INTRODUÇÃO A PROGRAMAÇÃO COM PYTHON

Sumário

INTRODUÇÃO

Olá, seja bem-vindo ao nosso curso introdutório de programação utilizando uma linguagem chamada Python. Neste curso você irá começar a aprender a criar programas de computador. Mas isso não será como nos outros cursos nos quais você faz seu programa e depois executa ele no computador. Aqui você irá ter atividades complementares que poderão ser realizadas na própria plataforma do curso e desafios para possa lhe auxiliar.

Vamos começar então falando um pouco sobre algoritmos, linguagens de programação e sobre a linguagem Python. Em seguida iremos para a parte prática, onde você irá aprender a programar.

Vamos começar nosso estudo então?

Lógica de Programação

O QUE É ?

Lógica de Programação é o modo como se escrevem programas de computador através de uma sequência de passos para executar uma ou várias funções, esta sequência é o algoritmo.

MAS O QUE SERIA UM ALGORITMO?

Algoritmo é basicamente uma “receita” para executar uma tarefa ou resolver um problema, ele tem começo meio e fim, assim como uma de receita de bolo, se você seguir os passos descritos, terá um resultado. Assim como na atividade de fazer um bolo, utilizando o algoritmo, você pode fazer atividades repetitivas ou ao mesmo tempo, pode delegar funções ou esperar um passo ser concluído para iniciar outro, desde que seja algo finito existem várias possibilidades.

PORQUE É BOM DESENVOLVER UMA LÓGICA PROGRAMACIONAL?

Quando você aprende lógica de programação, sua mente se desenvolve e você cria um perfil analítico que é muito valorizado no mercado, além disso, você desenvolve a habilidade de aplicar a lógica em outras atividades do dia a dia e descobre o valor da persistência. Entre os talentos que você pode adquirir, estão: a habilidade em escrita, pensamento crítico, trabalho em equipe e raciocínio lógico.

ALGORITMOS

Entender o conceito de algoritmo não é complicado. Se você já realizou alguma tarefa seguindo um “passo a passo”, então já utilizou um algoritmo. Um algoritmo descreve, na forma de texto ou fluxograma, as etapas para realização da tarefa que você queira executar. Nesse sentido, na computação, o algoritmo serve para organizar as etapas de um processo.

Imagine a fabricação de um bolo como sendo a tarefa que você deve executar. A receita do bolo que você segue para fazê-lo seria o algoritmo. Os ingredientes seriam os dados de entrada, uma vez que eles seriam os componentes que você precisa para fazer o bolo dar certo. Os recipientes utilizados para fazer o bolo são as variáveis envolvidas na fabricação, pois como o próprio nome já diz, elas variam de acordo

com a necessidade do programa, no caso, a receita do bolo. Por exemplo, imagine que você deseje fazer um bolo redondo, para isso precisa de uma forma redonda, então, o formato do bolo varia de acordo com a forma. E o "modo de fazer" consiste no passo a passo para executar a tarefa.

Em computação podemos definir um algoritmo como sendo uma forma genérica de se representar o passo a passo que o programa deve seguir para sua conclusão e solucionar o problema que ele foi construído para isso. Um algoritmo quando é escrito em uma linguagem que o computador entende (as chamadas linguagens de programação) se torna então um programa de computador.

A seguir vemos dois exemplos de algoritmos para problemas do dia-a-dia (que não são de

RECEITA DE BOLO SIMPLES

Modo de Preparo:

1. Bata as claras em neve;
2. Reserve;
3. Bata bem as gemas com a margarina e o açucar;
4. Acrescente o leite e farinha aos poucos sem parar de bater;
5. Por ultimo agregue as claras em neve e o fermento;
6. Coloque em forma grande de furo central untada e enfarinhada;
7. Asse em forno médio, preaquecido, por aproximadamente 40 minutos;
8. Quando espetar um palito e sair limpo estará assado.

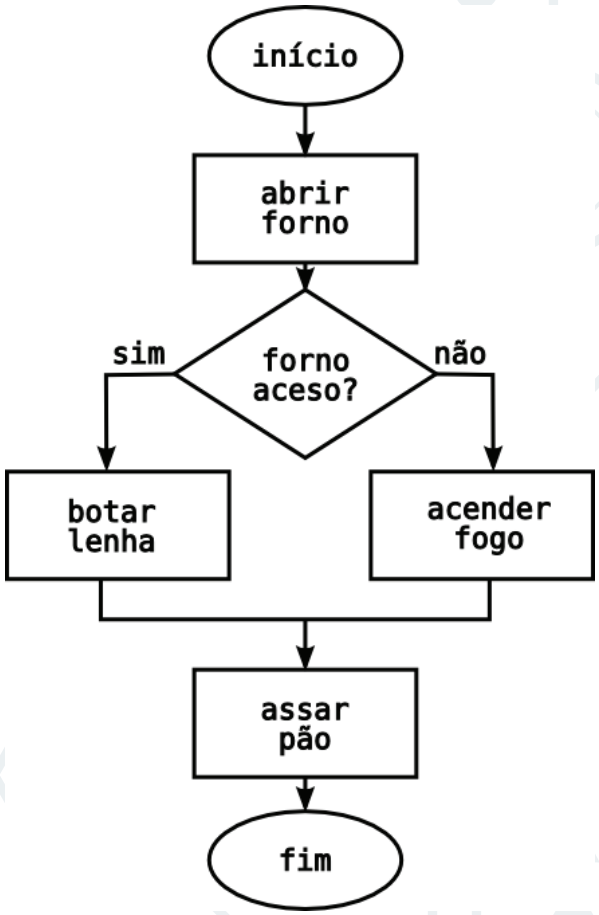

Figura 2: Algoritmo em forma de fluxograma.

Características de um algoritmo

- Um algoritmo sempre termina, ele deve ter um fim;
- As instruções do algoritmo devem estar em uma seqüência lógica, ou seja, deve existir uma ordem de execução dos passos da seqüência;

- Cada ação é descrita precisamente e sem ambigüidades, ou seja, o algoritmo não pode dar margem à dupla interpretação;
- Um algoritmo sempre produz um ou mais resultados (saídas), podendo não exigir dados de entrada;

DESAFIO:

Monte um algoritmo para trocar uma lampada.

LINGUAGEM DE PROGRAMAÇÃO

Quando programamos um computador não podemos escrever utilizando uma linguagem natural ao ser humano, pois, esse tipo de linguagem o computador não é capaz de entender. O computador só entende 0 ou 1. Mas não criemos pânico, não é necessário escrever programas na linguagem do computador (usando apenas 0 e 1). Para programar um computador usamos uma linguagem intermediária, mais próxima de nossa linguagem natural, que possa ser utilizada para comunicar-se com a máquina e assim enviar os comandos que queremos que ela faça (ou seja, programar o computador). A essa linguagem damos o nome de linguagem de programação.

Você sabia que o computador não tem inteligência? Quando queremos programar um computador nós precisamos dizer para o computador o que fazer. Ele é na verdade um grande circuito que pode realizar algumas tarefas, que chamamos de instruções. É a combinação dessas instruções que faz com que ele execute o que queremos. Então a inteligência não está no computador e sim nas pessoas que escrevem os programas que executam nos computadores.

Para se programar um computador primeiro precisamos criar o algoritmo com o passo a passo para a solução de nosso problema. Em seguida temos que escrever esse algoritmo de forma que o computador entenda, e para isso usamos as linguagens de programação.

Uma linguagem de programação define o os comandos (instruções) que podemos utilizar e as regras de como escrever esses comandos. A partir daí podemos então escrever um programa de computador. Então um programa é uma sequência ordenada desses comandos que fazem com que o computador execute a função que queremos. Você pode programar em muitas linguagens de programação dependendo do tipo de problema que você quer resolver. Alguns exemplo de linguagens de programação que são usadas atualmente são: Python, Java, C, C++, dentre outras. Mas no nosso curso iremos dar foco na linguagem Python.

Figura 3: Linguagem de programação Python

Figura 4: Linguagem de programação Java

```c
#include <stdio.h>
int main(){

        int i;
        char letra;
        for(i=1;i<=10;i++)
                printf("%d\n",i);

        for(letra = 'A';letra<='Z';letra++)
                printf("%c\n",letra);
                return(0);

}
```

Figura 6: Linguagem de programação C

Tutorial de intalação

Ambiente Python

Para aprendermos a linguagem Python de forma prática, precisamos preparar o nosso ambiente de trabalho, que nada mais é do que o conjunto de ferramentas que precisamos para escrever e executar códigos em Python. Devemos instalar na nossa máquina o interpretador Python 3. Se você estiver utilizando um sistema operacional Linux ou Mac não será preciso instalar nada porque o python já vem instalado nessas plataformas, mas se você está utilizando um ambiente Windows basta seguir as etapas que iremos apresentar.

Primeiro, digite no seu navegador ou copie e cole esse endereço: https://www.python.org
Essa janela irá se abrir:

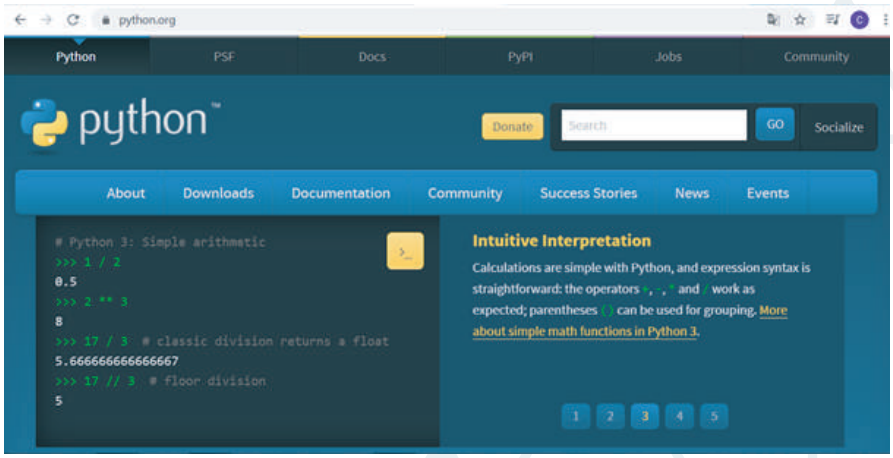

Figura : Página Oficial do Python

Coloque o cursor do mouse sobre o botão Downloads e selecione a opção Windows, dê um click sobre o botão, assim:

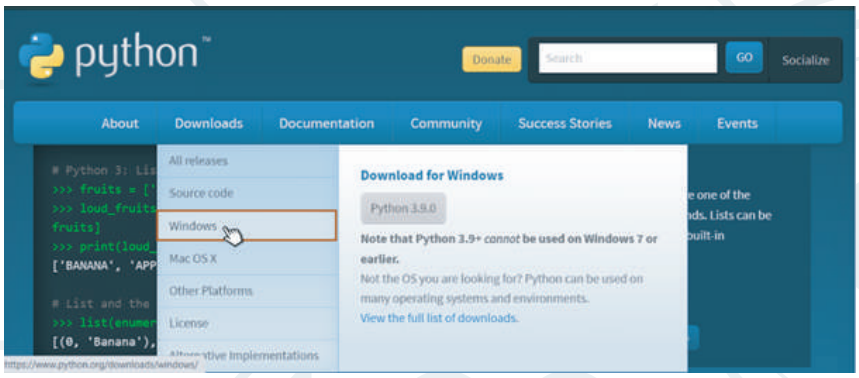

Figura : Página Oficial do Python

Agora selecione a primeira opção conforme a imagem.

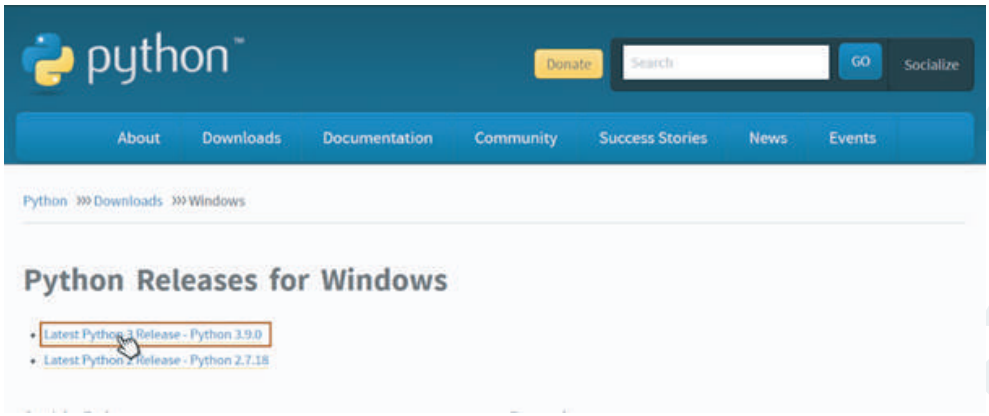

Figura : Página Oficial do Python

Essa tela se abrirá para você:

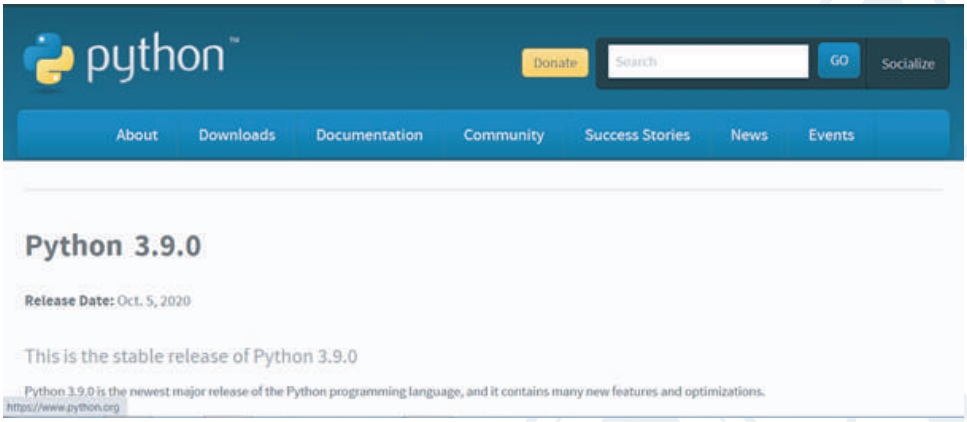

Figura : Página Oficial do Python

Role a página até encontrar a seguinte nomenclatura: Windows x86-64 executable installer, click nela para iniciar o download.

Figura : Página Oficial do Python

Com o download concluído, é hora de realizar a instalação, abra a pasta onde você baixou o arquivo e dê dois cliks com o mouse, uma janela irá se abrir, click em “executar” conforme a imagem.

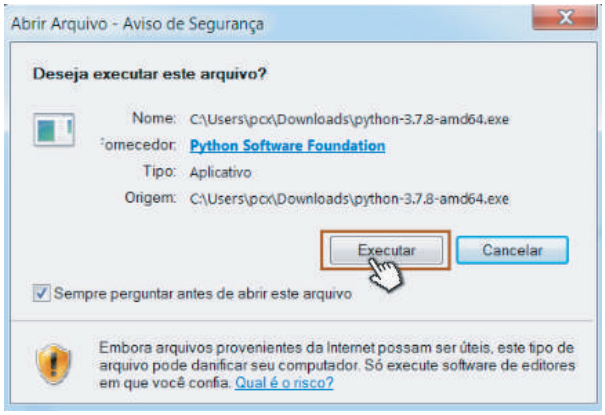

Figura : Página Oficial do Python

Na próxima janela click em “Install Now”.

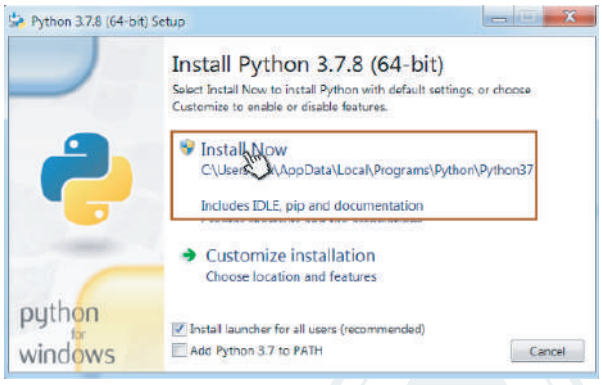

Figura : Página Oficial do Python

Aguarde a instalação ser concluída.

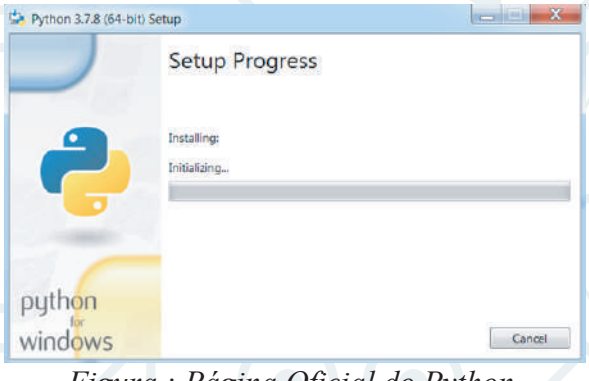

Figura : Página Oficial do Python

Após o programa instalar suas dependências, uma nova janela será exibida informando que a instalação foi realizada, click em “Close” para fechar o instalador.

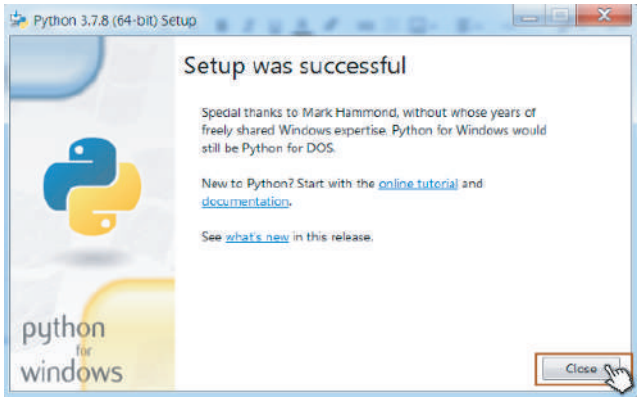

Figura : Página Oficial do Python

Pronto! Agora você já tem as ferramentas básicas para iniciar seus estudos utilizando a linguagem python. Para facilitar um pouco a nossa vida, vamos criar um atalho do IDLE Python na nossa área de trabalho.

Primeiro click no botão iniciar do seu windows, pesquise por python, coloque o cursor do mouse sobre a opção: IDLE (aqui haverá a informação da sua versão do python, conforme a imagem ilustrativa), click com o botão direito do mouse, vá até “Enviar para” e selecione “Área de trabalho”, é só clicar e está

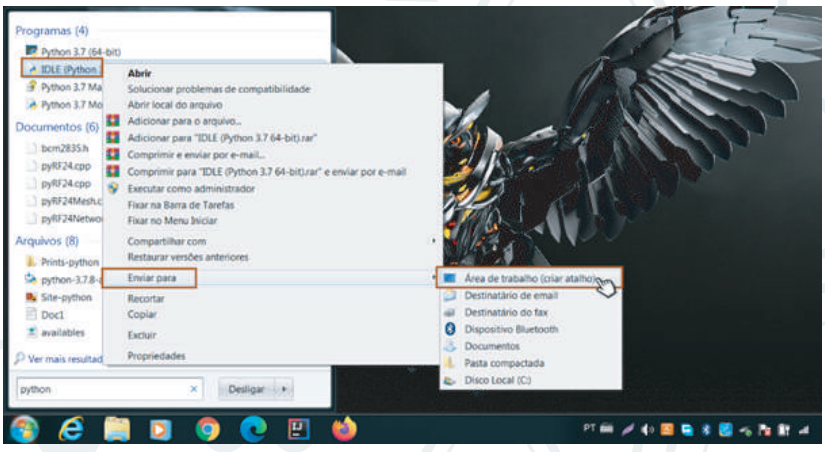

Figura : Página Oficial do Python

Agora você já pode acessar o IDLE do python pelo ícone da sua área de trabalho.feito.

Programação

Variáveis

Uma variável é um espaço na memória do computador destinado a um dado que é alterado durante a execução do programa. Para funcionar corretamente, as variáveis precisam ser definidas por nomes e tipos. Veja os diferentes tipos de dados:

Tipo	Descrição	Exemplo
Int(inteiro)	LED (cores diferentes)	0,1,2,3….
float (Real)	Resistor 220 Ω	0; 1,2
char(Caracteres)	Placa Arduino	a,b,2,*,#
String (texto)	Números, letras,símbolos e texto	Oi meu nome é João
Booleano (Lógico)	Comandos de comparação	Verdadeiro ou Falso True or False

Você pode imaginar que as variáveis são como “caixas” destinadas a guardar algo mutável ao longo do tempo. Pense em uma aplicação que trabalha com a idade dos usuários cadastrados. Cada um colocará um número inteiro diferente, certo?

Então para criar uma variável em Python basta escrever seu nome e atribuir um valor a ela. Por exemplo: `idade = 30`

Python usa o sinal de “=” para atribuir um valor a uma variável. Um detalhe importante da linguagem python é que não precisamos declarar qual o seu tipo antecipadamente, pois ela atribui automaticamente a variável como texto. Então para criar qualquer variável basta realizar a seguinte estrutura: `Nome_da_vairável = valor`

Então vamos ver alguns exemplos de criação de diferentes tipos de variáveis:

```
idade = 30

nota = 8.9

nome = "João"
```

Figura : Exemplos de váriaveis

Para variáveis do tipo float devemos utilizar ponto em vez de vírgula para separar as casas decimais. Já para os caracteres ou textos utilizamos aspas simples ou duplas ‘’ “”.

Podemos realizar operações básicas utilizando variáveis como valores, veja as possíveis operações na tabela abaixo:

Tipo	Descrição
Soma	x + 2
Subtração	10 - x
Multiplicação	x * 28
Divisão	4/x
Resto da divisão inteira	x%5
Potência	x**2

OBS: As regras de precedência são as mesmas da matemática: primeiros parênteses, depois multiplicação, divisão e potência, e só depois soma e subtração

Caso as operações de divisão inteira e resto da divisão inteira esteja um pouco confusa, temos uma imagem que pode ajudar a melhorar o entendimento:

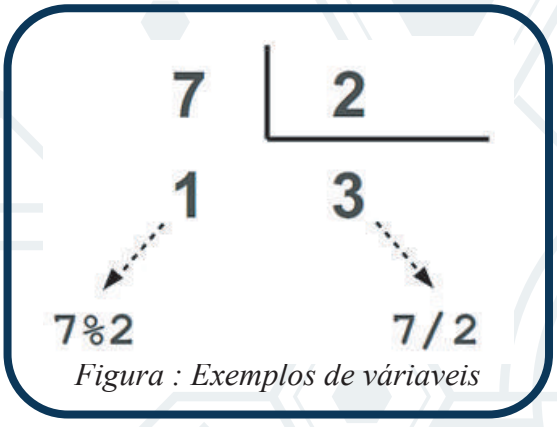

Figura : Exemplos de váriaveis

Podemos ainda realizar junções de operadores matemáticos simples com o símbolo de atribuição da variável, quando queremos somar, dividir, multiplicar um valor a ela, por exemplo.

Operação	Exemplo	Expressão Equivalente
+=	x += 1	x = x + 1
–=	x –= 5	x = x – 5
*=	x *= 2	x = x * 2
/=	x /= 4	x = x / 4
%=	x %= 2	x = x % 2
**=	x **=3	x = x ** 3

Estrutura sequencial e teste de mesa

Vimos anteriormente que o computador executa uma instrução por vez, seguindo a sequência estabelecida pelo desenvolvedor. Então é de grande importância que o programador consiga entender como seu programa funciona e o que devemos esperar na saída, para isso podemos utilizar um método muito prático e útil para entendermos como um programa de estrutura sequencial funciona. Esse método é chamado de teste de mesa, ele pode ser usado para simular a execução de um programa e detectar possíveis falhas no código e consiste em uma tabela representando o conteúdo da memória após o processamento de cada instrução do programa.

Um pouco confuso? vamos ver alguns exemplos para nos ajudar a entender melhor como é e como aplicar:

Para realizar um teste de mesa, primeiro criamos uma coluna para cada variável; depois criamos uma linha para cada instrução do nosso programa e por fim preenchemos as lacunas com o valor de cada variável após cada instrução.

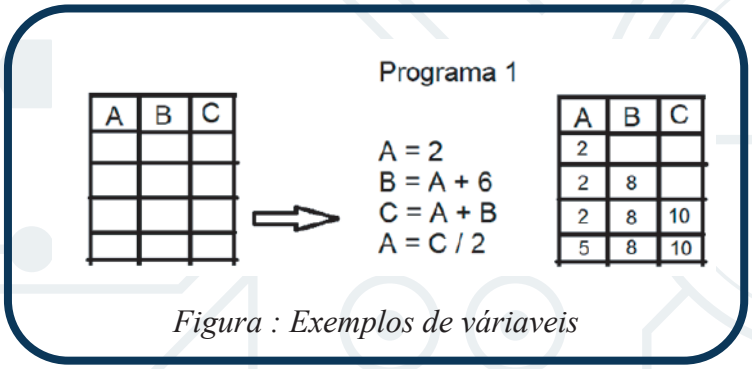

Figura : Exemplos de váriaveis

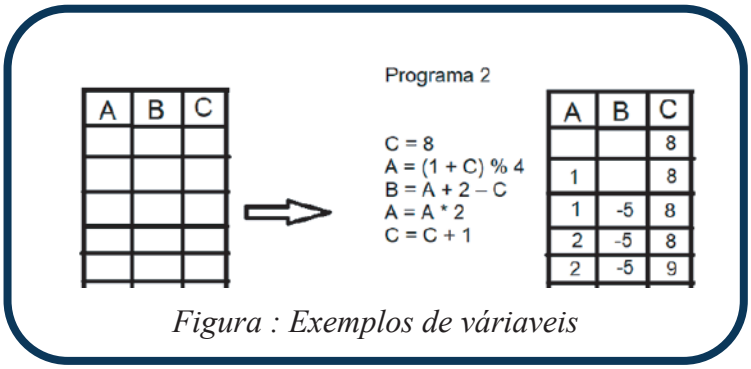

Figura : Exemplos de váriaveis

Claro que esse teste de mesa funciona em casos específicos, pois vamos ter casos onde nossos códigos vão superar a margem de 50 linhas facilmente, logo não é viável aplicar o teste de mesa em códigos extensos.

Entrada e Saída

Todo programa de computador pode ser dividido em três etapas: Entrada, processamento e saída. Onde:

• **Entrada:** São os dados de entrada do algoritmo, os valores que ele vai usar no processamento para chegar à solução do problema. Esses dados de entrada, geralmente, são fornecidos pelo usuário, fazendo uso de algum dispositivo de entrada de dados, tal como, um teclado, um mouse ou um leitor de código de barras.

Figura : Exemplos de váriaveis

• **Processamento:** São os procedimentos utilizados para chegar ao resultado final, para alcançar a resolução do problema. Esses procedimentos são executados no processador do computador e os valores parciais são armazenados na memória (memória RAM).

• **Saída:** São os dados já processados. É o resultado do processamento. Estes dados de saída são apresentados em algum dispositivo de saída. Por exemplo, a tela de um monitor.

Figura : Exemplos de váriaveis

Sempre que pensarmos em desenvolver um programa precisamos identificar e pensar nessas três etapas. Se identificamos essas fases ficará extremamente fácil começar a sequência lógica para que possamos desenvolver nosso programa. Portanto, para construir um programa precisamos pensar e identificar:

1. **Entrada:** Quais dados são necessários para começar o programa? que dados ele vai precisar executar?
2. **Processamento:** Quais são os cálculos que precisam ser feitos e quais decisões precisam ser tomadas?
3. **Saída:** Quais dados devem ser exibidos para o usuário?

Entrada de dados

Em Python vamos ter como comando de entrada de dados a função **input ()**, ela é responsável por receber os dados informados através da entrada padrão, o teclado. O dado recebido por essa função pode ser armazenado em uma variável, da seguinte forma: `nome = input()`

Podemos também exibir uma mensagem para o usuário, explicando o que deve ser digitado. Essas mensagens devem estar dentro do parênteses e entre aspas da função **input ()**.

```
nome = input("Informe seu nome: ")
```

```
Informe seu nome: João
```

Figura : Exemplos de váriaveis

A função input sempre irá retornar por padrão dados do tipo String, então para podermos utilizarmos os dados que não queremos que seja String, utilizamos outras duas funções de conversão para converter os dados de entrada antes de armazená-los.

```
idade = int(input("Informe sua idade:"))
altura = float(input("Informe a sua altura"))

Informe sua idade:25
Informe a sua altura1.85
```

Figura : Exemplos de váriaveis

OBS: Caso o dado que se deseja converter contenha algum caractere não numérico, essas funções irão gerar erros.

```
idade = int(input("Informe sua idade:"))
altura = float(input("Informe a sua altura"))

Informe sua idade:25
Informe a sua altura1.85b
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-4-8b444ea212d4> in <module>()
      1 idade = int(input("Informe sua idade:"))
----> 2 altura = float(input("Informe a sua altura"))

ValueError: could not convert string to float: '1.85b'
```

Figura : Exemplos de váriaveis

Também temos a possibilidade de padronizar os textos recebidos pelos usuários definindo se todo o texto vai ser com letras maiúsculas ou minúsculas. Essa técnica é muito utilizada quando precisamos fazer uma comparação entre dois textos. Por exemplo, o texto “Olá meu nome é Jhon”, possui letras maiúsculas e minúsculas, caso realizamos uma comparação com “olá meu nome é jhon” o nosso computador iria identificar como textos diferentes, pois na programação letras maiúsculas e minúsculas são consideradas caracteres diferentes.

Então para resolvermos esse impasse utilizamos duas funções: **str.upper** e **str.lower**.

```
nome = str.upper(input("Qual seu nome?:"))
curso = str.lower(input("Informe seu curso: "))

Qual seu nome?:José
Informe seu curso: PROGRAMAÇÃO
```

Figura : Exemplos de váriaveis

Esse seria o resultado:

```
JOSÉ
programação
```

Saída de dados

Para a saída vamos encontrar a função **print ()**, onde usamos para exibir mensagens, variáveis na tela e etc.

```
print (nome)
print (nome, idade)
print ("Eu sou um pato")
```

Figura : Exemplos de váriaveis

É possível também exibir mensagens juntamente com as variáveis, basta separar cada elemento por uma vírgula.

```
nome = "José"
idade = 22

print ("Seu nome é ",nome)
print ("Sua idade é ",idade)

Seu nome é  José
Sua idade é  22
```

Figura : Exemplos de váriaveis

Podemos exibir operações matemáticas básicas como 8/3.

```
print(8/3)
2.6666666666666665
```

Figura : Exemplos de váriaveis

Por padrão os números reais são exibidos com várias casas decimais, mas podemos utilizar a expressão **"%.nf"%nome_da_variavel**, para limitar as casas decimais, onde n representa a quantidade de casas decimais desejadas.

```
numero = 8/3

print("%.4f"%numero)
print("%.2f"%numero)

2.6667
2.67
```

Figura : Exemplos de váriaveis

Para não ficar só na leitura, tente criar:

1. Programa que some dois números informados pelo usuário.
2. Programa que receba o ano atual e o ano de nascimento do usuário e exiba a sua idade.
3. **Adicionar mais listas de exercícios.**

Estrutura Condicional

Um comando condicional é uma estrutura que permite a escolha de um grupo de comandos ou instruções a serem executadas quando determinada condição composta por expressões e operadores relacionais é satisfeita, ou seja ela é verdadeira. Dentro dessa estrutura condicional encontramos dois tipos: simples e composta. Onde a simples é utilizada para verificar se uma condição é verdadeira antes de executar uma instrução e podemos representar visualmente através do seguinte fluxograma:

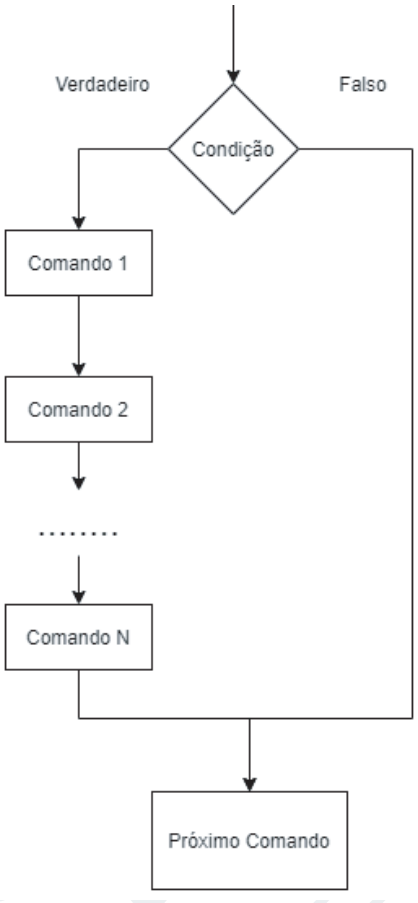

Figura : Exemplos de váriaveis

Para utilizarmos essa estrutura nos nossos códigos utilizamos a seguinte estrutura:

```
if (condição):
  comando 1
  comando 2
  comando 3
```

Figura : Exemplos de váriaveis

Ficou um pouco confuso ainda? Não consegue imaginar em qual situação utilizamos esse tipo de condição? Então veja esse exemplo para mostrar se um aluno foi aprovado ou reprovado por média:

```
nomeAluno = input("Informe o nome do aluno: ")
nota1 = float(input("Informe a primeira nota: "))
nota2 = float(input("Informe a segunda nota: "))
media = (nota1 + nota2)/2
print("A média de ",nomeAluno, "foi ", media)
if(media >= 7):
  print("Parabéns você foi aprovado! ^^")
```

```
Informe o nome do aluno: Maria
Informe a primeira nota: 10
Informe a segunda nota: 7
A média de  Maria foi  8.5
Parabéns você foi aprovado! ^^
```

Figura : Exemplos de váriaveis

Para o comando condicional composta utilizamos em situações, onde duas alternativas dependem da mesma condição, ou seja, temos uma ação para quando a condição for verdadeira e outra para quando ela for falsa. Podemos representar visualmente em um fluxograma na seguinte forma:

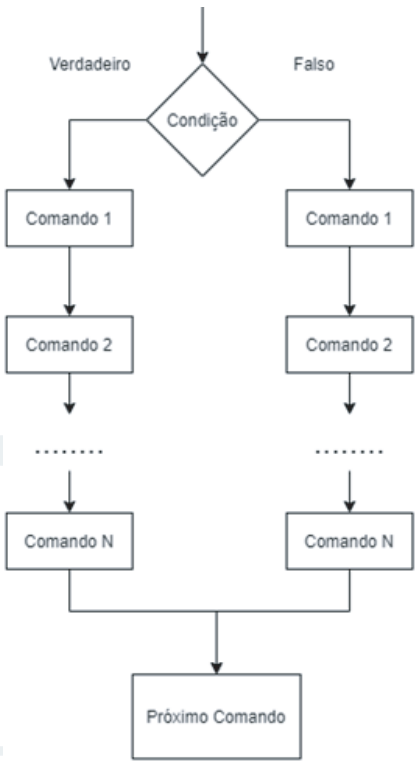

Figura : Exemplos de váriaveis

A estrutura em código ficaria da seguinte forma:

```
if (condição):
  comando 1
  comando 2
  comando 3

else:
  comando 1
  comando 2
  comando 3
```

Figura : Exemplos de váriaveis

Agora vamos implementar essa estrutura no nosso exemplo anterior:

```
nomeAluno = input("Informe o nome do aluno: ")
nota1 = float(input("Informe a primeira nota: "))
nota2 = float(input("Informe a segunda nota: "))
media = (nota1 + nota2)/2
print("A média de ",nomeAluno, "foi ", media)
if(media >= 7):
  print("Parabéns você foi aprovado! ^^")
else:
  print("Que pena! Mas você não foi aprovado. :(")
```

```
Informe o nome do aluno: Julia
Informe a primeira nota: 6
Informe a segunda nota: 6
A média de  Julia foi  6.0
Que pena! Mas você não foi aprovado. :(
```

Figura : Exemplos de váriaveis

Dentro do condicional composto ainda encontramos outro comando chamado elif, que utilizamos caso exista vários valores para uma mesma variável. Ficando da seguinte forma:

```
if (condição):
  comando 1
  comando 2
  comando 3

elif (condição):
  comando 1
  comando 2
  comando 3
else:
  comando 1
  comando 2
  comando 3
```

Figura : Exemplos de váriaveis

Implementando no exemplo das médias, poderíamos deixar da seguinte forma:

```
nomeAluno = input("Informe o nome do aluno: ")
nota1 = float(input("Informe a primeira nota: "))
nota2 = float(input("Informe a segunda nota: "))
media = (nota1 + nota2)/2
print("A média de ",nomeAluno, "foi ", media)
if(media >= 7):
  print("Parabéns você foi aprovado! ^^")
elif(media >=4 and media < 7):
  print("Você ficou em recuperação! 'o'")
else:
  print("Que pena! Mas você não foi aprovado. :(")
```

```
Informe o nome do aluno: Jessica
Informe a primeira nota: 7
Informe a segunda nota: 2
A média de  Jessica foi  4.5
Você ficou em recuperação! 'o'
```

Figura : Exemplos de váriaveis

Simples não? Um detalhe importante é que esses comandos **elif** e **else** são opcionais, podemos ter ou não eles em nosso código, ou melhor ainda podemos fazer algumas combinações, como:

1. Apenas if
2. if + else
3. if + um (ou vários) elif
4. if + um (ou vários) elif + else

Isso vai depender muito do programa na qual você está construindo. Devemos lembrar também que em qualquer combinação, apenas **UMA SEQUÊNCIA** será executada e que os comandos correspondentes dessas sequências devem estar obrigatoriamente indentados, ou seja, tabulados. Essa tabulação corresponde ao espaço em branco que fica logo após os comandos **if**, **elfi** e **else**. Caso isso fique um pouco confuso olhe a imagem abaixo para esclarecer o que seria essa tabulação:

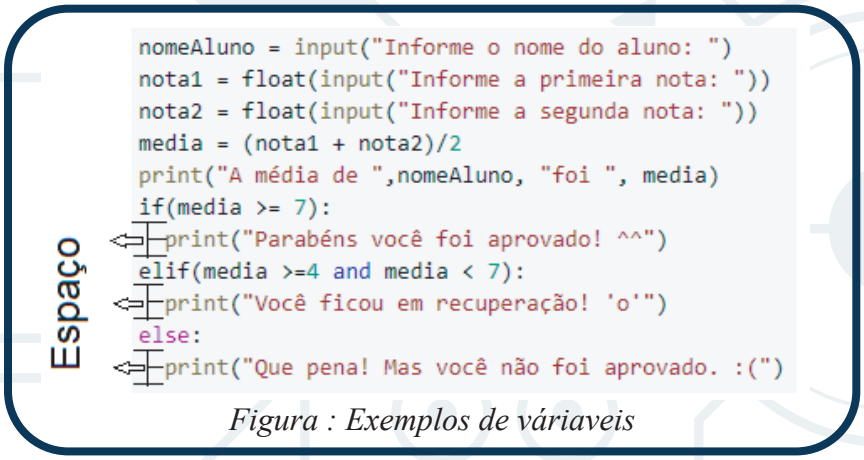

Figura : Exemplos de váriaveis

OBS: Sempre é bom colocar as condições em parênteses para facilitar nossa identificação e leitura do código.

Operadores para comparação

Na linguagem Python usamos os seguintes operadores lógicos para realizar as comparações:

Operador	Descrição	Exemplo
==	Igual	X == 4
!=	Diferente	Y != X
>	Maior	5 > 3
>=	Maior ou igual	X >= 8
<	Menor	2 < 6
<=	Menor ou igual	Y <= 7

Já para operações com mais de uma condição utilizamo and, or ou not:

Operador	Descrição	Exemplo
and	A expressão só será verdadeira se todas as condições forem verdadeiras	(8 > 4) and (3 < 5)
or	A expressão será verdadeira se pelo menos uma condição for verdadeira	(1 <= 2) or (6 > 0)
not	Nega o valor da expressão	not (4 > 3)

Comando de repetição (while)

While… While é uma estrutura de repetição que assegura um comportamento no programa enquanto uma condição é atendida(aliás o significado de while é exatamente isso, “enquanto”). De forma abstrata quer dizer que algo vai se repetir enquanto uma condição estiver sendo satisfeita ou até que ela seja satisfeita. Que tal colocarmos a mão na massa para entendermos melhor? Abra o seu IDLE python e vamos programar.

```
'''Agora nós vamos aprender como utilizar a estrutura de repetição While!
Para isso iremos escrever um programa que somará 2 à uma variavel
que armazena o valor 0, até que seu valor chegue a 100'''

#primeiro declaramos a variavel e atribuimos à ela o valor inicial:

numeroInicial = 0

#agora construimos a estrutura de repetição usando o while

''' As linhas de codigo abaixo dizem o seguinte:
enquanto a variavel "numeroInicial" for menor
que 100, então some 2 à variavel numeroInicial'''

while(numeroInicial < 100):
    numeroInicial+= 2
    print('Valor = ', numeroInicial)
```

Figura : Exemplos de váriaveis

Código sem os comentários:

```
numeroInicial = 0

while(numeroInicial < 100):
    numeroInicial+= 2
    print('Valor = ', numeroInicial)
```

Figura : Exemplos de váriaveis

Comando de repetição (while)

O comando de repetição for, difere do while, cuja sua quantidade de repetições é baseada em uma condição lógica(verdadeiro ou falso) e nem sempre pode ser prevista, já o For sempre trabalhará com uma quantidade de repetições fixa. O for é utilizado quando desejamos executar várias vezes um comando ou bloco de comandos, onde a quantidade de repetições é controlada por uma variável que assume um valor pré-estabelecido. Costumamos chamar essa variável de contador.

Para cada repetição, a variável que controla o for assume valores em um intervalo gerado pela função range(). Essa função tem a seguinte estrutura: `range(inicial, limite, incremento)` , onde:

- **Inicial:** É o primeiro valor do intervalo. Quando não informado, assume zero por padrão.
- **Limite:** É o valor que finaliza o intervalo, mas que não pode ser assumido pela variável.
- **Incremento:** É o valor a ser somado à variável a cada nova repetição. Pode ser positivo ou negativo, e, quando não informado, assume +1 por padrão.

OBS: É importante que essa variável não tenha seu valor modificado dentro da repetição para manter a contagem.

Para definirmos um comando de repetição for utilizamos a seguinte estrutura:

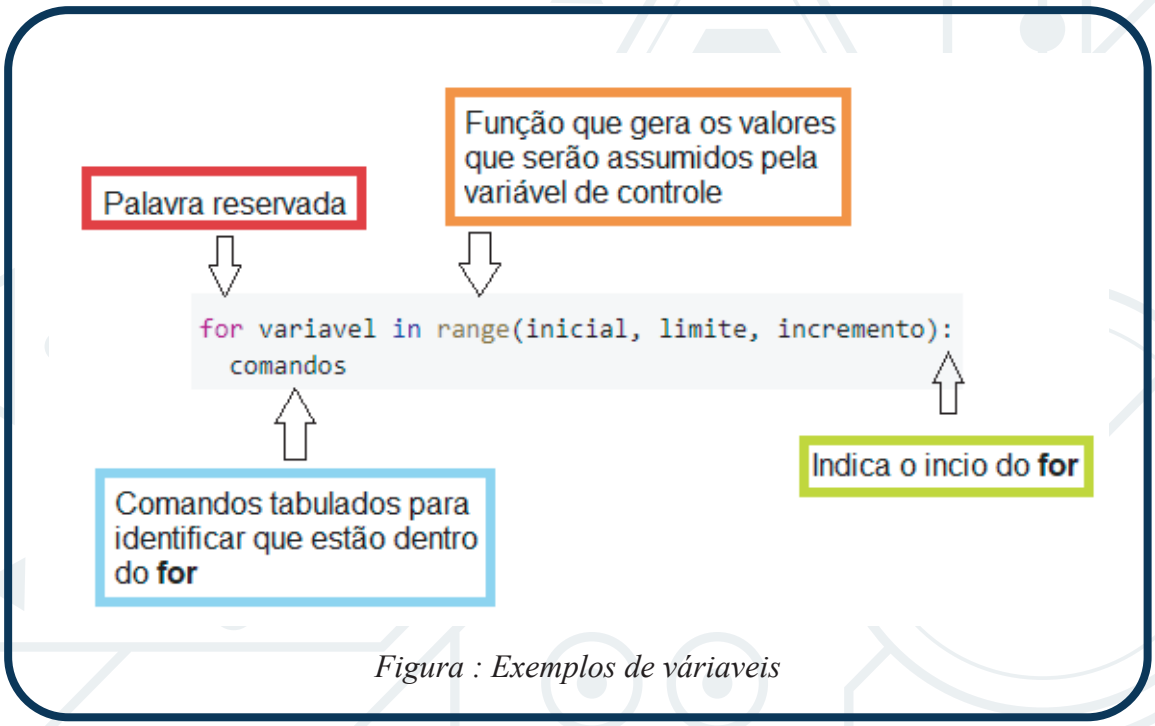

Figura : Exemplos de váriaveis

Agora vamos colocar um pouco a mão na massa, vamos criar um programa que vai exibir a tabuada de 9:

```
contador = 0
resultado = 0

for contador in range(0,11):
  resultado = 9 * contador
  print("9 x ", contador, " = ", resultado)
```

```
9 x  0  =  0
9 x  1  =  9
9 x  2  =  18
9 x  3  =  27
9 x  4  =  36
9 x  5  =  45
9 x  6  =  54
9 x  7  =  63
9 x  8  =  72
9 x  9  =  81
9 x  10  =  90
```

Figura : Exemplos de váriaveis

OBS: Perceba que por não ter colocado algum valor no campo de incremento, a função incrementou o valor 1 por padrão.

ATIVIDADE:

Crie um programa que faça toda a tabuada de 1 a 10.