Princípios da lógica de programação e características da linguagem Python.

Algoritmos

Para desenvolver softwares, aplicativos e diversas outras tarefas na área da tecnologia, escrevemos algoritmos que atendam nosso objetivo. Um algoritmo é uma sequência lógica de passos que devem ser seguidas para resolução de um problema ou realização de uma tarefa. Os algoritmos podem ser descritos de três formas, descrição narrativa, Fluxogramas e pseudocódigo. Vamos entender melhor cada um deles mais a frente.

Podemos descrever ainda um algoritmo como uma receita, pense em uma receita de bolo, temos os ingredientes e um passo a passo a ser seguido rigorosamente para chegar no resultado final que deve ser o bolo. Para se chegar ao bolo desejado, temos que seguir corretamente os passos, caso contrário o bolo pode não sair como desejado.
Por isso a lógica de programação deve utilizar diversos recursos a fim de mitigar possíveis falhas, comportamentos não previstos no algoritmo e consequentemente no programa.

Programa x Algoritmo

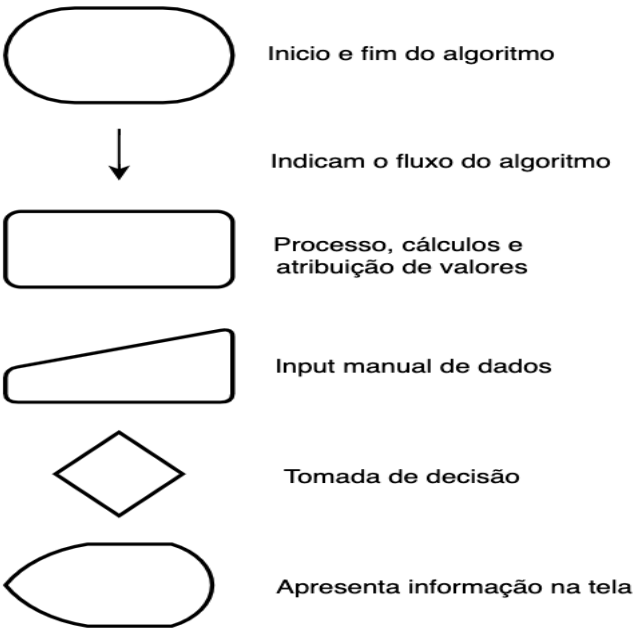
Vimos que um algoritmo é uma sequência lógica de passos que atendem a determinada necessidade. Mas um algoritmo não é um programa de computador por si só, em um programa de computador atendemos não apenas uma, mas várias necessidades, logo, um programa é composto por vários algoritmos interligados ou não que atendem o propósito como um todo.

Descrição narrativa

Esta abordagem descreve os passos do algoritmo usando linguagem natural, ou seja, é como se estivéssemos falando ao computador o que deve ser feito não usando uma linguagem de programação, mas sim nossa própria linguagem de comunicação. É claro que isto não serve para criar um algoritmo ou programa de computador, serve para desenvolver a lógica do algoritmo a ser desenvolvido em uma linguagem de programação.

Fluxograma

Esta abordagem cria uma representação das sequências do algoritmo usando formas geométricas. É uma alternativa visual e muito utilizada para analisar se a lógica desenvolvida faz sentido e atende a necessidade inicial. Vejamos as formas e descrição utilizadas nos fluxogramas.



Pseudocódigo
Esta abordagem é uma representação intermediaria entre a linguagem de computadores e linguagem humana. Também conhecida como Portugol ou português estruturado é um modelo de código semelhante ao código em linguagem de programação.
No pseudocódigo temos uma estrutura que deve contemplar os passos necessários para o funcionamento do algoritmo, veja a análise da estrutura do pseudocódigo abaixo:

Algoritmo: Nome	Nomear o algoritmo.
Var	Área reservada para declaração de variáveis, junto aos seus tipos de dados, por exemplo: Valor: Float, Idade: Int, Nome: Varchar
Início / Start Leia Idade Se idade > 18 Escreva “Maior de 18” Senão Escreva “menor de 18”	Sinaliza o início do algoritmo e contém toda parte principal do algoritmo, regras, lógica do algoritmo estão todas neste bloco de código.
FIM	Sinaliza o fim do algoritmo.

Ao longo do tópico vamos ter exemplos nas três abordagens.

Vamos criar um algoritmo de exemplo, com algo que fazemos com frequência, vamos enviar uma mensagem para uma pessoa pelo nosso aplicativo, passo a passo usando descrição narrativa.

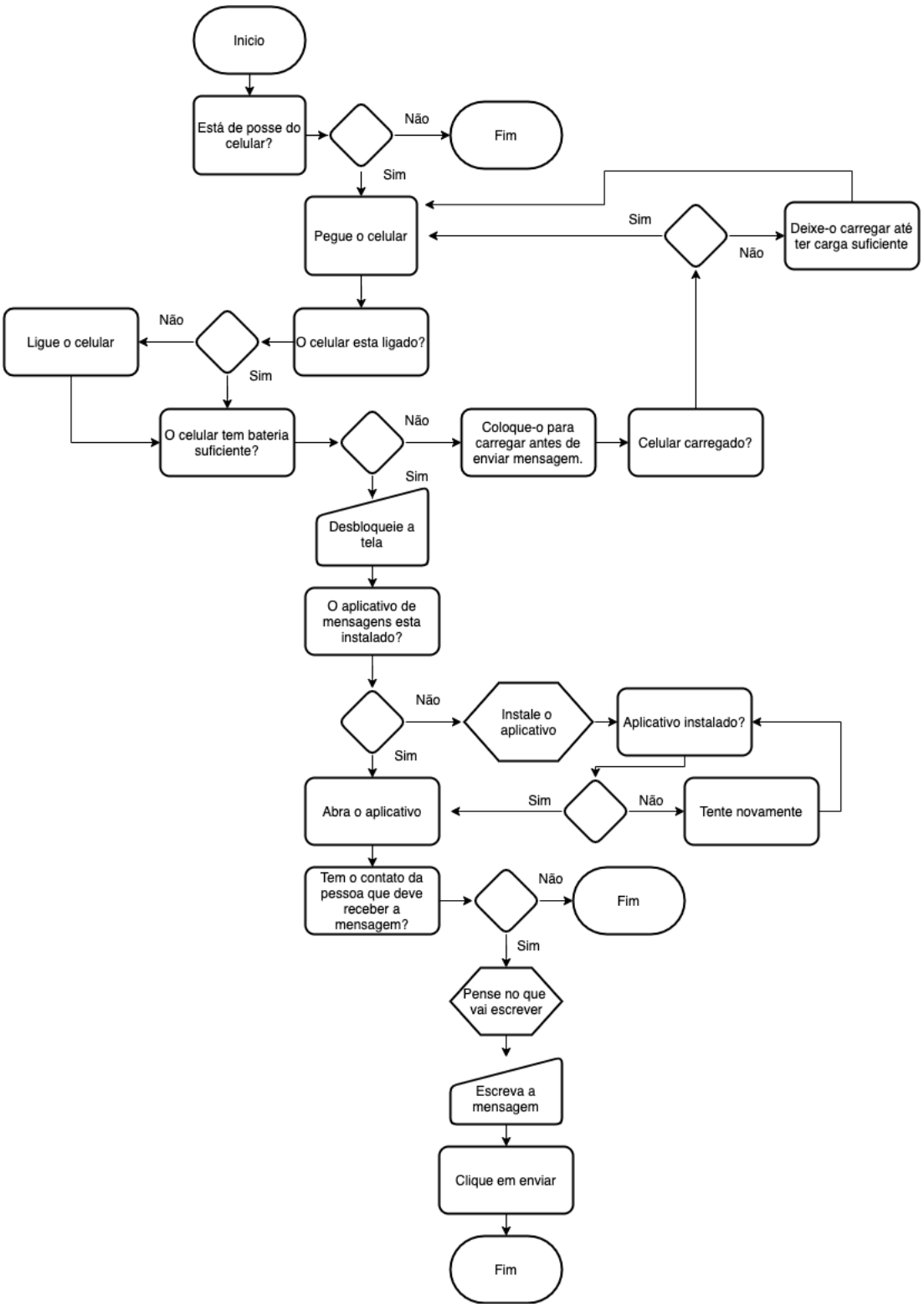
1. Veja se está de posse do celular
2. Veja se está ligado
3. Veja se tem bateria
4. Desbloqueie a tela

5. Encontre o aplicativo de mensagens
6. Abra o aplicativo
7. Encontre o contato da pessoa que vai enviar mensagem
8. Pense no que vai escrever
9. Escreva a mensagem
10. Clique em enviar

Com os passos acima podemos ter situações que não pensamos em como lidar com elas, e se não estivermos de posse do celular? E se o celular não estiver ligado? E se eu tiver perdido o contato de quem estava a mandar mensagem?
Todos estes são exemplos de situações que podem acontecer e quando vamos desenvolver um algoritmo ele deve estar preparado para atender as várias situações, vamos escrever a lista acima novamente usando fluxogramas com tratativas para algumas situações.

Veja na imagem abaixo onde tratamos algumas situações que podem ocorrer ao enviar uma mensagem para alguém, tratamos isso de forma simples em nosso cérebro, planejamos algo e tomamos decisões de como agir conforme surgem imprevistos, no entanto, um programa não pode tomar estas decisões sozinhos, deve ser programado com os passos necessários para entregar o que se propõe.

No exemplo abaixo podemos ter mais passos, mais situações a serem tratadas, mas tornaria o modelo mais complexo e mais longo.



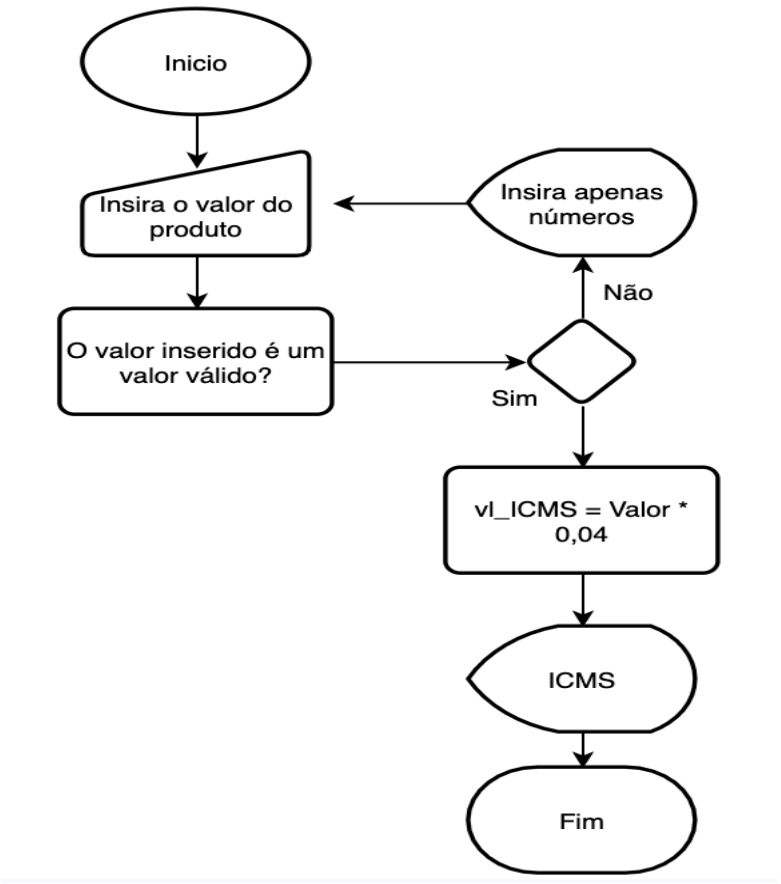
Agora vamos pensar em outro exemplo, um programa que calcula o valor do imposto de circulação de mercadorias e serviços (ICMS) de um produto. Claro que cada município tem suas taxas de impostos, o imposto pode variar de acordo com o enquadramento tributário da empresa entre outras coisas, mas não vamos abordar essa riqueza de detalhes agora, vamos

focar apenas em criar a lógica de um programa que vai receber o valor de um produto e apresentar o valor do ICMS deste produto que vamos definir como 4% neste exemplo. Vamos fazer a descrição narrativa e na sequência o fluxograma e pseudocódigo.

Início

1. Pedir para digitar o valor do produto
2. Calcular o valor do ICMS usando a fórmula ICMS = valor do produto * 0,04
3. Exibir o valor de ICMS na tela

Fim



Algoritmo: ICMS

```
Var
        Valor: Float                    # Declaração de variáveis
Início
        Leia(valor)                     # Lê a variável valor
        Se tipo(valor) = float          # Se o tipo da variável for numérico calcula icms
                Vl_ICMS = valor * 0,04
                Escreva vl_ICMS         # Apresenta valor do ICMS
        Escreva "Insira apenas números" # Aviso de inserir apenas números
Fim algoritmo                           # Fim do algoritmo
```

Os computadores não compreendem instruções como as escritas nos exemplos anteriores, para isso eles usam linguagem binária, ou seja, zero e um. Mas não precisamos programar em

binário para fazer um programa de computador, para isto existem linguagens de computador que permite que criemos nossos algoritmos e programas.
Temos diversas linguagens de programação no mercado, cada uma com suas particularidades, mas todas exigem um mesmo requisito na hora de criar um programa, coerência lógica dos passos desenvolvidos nos algoritmos, esse conjunto todo chamamos de programa.

Temos um fluxo na hora de pensar em um programa de computador, temos as entradas de dados, o processamento e em alguns casos uma saída.

Variáveis e constantes

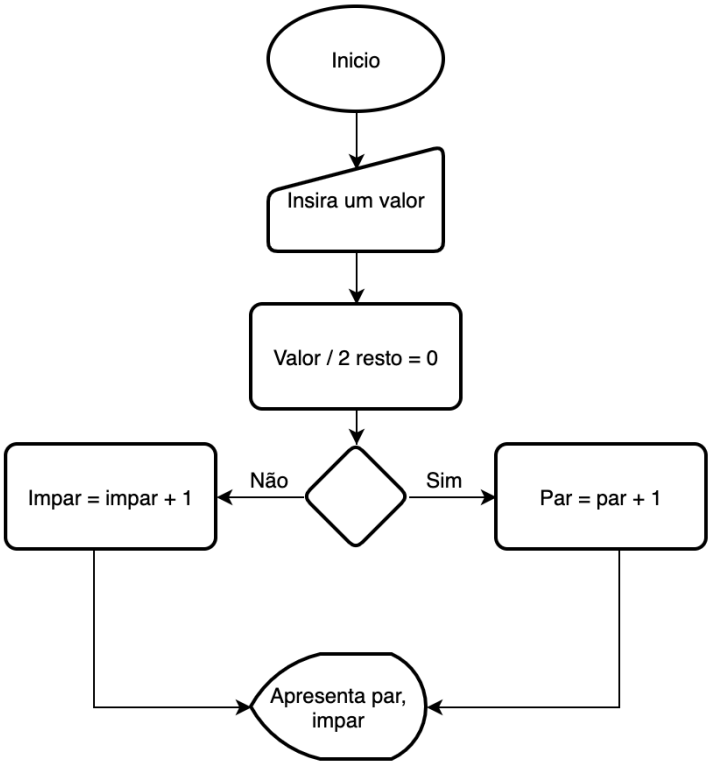
No exemplo acima do cálculo de ICMS, tivemos que armazenar o valor do imposto em uma variável chamada ICMS, é muito comum termos que armazenar resultados que podem ou não sofrer mutação ao longo da execução do algoritmo, e para isto usamos variáveis e constantes.

Uma variável funciona como um repositório de dados que atende a aquela execução, dentro de um programa. Um dado pode ser variável ou constante, em outras palavras mutável ou não mutável. Um dado variável é aquele que pode sofrer alteração durante a execução do programa, um dado constante é aquele que vai manter o mesmo valor do início ao fim da execução do programa.

Para exemplificar, vamos criar um algoritmo que receba vários números, veja se eles são pares ou ímpares, e apresente a quantidade de números pares e a quantidade de números ímpares.
A descrição narrativa ficaria assim:

1. Pedir para digitar os números
2. Checar se número é par ou ímpar usando a fórmula (se número / 2 resto = 0, par = 1 senão ímpar = 1)
3. Apresentar conteúdo das variáveis par e ímpar

Fluxograma



Pseudocódigo

```
Algoritmo: Testa par e ímpar
Var
        Número, par, ímpar: inteiro                 # Declaração de variáveis
Início
        Leia(número)                                # Lê a variável número
        Se numero / 2 mod 0                         # Testa se número é par
                Par = 1                             # Contabiliza um caso seja par
        Senão
                Ímpar = 1                           # Contabiliza um caso seja ímpar
        Escreva par, ímpar                          # Apresenta variável par e ímpar
Fim algoritmo                                       # Fim do algoritmo
```

Estruturas de repetição - Loops

Uma estrutura de repetição é uma instrução que se repete até que determinada condição seja atendida. Vamos usar o exemplo de algoritmo de número par ou ímpar, neste caso vamos percorrer os valores entre 1 e 10, queremos testar todos estes números e ver qual é par e qual é ímpar. Precisaremos de um loop que percorra os números até 10 e submeta cada um deles ao algoritmo que criamos no passo anterior. Nossa variável de contagem do loop inicia com valor 1, este valor é submetido ao algoritmo que checa se é par ou ímpar, faz a atribuição na variável par ou ímpar e na sequência atribui mais um ao contador, que passa a valer 2, e assim sucessivamente até chegar a 10. Neste caso temos outra particularidade, note que ao atribuir o valor as variáveis par ou impar adicionamos o valor da própria variável e somamos 1 para que seja contabilizado o total e não atribuído o valor 1 a variável. Vamos fazer o pseudocódigo deste exemplo:

```
Algoritmo: Loop Teste par ou ímpar
Var
        Número, par, ímpar: inteiro                 # Declaração de variáveis
Início
        Número = 1                                  # Atribuição de valores

Para i de 1 até 10 faça                             # Loop de 1 a 10
        Leia(número)                                # Lê a variável número
        Se numero / 2 mod 0                         # Testa se número é par
                Par = par + 1                       # Soma mais um caso seja par
        Senão
                Ímpar = ímpar + 1                   # Soma mais um caso seja ímpar
Fim para                                            # Fim da estrutura de repetição
Escreva par, ímpar                                  # Apresenta variável par e ímpar

Fim algoritmo                                       # Fim do algoritmo
```

Um bom exemplo utilizando tanto loop quanto variáveis que podemos treinar é a sequência de Fibonacci (0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 ...). Esta é uma sequência cujo novo número é a soma dos dois números anteriores. Vamos implementá-la usando pseudocódigo abaixo:

```
Algoritmo: Fibonacci
Var
        Anterior, atual, próximo, i: Inteiro        # declaração de variáveis
Inicio
Anterior = 0                                        # Atribuição de valores
Atual = 0
Próximo = 1

Escreva (“Vinte primeiros números de Fibonacci”)
Escreva (atual)                                     # Apresenta valor atual (0)
Para i de 1 ate 20 faça                             # Loop de 1 a 20
        Escreva(próximo)                            # Apresenta próximo número
        Anterior := atual                           # Anterior recebe numero atual
        Atual := próximo                            # Atual recebe próximo numero
        Próximo := atual + anterior                 # Próximo recebe a soma
Fim para                                            # Fim da estrutura de repetição
Fim algoritmo                                       # Fim do algoritmo
```

Python

A linguagem python foi criada em meados de 1990 por Guido Van Rossum, é uma linguagem open Source que pode ser utilizada inclusive para fins comerciais.

Por se tratar de uma linguagem dinâmica, interpretada, robusta, multi-plataforma, orientada a objetos e de propósito gerais é uma linguagem que sua adoção cresceu muito nos últimos anos, principalmente por seu papel no treinamento de modelos de machine learning e deep learning.

Como é uma linguagem de alto nível não precisamos de conhecimento em outras linguagens para começar a programar em python, a lógica de seu programa pode ser aplicada diretamente no código em python, facilitando o aprendizado de quem ainda não tem grande experiência com programação e facilitando a vida dos mais experientes, uma vez que que exige menos código em relação a algumas linguagens de programação.

Seu interpretador é escrito em C++, ou seja, em uma linguagem muito difundida na área de tecnologia, desta forma, python pode ser implementado em qualquer plataforma que tenha o compilador, o que não é difícil pois existem compiladores nativos ou portados para quase todas as plataformas atuais.

Por se tratar de uma linguagem interpretada, é possível o desenvolvimento majoritariamente em linha de comando, mas com a possibilidade de escrever código e analisar seus resultados em ferramentas com interface mais amigável como Jupyter notebook, PyCharm e VSCode.

Um detalhe interessante é que Python pode ser utilizado desde aplicações desktop, servidor e aplicativos. Python ganhou um espaço importante na área de big data e data Science, em ambientes big data há uma versão do python com Spark chamada Pyspark muito utilizada para ingestão e análise de dados.

Há diversas bibliotecas para python, bibliotecas para aplicações matemáticas como numpy, trabalhar com datasets usando pandas, criar gráficos com matplotlib, seaborn, além de frameworks para uso em inteligência artificial como Tensorflow, Pytorch, keras entre outros.

O python é distribuído em duas versões, Python 3 e Python 2, não há grandes diferenças entre as versões, mas neste tópico vamos trabalhar com Python 3.7.

Para começarmos vamos instalar o Python que pode ser baixado através da página oficial https://www.python.org/downloads/ sua instalação é simples e intuitiva, basta seguir os passos abaixo:
1. Selecione o instalador de acordo com seu sistema operacional.

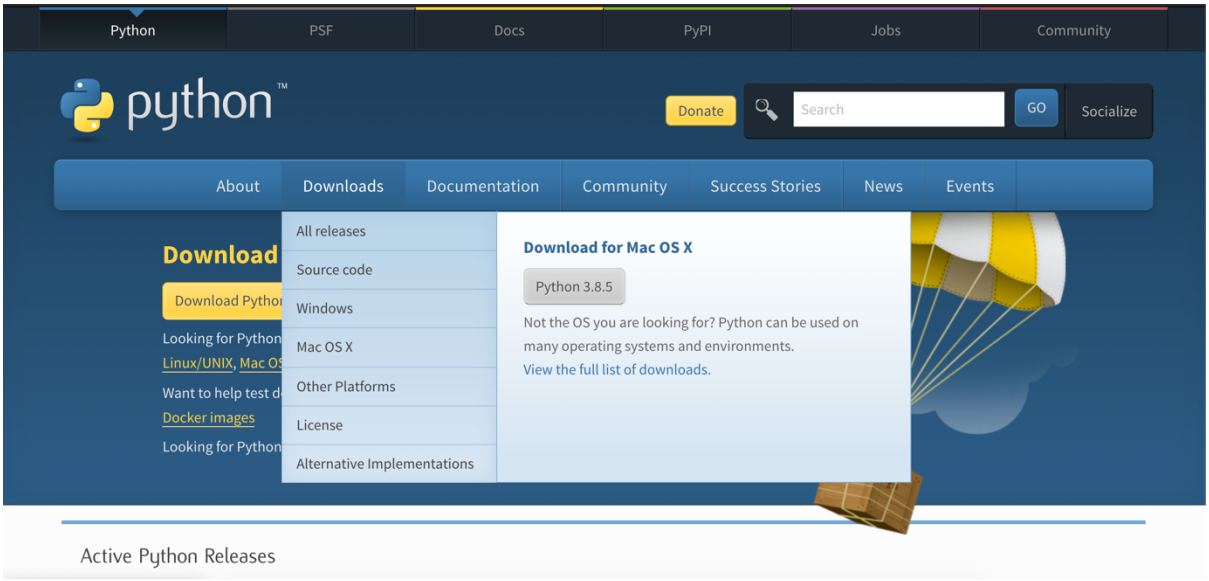
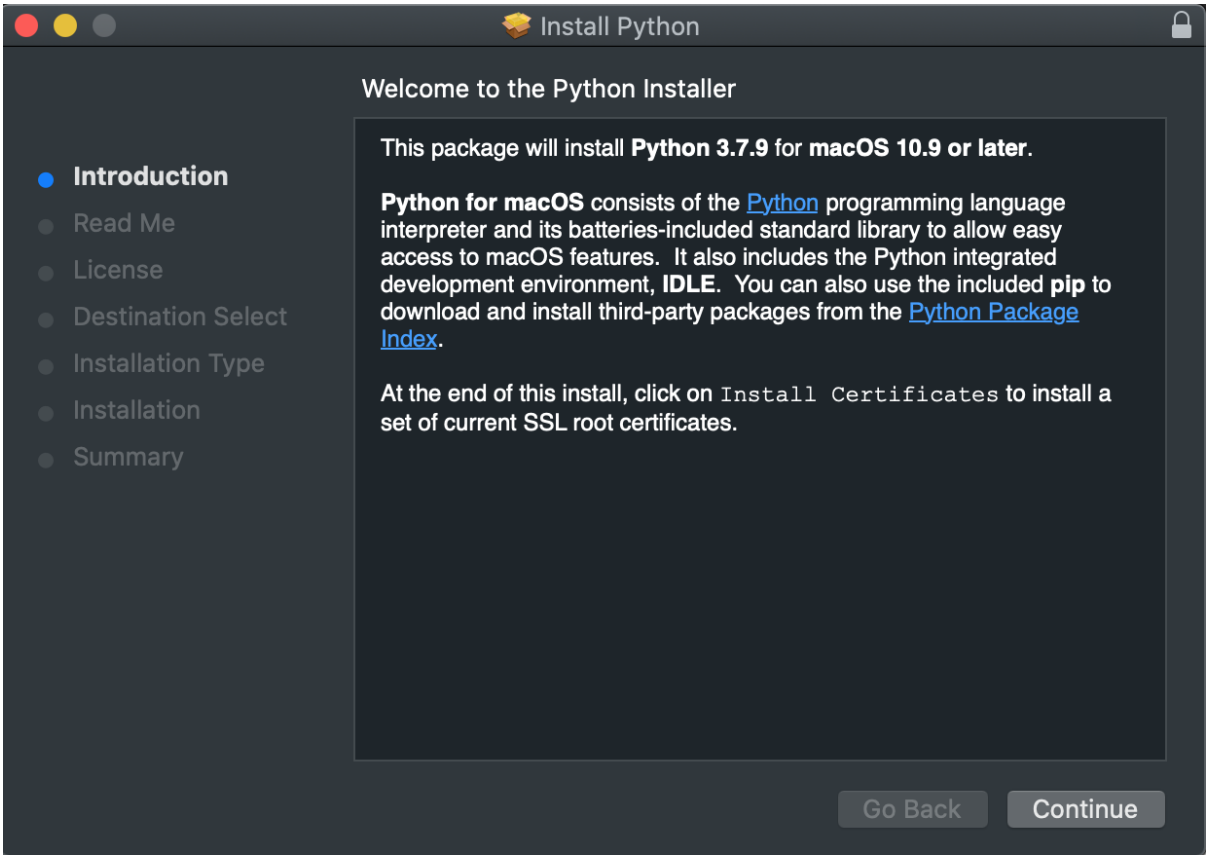
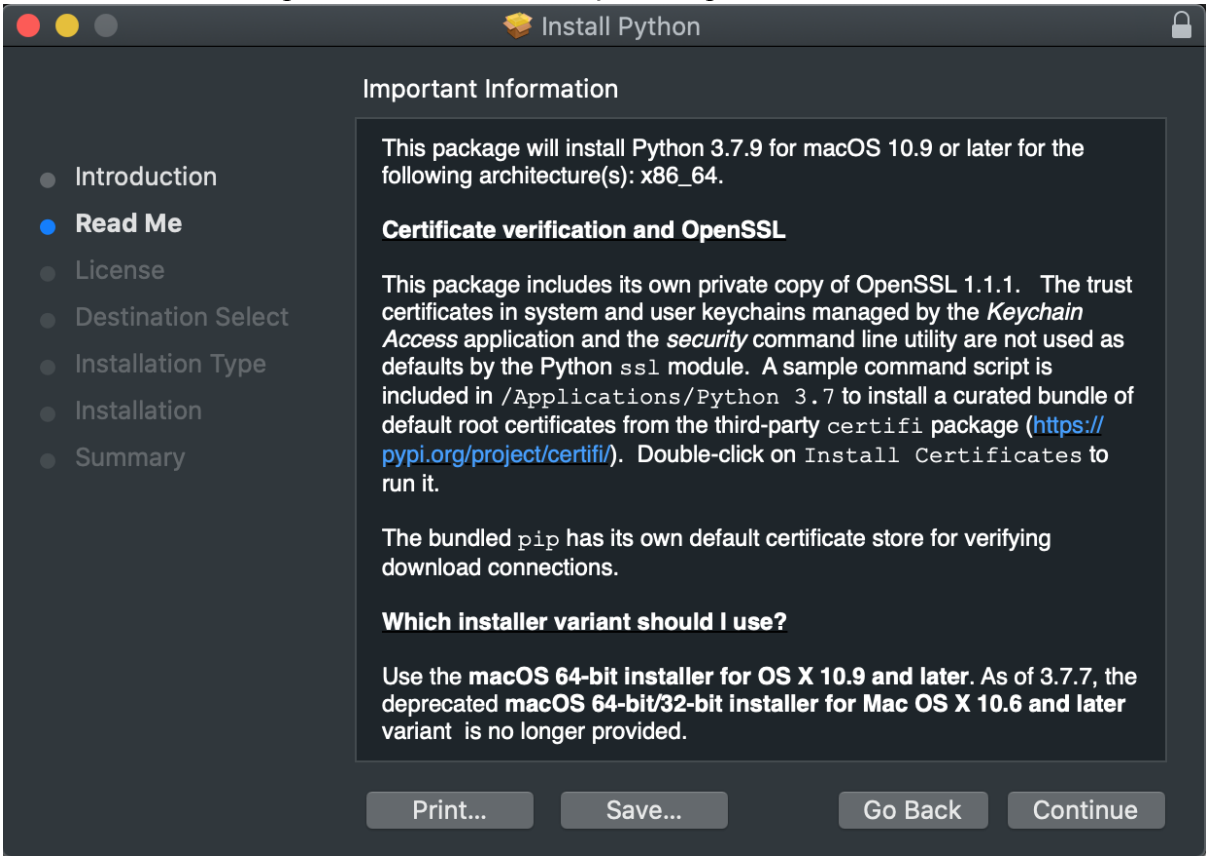


Imagem: Download Python
Fonte: https://www.python.org/downloads/

2. Execute o instalador e esta tela deve ser apresentada, clique em continue.

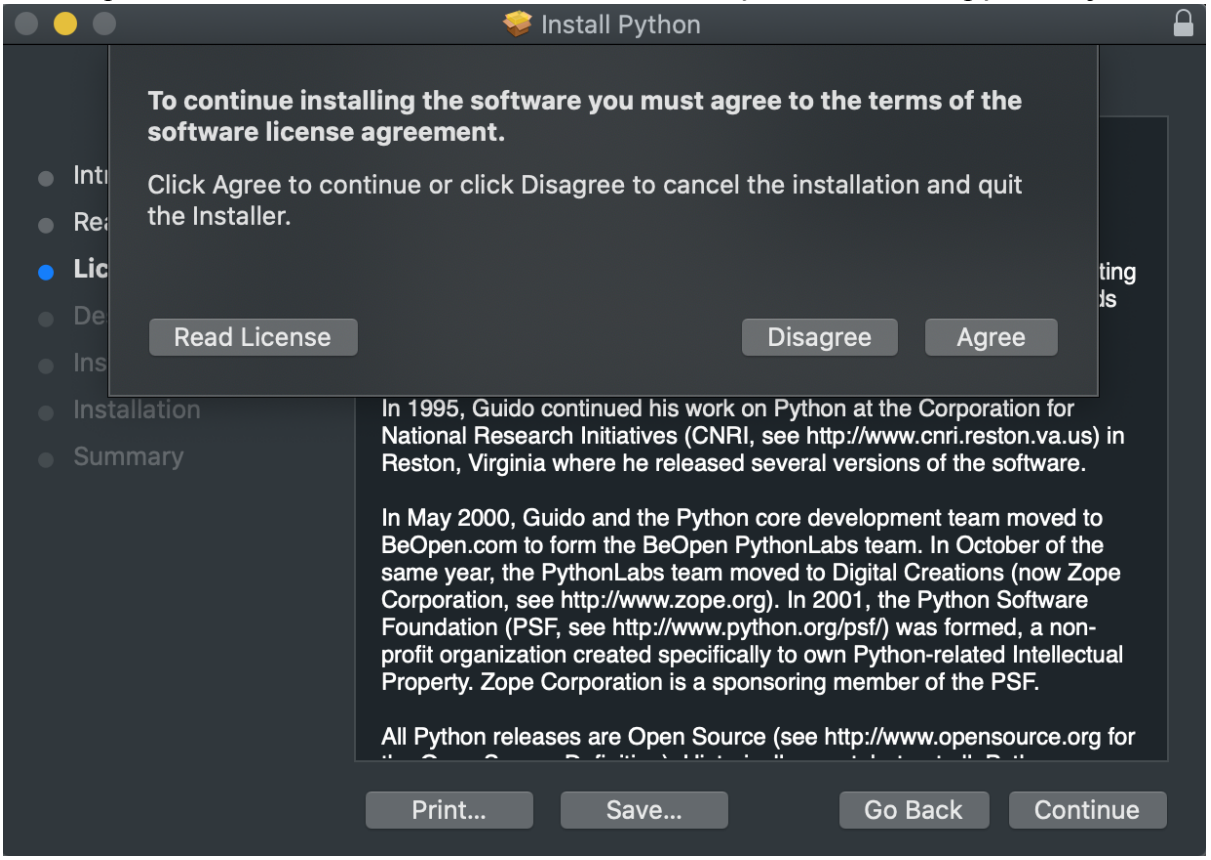


3. A tela abaixo será apresentada com informações, clique em Continue.

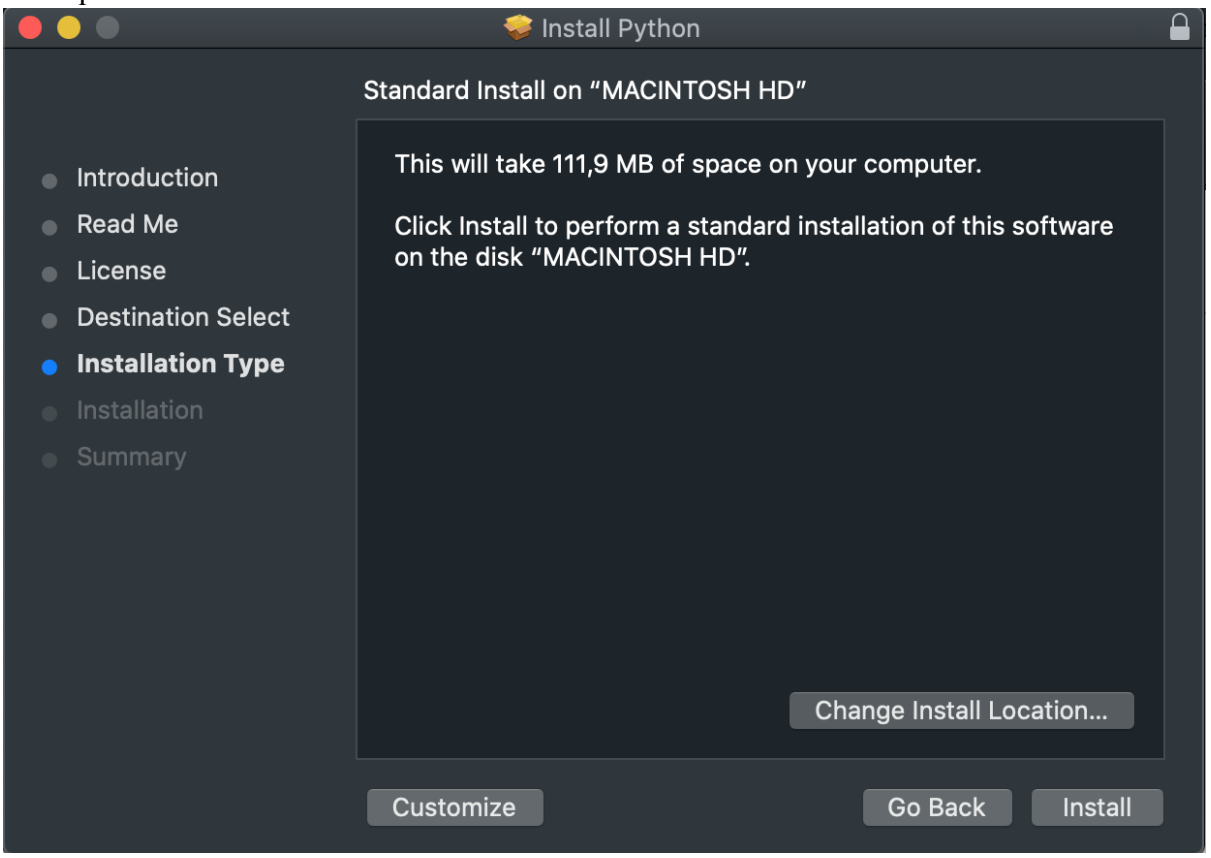


4. A tela de licenciamento será apresentada, clique em Continue.

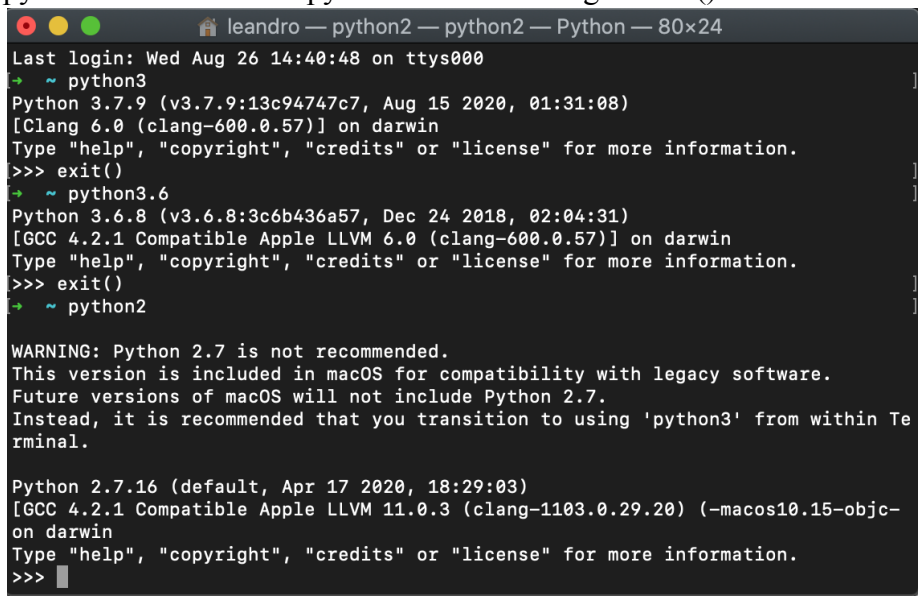
5. Será questionado de concorda com os termos de instalação, selecione a opção desejada.



6. Clique em install na tela final.



Com o python instalado vamos conhecer na prática mais desta linguagem de programação. Abra o terminal do seu computador e digite python3, conforme imagem abaixo.
Caso haja outras versões instaladas no seu computador, podemos executar uma versão especifica utilizando a versão após a palavra python, como: python3, python2, python3.6, python3.5. Para sair do python no terminal digite exit ().



Neste tópico aprendemos um pouco sobre a lógica necessária para desenvolvimento de algoritmos, entendemos sobre programas e pudemos conhecer um pouco sobre as características da linguagem de programação Python, bem como sua instalação.
Este é um tópico extenso, com bastante conteúdo, mas desenvolver o pensamento lógico para desenvolvimento de algoritmos é de suma importância.

Quiz:

1. Seleciona a alternativa correta sobre um programa de computador:
    a. Um programa de computador é um algoritmo que faça qualquer coisa.
    b. Um programa é uma parte de um algoritmo.
    c. **Um programa é composto por vários algoritmos interligados ou não que atendem o propósito como um todo**
    d. Nenhuma das anteriores.

2. Com linguagem Python não é possível desenvolver soluções mobile. Esta afirmação é:
    a. **Falso**
    b. Verdadeiro

3. Um algoritmo antes de ser desenvolvido em linguagem de programação, pode ser representado por:
    a. Descrição narrativa apenas
    b. Descrição narrativa, Fluxogramas e pseudocódigo
    c. Apenas Pseudocódigo
    d. Apenas por linguagem de programação

4. Um pseudocódigo pode ser executado em um computador e funcionará normalmente, desde que a lógica esteja correta.
    a. Verdadeiro
    b. **Falso**

Referências:

PYTHON SOFTWARE FOUNDATION. Documentação Python 3.7.8. Python ORG, 2020. Disponível em https://docs.python.org/pt-br/3/index.html. Acesso em: 25 ago. 2020.

DAURICIO, Fernanda Schiavetto; et el. Algoritmos e Programação Contextos e Práticas. In DAURICIO, Fernanda Schiavetto: Algoritmos e Lógica de programação. Cap 1, 2 e 3.Disponível em https://www.academia.edu/30062425/LIVRO_ALGORITIMOS_LOGICA_E_PROGRAMACAO. Londrina: Academia.edu. 2015. Acesso em 25 ago. 2020.