
- Se $\text{seq} = [s1, s2, s3, \dots, sn]$, a redução de chamada `(funcção, seqüência)` funciona assim:
 - No início, os dois primeiros elementos de seq serão aplicados à função, isto é, $\text{func}(s1, s2)$
 - A lista em que a `reduce()` funciona parece assim: $[\text{funcão}(s1, s2), s3, \dots, sn]$
 - No próximo passo, a função será aplicada no resultado anterior e no terceiro elemento da lista, ou seja, $\text{funcão}(\text{funcão}(s1, s2), s3)$
 - A lista parece agora: $[\text{funcão}(\text{funcão}(s1, s2), s3), \dots, sn]$
 - Continua assim até apenas um elemento é deixado e retorna esse elemento como resultado de reduzir ()

Filter / Map / Reduce / Zip



Filter / Map / Reduce / Zip

- `zip()` cria um iterador que agrega elementos de cada um dos iteráveis.
- Retorna um iterador de tuplas, onde a i -ésima tupla contém o i -ésimo elemento de cada uma das seqüências ou do iterável passado como argumento.
- O iterador para quando a entrada mais curta disponível se esgota. Com um único argumento iterável, ele retorna um iterador de n -tuplas. Sem argumentos, ele retorna um iterador vazio.
- O `zip()` só deve ser usado com entradas de comprimento desiguais quando você não se preocupa com os valores ininterruptos, além dos valores mais longos.

Unindo com Lambda λ

```
notas_NEI = [  
    [1.0, 7.1, 8.5, 7.5, 0.0, 4.0, 9.5, 4.5, 8.0, 7.0, 9.0, 3.0, 9.5, 10.0, 6.0, 9.0, 6.0, 7.  
    0],  
    [6.0, 9.5, 7.0, 9.5, 7.5, 8.5, 7.0, 0.0, 5.0, 9.0, 10.0, 7.0, 8.5, 0.0, 8.5, 7.5, 5.5, 1  
    0.0]  
]
```

```
medias = list(map(lambda x: round(sum(x) / len(x), 1), notas_NEI))  
print(medias)
```

Unindo com Lambda λ

```
alunos = ["ALYSSON", "ANDRINO", "BRUNA", "EDGAR", "GABRI  
EL", "HENRIQUE", "JEAN", "JOÃO", "KAUAN", "LUAN", "LUCAS", "  
LUIZ", "MATEUS", "RICARDO", "RODRIGO", "VINICIUS", "WAGNE  
R", "WILLIAM"]
```

```
alunos_com_nome_pequeno = list(filter(lambda x: len(x) <= 5, a  
lunos))  
print(alunos_com_nome_pequeno)
```

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br



targettrust⁺
treinamento e tecnologia

Enumerate

- Enumerate permite você contar elementos enquanto itera através de um objeto. O método retorna uma tupla na forma (contagem, elemento).
- enumerate() torna-se particularmente útil quando você precisa ter algum tipo de rastreador. Por exemplo:

```
for count,item in enumerate(lst):
    if count >= 2:
        break
    else:
        print item
```

List / Dict Comprehension

- Além das operações de sequência e métodos de lista, o Python inclui uma operação mais avançada chamada de compreensão de lista ou dicionário.
- As compreensões de lista ou dicionário nos permitem construir esses objetos usando uma notação diferente. Você pode pensar nisso essencialmente como um loop construído dentro de colchetes.

```
# Pega todas as letras em uma string
```

```
[st = [x for x in 'word']]
```



Prática

Crie uma lista de 3 elementos e calcule a terceira potência de cada elemento usando List Comprehension.



05 min



Prática

Crie duas funções, uma para elevar um número ao quadrado e outra para elevar ao cubo. Aplique as duas funções aos elementos da lista abaixo.

Obs: as duas funções devem ser aplicadas simultaneamente.



05 min



Prática

Usando a função `filter()`, encontre os valores que são comuns às duas listas abaixo.



10 min



Prática

Considere a lista abaixo e retorne apenas os elementos cujo índice for maior que 5.

```
lista = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```



10 min



Prática

Considere os dois dicionários abaixo.

Crie um terceiro dicionário com as chaves do dicionário 1 e os valores do dicionário 2.

```
dict1 = {'a':1,'b':2}  
dict2 = {'c':4,'d':5}
```



10 min



Prática

Reescreva o código abaixo, usando a função map(). O resultado final deve ser o mesmo!

```
palavras = 'Minha terra tem palmeiras onde canta o Sabiá, As aves, que aqui gorjeiam, Não gorjeiam como lá.'.split()  
  
resultado = [[w.upper(), w.lower(), len(w)] for w in palavras]  
  
for i in resultado:  
    print (i)
```



Tratamento de Exceções

```
print('Hello')
```

Python

```
File "<ipython-input-1-db8c9988558c>", line 1
  print('Hello')
^
```

```
SyntaxError: EOL while scanning string literal
```

- Observe como obtemos um SyntaxError, com a descrição adicional de que era um EOL (End of Line Error). Isso é suficiente para que percebamos que esquecemos um único apóstrofe no final da linha. Compreender estes vários tipos de erro irá ajudá-lo a depurar seu código muito mais rápido.

Tratamento de Exceções

- Este tipo de erro e descrição é conhecido como uma Exceção. Mesmo que uma declaração ou expressão seja sintaticamente correta, pode causar um erro quando tentamos executá-la. Os erros detectados durante a execução são chamados de exceções e não são incondicionalmente fatais.
- Você pode verificar a lista completa de exceções embutidas (<https://docs.python.org/2/library/exceptions.html>). Vamos aprender a lidar com erros e exceções em nosso próprio código.

Tratamento de Exceções

- A terminologia básica e a sintaxe usadas para lidar com erros no Python são as instruções **try** e **except**. O código que pode causar uma exceção ocorre é colocado no bloco **try** e o tratamento da exceção é implementado no **except** do bloco de código. Sintaxe é:

```
try:  
    Você tenta fazer algo aqui...  
    ...  
except ExceptionI:  
    Se causar a ExceptionI, roda isso.  
  
except ExceptionII:  
    Se causar a ExceptionII, roda isso.  
    ...  
else:  
    Se não causar excessões, roda isso.
```

Tratamento de Exceções

- Agora, e se continuássemos querendo executar código após a ocorrência da exceção? É aí que o ****finally**** entra.
- Usando o finally: o bloco de código sempre será executado, independentemente de existir uma exceção no bloco de código try. A sintaxe é:

```
try:  
    Seu código aqui  
    ...  
    Devido a qualquer exceção, este código pode ser ignorado!
```

```
finally:  
    Este bloco de código sempre seria executado.
```

Datetime

- O Python possui o módulo de datetime para ajudar a lidar com timestamps em seu código. Os valores de tempo são representados com a classe de time. Time têm atributos por hora, minuto, segundo e microsegundo. Eles também podem incluir informações de fuso horário. Os argumentos para inicializar uma instância de time são opcionais, mas é improvável que o padrão de 0 seja o que você deseja.
- O tempo de data também nos permite trabalhar com timestamps de data. Os valores da data do calendário são representados com a classe de data. As instâncias possuem atributos por ano, mês e dia.
- É fácil criar uma data que represente a data de hoje usando o método de classe today().

Lendo e Escrevendo em Arquivos

Expressões Regulares

- Expressões regulares são padrões de correspondência de texto descritos com uma sintaxe formal.
- Muitas vezes você ouvirá expressões regulares referidas como 'regex' ou 'regexp' na conversa.
- As expressões regulares podem incluir uma variedade de regras, a busca de repetição, a correspondência de texto e muito mais. Ao avançar no Python, você verá que muitos dos seus problemas de análise podem ser resolvidos com expressões regulares.

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br



targettrust
treinamento e tecnologia

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br