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 que lhe auxiliar.

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

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

A palavra algoritmo pode ser um pouco estranha para quem está iniciando no mundo da programção, mas fazermos uso de algoritmos a todo momento da nossa vida, seja para escovar os dentes, cozinhar, dirigir e etc. Como vimos no tópico anterior um algoritmo é literalmente uma “receita” a ser seguida para executar uma determinada tarefa ou resolver um problema.

Quando queremos resolver um problema é necessário que seja encontrada uma maneira de descreve-la de uma forma simples, clara e precisa. Em seguida vai ser preciso encontrar uma sequencia de passos que vão permitir cehgar na resolução do problema. Essa sequencia de passos é o que chamamos de algoritmos.

Esses algortimos podem ser representados de diferentes formas, na qual podemos citar o português narrativo ou até mesmo através de fluxogramas. No português narrativo podemos simplismente escrever em texto corrido ou em forma de instruções que encontramos facilmente em manuais de instruções de jogos, eletrodomésticos e etc. Já o fluxograma é uma representação visual utilizando formas geometricas para representar um algoritmo, essa técnica é muito utilizada na administração e claro na programação!

Não vamos entrar nesse mundo conceitual de algortmos, com o decorrer do livro iremos abordar de forma pratica certos conceitos presentes nesse vasto universo.

A seguir temos alguns exemplos de algoritmos:

Manual de troca de pneu

Para trocar:

1. Afrouxe ligeiramente os parafusos dos pneus.
2. Suspender o carro usando um macaco.
3. Retirar os parafusos.
4. Retirar o pneu
5. Colocar o pneu reserva.
6. Recolocar os parafusos no lugar
7. Abaixar o carro
8. Dar o aperto final nos parafusos.

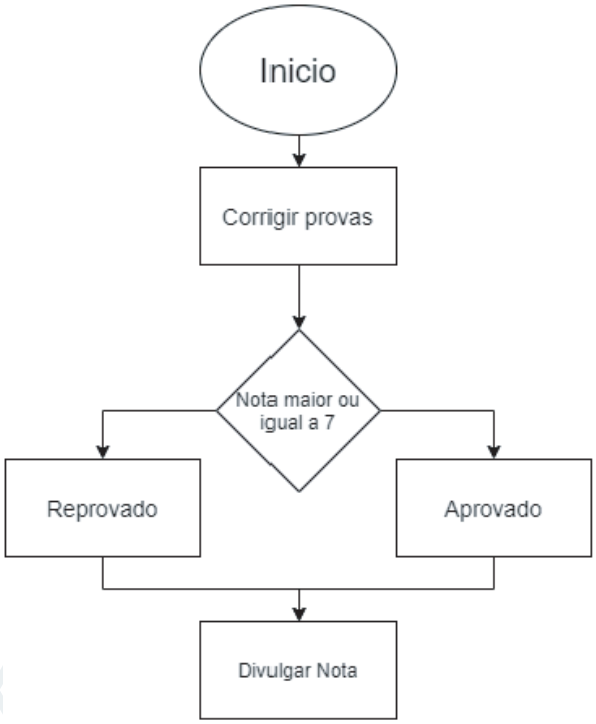

Figura 1: Algoritmo em forma de fluxograma.

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 atravessar a rua.

LINGUAGEM DE PROGRAMAÇÃO

Uma linguagem de programação é um paradigma de programação que faz uso da lógica matemática para resolver determinados problemas. Mas o que seria a lógica? e a lingaugem de programação propriamente dita? e um programa?

Todos esses conceitos acabam passando despercebidos por muitos, durante a aprendizagem de programação ou até mesmo quando já sabemos sobre a programação. Podemos entener a lógica como a ciência que estuda o raciocinio; a lógica de programação como o encadeamento lógico de instruções para o desenvolvimento de programas; e por fim programas como a implementação das instruções de um algoritmo em uma linguagem de programação. São conceitos que mesmo parecendo distintos, são interligados tanto no conceito como na prática.

Como os computadores não entendem o que falamos e por isso não podemos encrever programas utilizando a nossa linguagem natural, afinal ele só entende a linguagem binária, ou seja 0 e 1. Temos que utilizar para facilitar a comunicação entre o programador e o computador as famosas linguagens intermédiarias, pois são mais proximas da nossa linaguagem e ainda conseguem fazer a comunicação com o computador, já sabe como se chamam essas linguagens? Exato! As famosas linaguagens de programação.

Apesar do computador ser extremamnete poderoso e fazer coisas em questões de segundos, ele ainda é extremamente “burro”. Por que? Porque ele é apenas um conjunto de peças que consegue realizar determinadas tarefas e para isso ele precisa que alguém diga exatamente o que fazer, por isso, existe os programas de computadores e os programadores.

Assim como na nossa linguagemm temos um amontoado de regras gramaticais, as lingaugens programacionais também tem e são chamadas de sintaxe. A partir de uma linguagem de programação conseguimos ter acesso a certos comandos que ao utilizarmos seguindo um conjuntos de regras

conseguimos escrever um programa de computador.

Podemos encontrar grandes semelhanças das linguagens de programação com o nosso mundo, afinal temos em nosso planeta diversos países, onde cada país fala uma lingua especifica. Da mesma forma são as linguagens de programação, podemos encontrar várias atualmente, podendo citar como exemplos: Java, C, C#, Ruby e claro a linguagem do nosso curso! Python.

Figura 2: Linguagem de programação Python

Figura 3: Linguagem de programação Java

Figura 4: Linguagem de programação C

Figura 5: Linguagem de programação C#

Figura 6: Linguagem de programação Ruby

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:

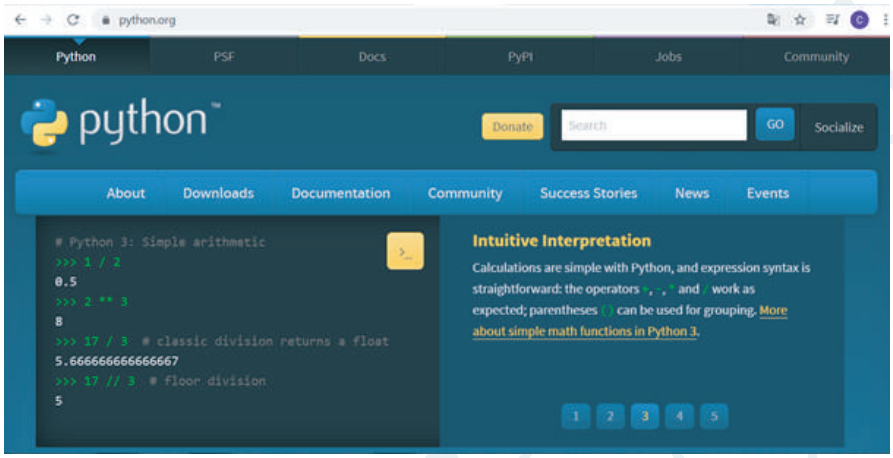

Figura : Página Oficial do Python

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

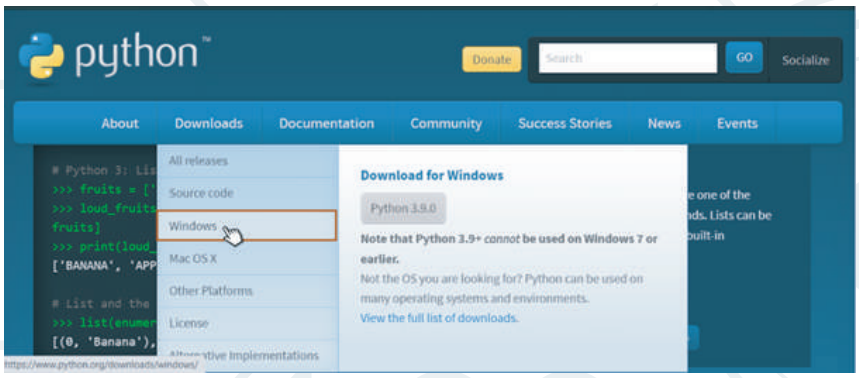

Figura : Página Oficial do Python

Agora selecione a primeira opção conforme a imagem.

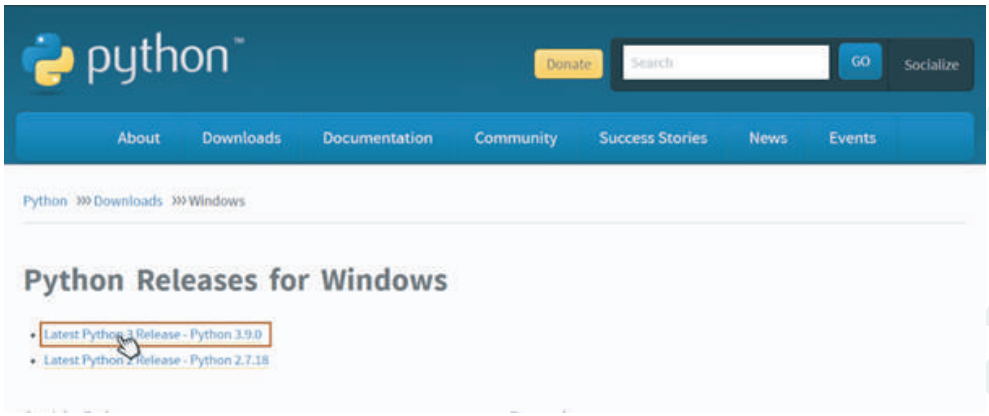

Figura : Página Oficial do Python

Essa tela se abrirá para você:

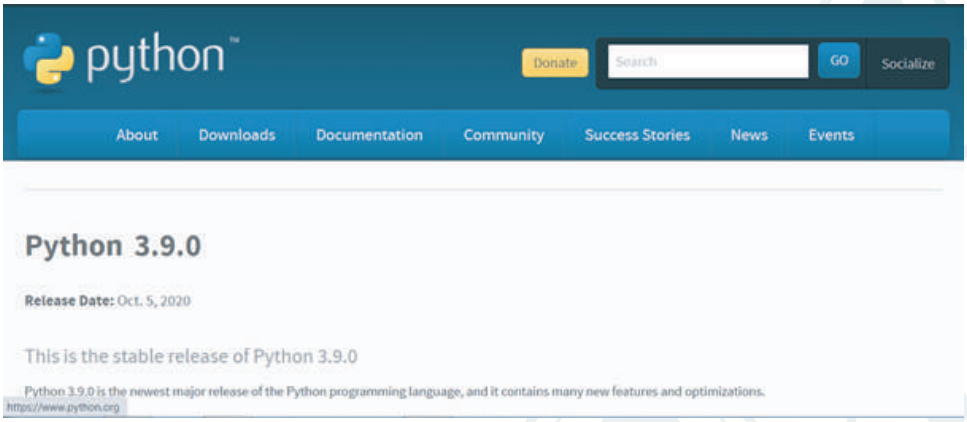

Figura : Página Oficial do Python

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.

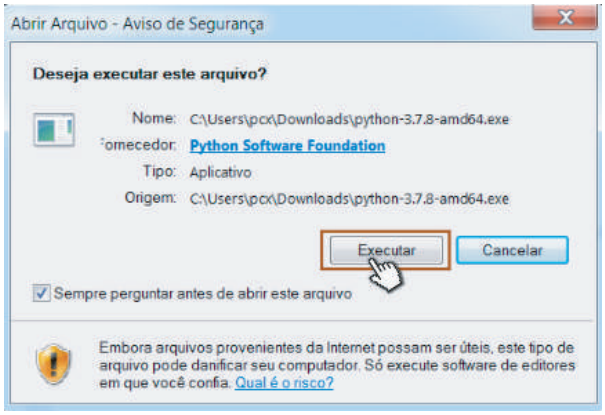

Figura : Página Oficial do Python

Na próxima janela click em “Install Now”.

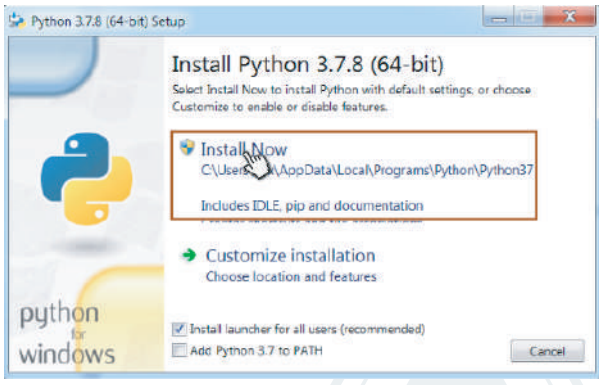

Figura : Página Oficial do Python

Aguarde a instalação ser concluída.

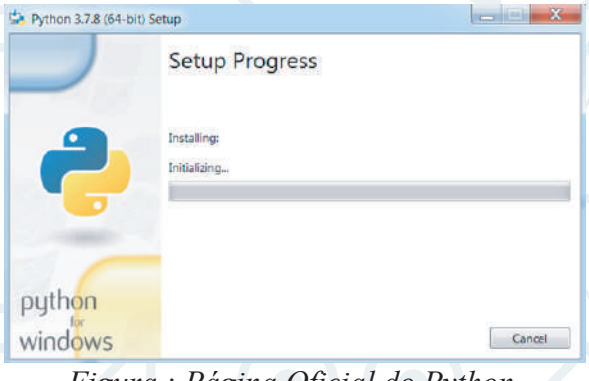

Figura : Página Oficial do Python

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

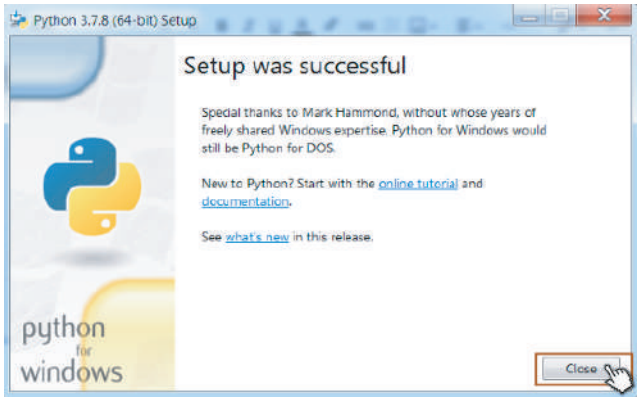

Figura : Página Oficial do Python

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

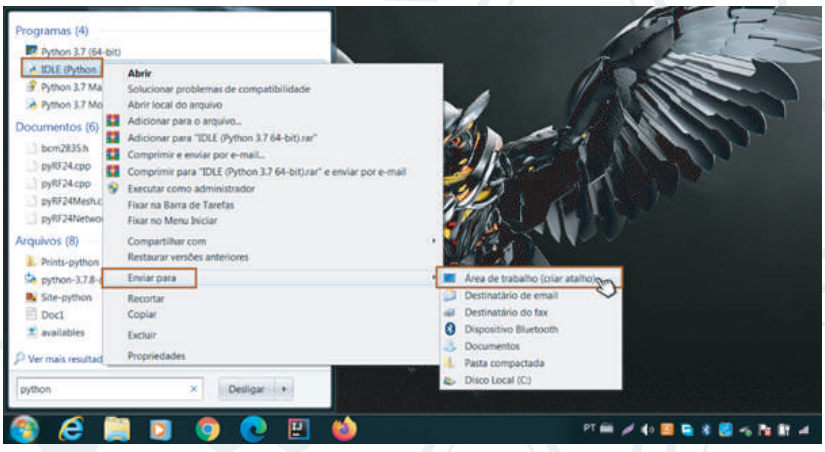

Figura : Página Oficial do Python

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)	Números do conjunto dos Inteiros	0,1,2,3….
float (Real)	Números dos conjuntos Reais	0; 1,2
char(Caracteres)	Unico Caractere	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:

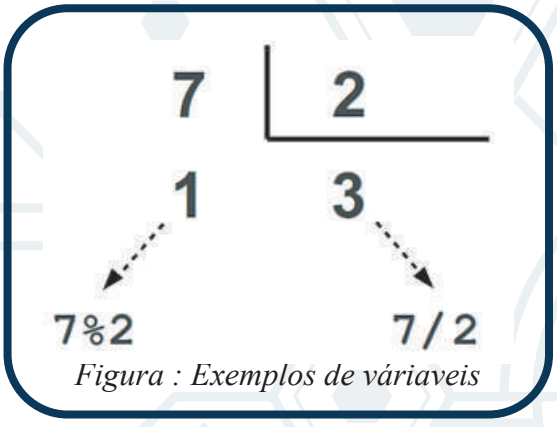

Figura : Exemplos de váriaveis

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.

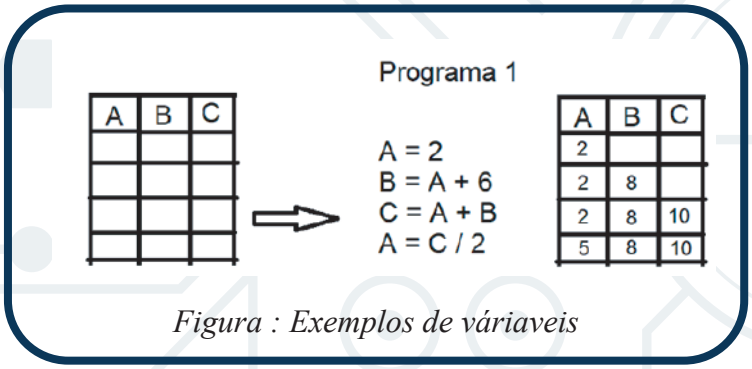

Figura : Exemplos de váriaveis

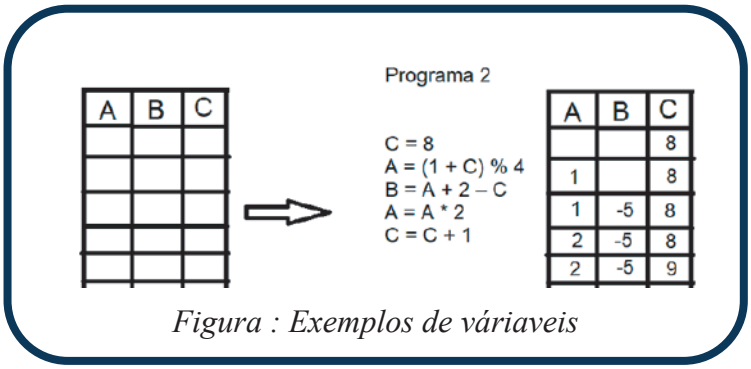

Figura : Exemplos de váriaveis

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.

Para conseguirmos definir um valor inteiro(int) ou Real(float) através do usúario, precisamos utilizar duas funções, onde uma será **int()** e a outra **float()**, que realizaram a conversão de String para o respectivo tipo desejado (inteiro ou real).

```
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:

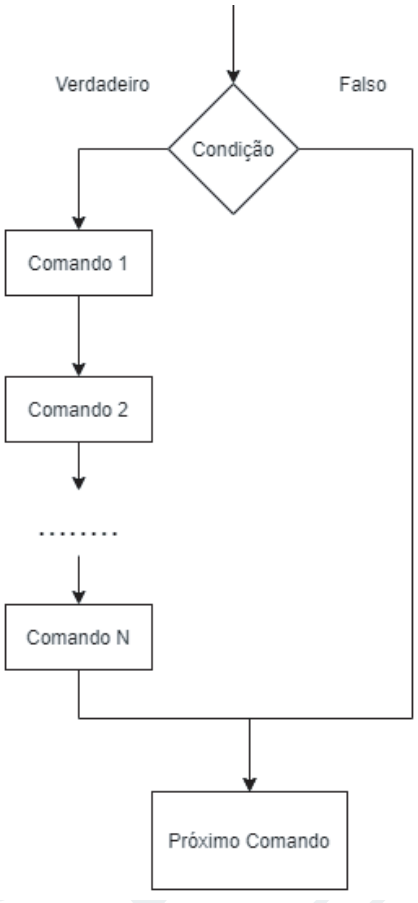

Figura : Exemplos de váriaveis

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:

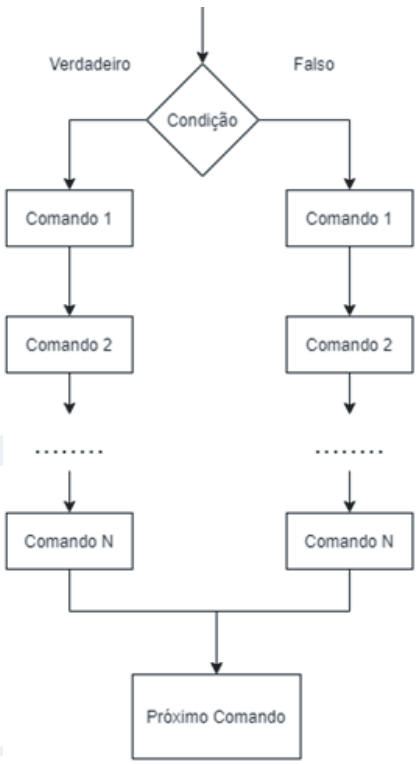

Figura : Exemplos de váriaveis

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:

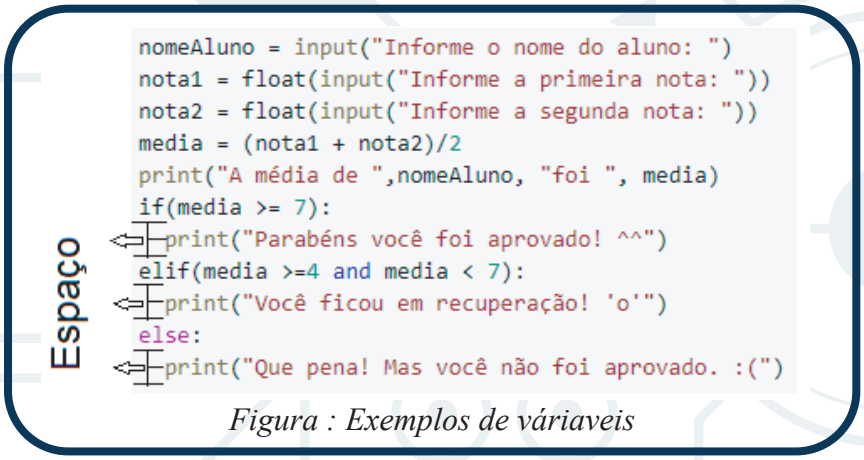

Figura : Exemplos de váriaveis

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 (FOR)

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:

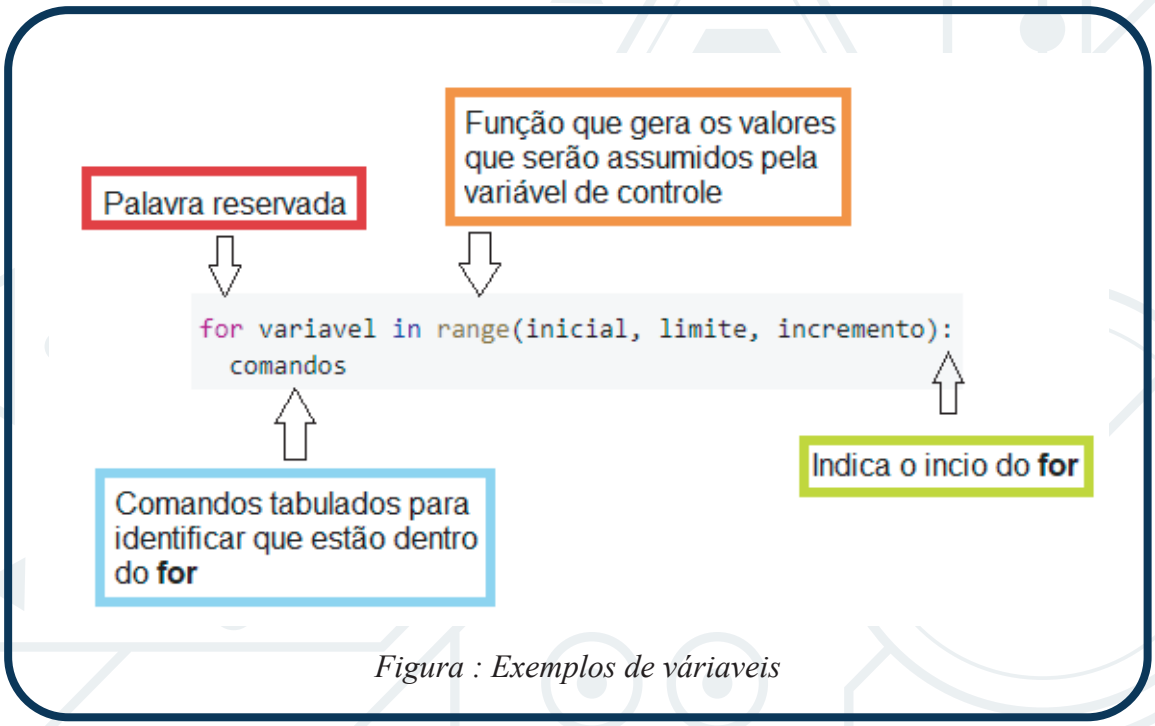

Figura : Exemplos de váriaveis

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.

LISTAS

Quando estamos desenvolvendo programas é muito comum a necessidade de organizar dados em listas. Uma lista nada mais é que uma estrutura capaz de armazenar informações de forma linear, nos permitindo acessar seus elementos por meio de índices, cada índice representa a posição de um item na lista, uma lista telefônica por exemplo ilustra bem o conceito geral.

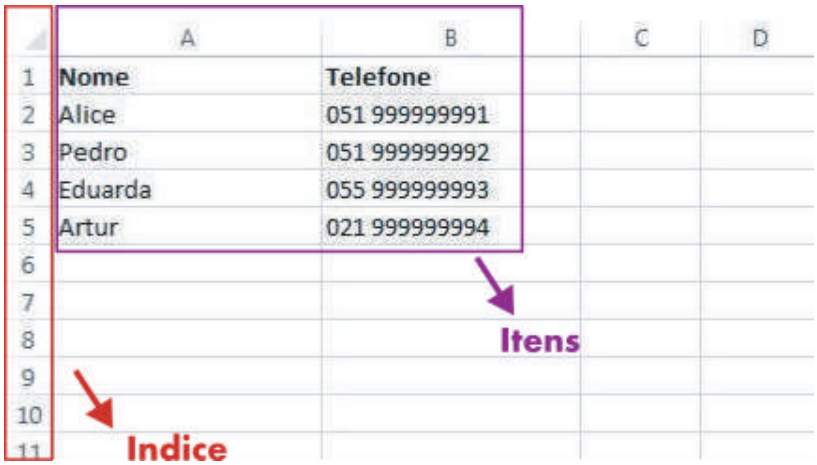

Figura : Exemplos de váriaveis

Diferentemente de listas telefônicas ou outros exemplos de listas, em Python os índices são numerados a partir do número “0”(zero), por isso é muito importante que não esqueçamos que a contagem sempre inicia em zero. Porque devemos estar atentos a isso? A resposta é simples, se não considerarmos a contagem a partir do zero, é certeza de que o nosso código não irá se comportar da maneira que esperamos, gerando resultados errados. Usar uma lista torna a busca por um item muito mais eficiente e organizada. Vamos ao código!?

Primeiro inicie o seu ambiente Python e crie um novo arquivo com a nomenclatura “manipulando listas”. Siga os passos abaixo.

```
""" Vamos iniciar criando a nossa lista, para isso, basta escrever o nome
da nossa lista como se estivessemos declarando uma variavél, seguido do sinal
'=' e de colchetes([]).
"""
# Assim:
listaNomes = []

""" Agora vamos declarar uma variavél
que receberá uma string fornecida pelo usuário"""

nome = str.upper(input("Digite um nome: "))

# Após inserir o nome que desejamos, devemos adicionar esse nome na nossa lista
# Utilizaremos o metodo 'append' para armazenar a imformação que queremos

listaNomes.append(nome)

"""Vamos fazer um print para verificar
se o nome foi armazenado corretamente na nossa lista"""

print(listaNomes)

# É só executar o programa e verificar o resultado!
```

Figura : Exemplos de váriaveis

O resultado deve ser algo assim:

```
Digite um nome: geoclecio
['GEOCLECIO']
```

Figura : Exemplos de váriaveis

Código sem comentários:

```
listaNomes = []
nome = str.upper(input("Digite um nome: "))
listaNomes.append(nome)
print(listaNomes)
```

Figura : Exemplos de váriaveis

No código acima inserimos uma informação na lista e em seguida imprimimos na tela, como só colocamos um dado e fizemos o print da lista inteira, o formato exibido traz mais informações do que necessário, isso ficará mais claro adiante.

Iremos escrever um programa onde possamos inserir mais de uma informação, para fazer isso vamos utilizar elementos da linguagem Python que já usamos anteriormente.

```
# Primeiro declaranmos nossa lista
listaNomes = []

# Definimos uma variavel para controlar a continuação do programa
continuar = "S"

""" Criamos uma estrutura de repetição 'while'
para deixar a execução em loop
até que a variavel "continuar" mude de "S" para "N". """

while (continuar == "S"):
  # Declaramos uma variavel "nome" para fazer inserção de informações
  nome = str.upper(input("Digite um nome: "))
  # Inserimos a informação na nossa lista
  listaNomes.append(nome)
  # Aqui repetimos a variavel "cotinuar" para que o usuario decida quando sair do programa
  continuar = str.upper(input("Deseja continuar? Digite 'S' para sim ou 'N' para não."))
# Exibimos toda nossa lista na tela
print(listaNomes)
```

Figura : Exemplos de váriaveis

Resultado do programa:

```
['GEOCLECIO', 'WILLAMS', 'DARLLAN', 'RAFA']
```

Figura : Exemplos de váriaveis

Mas e se a gente não quiser toda a lista, se nós quisermos imprimir na tela apenas 1 elemento? Como é que faz? Presta bastante atenção na imagem a seguir, nós vamos usar o mesmo código, porém iremos modificar uma pequena parte para exibir apenas o nome “Darllan”.

```
# Primeiro declaranmos nossa lista
listaNomes = []

# Definimos uma variavel para controlar a continuação do programa
continuar = "S"

""" Criamos uma estrutura de repetição 'while'
para deixar a execução em loop
até que a variavel "continuar" mude de "S" para "N". """

while (continuar == "S"):
  # Declaramos uma variavel "nome" para fazer inserção de informações
  nome = str.upper(input("Digite um nome: "))
  # Inserimos a informação na nossa lista
  listaNomes.append(nome)
  # Aqui repetimos a variavel "cotinuar" para que o usuario decida quando sair do programa
  continuar = str.upper(input("Deseja continuar? Digite 'S' para sim ou 'N' para não."))
# Exibindo apenas o nome "Darllan"
print(listaNomes[2])
```

Figura : Exemplos de váriaveis

Resultado do programa:

```
DARLLAN
```

Figura : Exemplos de váriaveis

Como podemos observar, usamos o mesmo programa, no entanto, para controlar o que iríamos mostrar na tela, informamos ao fazermos o print, dentro do colchetes, qual era o índice que queríamos exibir.

Observem:

```
print(listaNomes[2])
```

Com a adição de apenas três caracteres no nosso programa, fizemos um print em “listaNomes” índice 2.

Mas o nome “Darllan” não estava na posição 3?

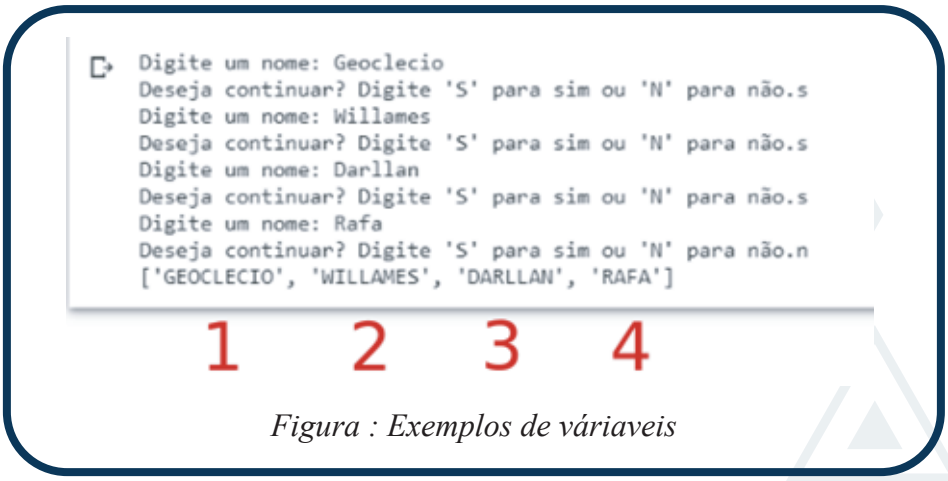

Figura : Exemplos de váriaveis

Se utilizarmos a contagem comum com a qual estamos habituados iremos obter resultados imprecisos, lembremos que lá atrás havíamos falado que em Python os índices das listas iniciam em “0”(zero), por isso, dentro da lista o nome ”Darllan” se encontra na posição 2.

Vejamos uma imagem com a contagem correta.

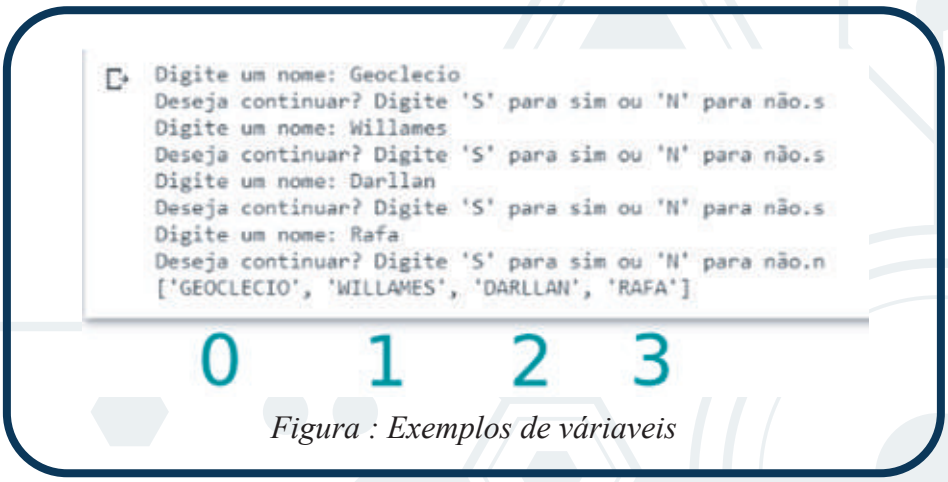

Figura : Exemplos de váriaveis

As listas podem ser manipuladas de várias formas utilizando a sintaxe: nome-lista[valor-índice]. Podemos por exemplo substituir um item da nossa lista por outro, vejamos como podemos fazer isso:

```python
# Primeiro criamos nossa lista com alguns itens dentro

listaNomes = ['Geoclecio', 'Willams', 'Darllan', 'Rafa']

# Faremos um print para exibir toda nossa lista

print(listaNomes)

""" Agora vamos substituir um item da lista por outro
utilizando a sintaxe nome-lista[valor-índice] """

listaNomes[2] = "Julia"

# Iremos fazer mais um print para ver o resultado da substituição

print(listaNomes)
```

Figura : Exemplos de váriaveis

Vamos observar como ficou a saída:

```
['Geoclecio', 'Willams', 'Darllan', 'Rafa']
['Geoclecio', 'Willams', 'Julia', 'Rafa']
```

Figura : Exemplos de váriaveis

Viu só? Substituímos um item da nossa lista de forma super fácil. Mas e se ao invés de substituir, a gente quiser inserir um dado a mais em algum índice específico, como fazemos isso? Simbora acompanhar o exemplo:

Para colocarmos uma informação em um índice específico iremos utilizar um método novo da linguagem Python que se chama “insert”, a sintaxe é bem simples: `nome-lista.insert(índice, “elemento”)`. Vamos ao código para que isso fique mais claro.

```python
# Primeiro criamos nossa lista com alguns itens dentro

listaNomes = ['Geoclecio', 'Willams', 'Darllan', 'Rafa']

# Faremos um print para exibir toda nossa lista

print(listaNomes)

""" Utilizando o metodo 'insert'.
Vamos adicionar o item 'Mariana' na posição 1 da nossa lista
que já possui o elemento 'Willams' armazenado """

listaNomes.insert(1, "Mariana")

# Iremos fazer mais um print para ver o resultado da substituição

print(listaNomes)
```

Figura : Exemplos de váriaveis

Resultado:

```
['Geoclecio', 'Willams', 'Darllan', 'Rafa']
['Geoclecio', 'Mariana', 'Willams', 'Darllan', 'Rafa']
```

Figura : Exemplos de váriaveis

Observemos que o item ‘Willams’ não foi apagado da lista, mas sim movido para uma nova posição, enquanto que o item ‘Mariana’ ocupou o espaço 1 da nossa lista.

E se o nosso desejo fosse simplesmente apagar um item sem substituí-lo, apenas eliminá-lo, como faríamos isso? Em Python existem três comandos básicos, são eles: del, pop e remove, o comando ‘del’, como você já deve ter imaginado, deleta um elemento específico através do índice, iremos ver a sintaxe diretamente no código. O método ‘pop’ geralmente é utilizado para remover o último elemento da lista, mas também podemos usar para retirar um elemento específico passando como parâmetro o índice que desejamos remover. Com o método ‘remove’, diferentemente dos outros dois, iremos informar como parâmetro o nome do item que queremos remover e o próprio Python se encarregará de encontrar o índice e fazer a remoção. Vamos ver como isso funciona? Continuaremos usando a mesma lista para esses exemplos.

Observe o resultado das listas utilizando o método ‘del’:

```
# Primeiro criamos nossa lista com alguns itens dentro

listaNomes = ['Geoclecio', 'Willams', 'Darllan', 'Rafa']
print("Resultado das listas! \n")

# Faremos um print para exibir toda nossa lista
print(listaNomes)

# Metodo 'del'
del listaNomes[0]

# Exibindo como ficou a lista após a utilização do metodo
print(listaNomes)
```

método ‘del’:

```
Resultado das listas!

['Geoclecio', 'Willams', 'Darllan', 'Rafa']
['Willams', 'Darllan', 'Rafa']
```

Figura : Exemplos de váriaveis

Método ‘pop’:

```
# Primeiro criamos nossa lista com alguns itens dentro

listaNomes = ['Geoclecio', 'Willams', 'Darllan', 'Rafa']
print("Resultado das listas! \n")

# Faremos um print para exibir toda nossa lista
print(listaNomes)

# Metodo 'pop'
listaNomes.pop(3)

# Exibindo como ficou a lista após a utilização do metodo
print(listaNomes)
```

```
Resultado das listas!

['Geoclecio', 'Willams', 'Darllan', 'Rafa']
['Geoclecio', 'Willams', 'Darllan']
```

Figura : Exemplos de váriaveis

Método ‘remove’:

```
# Primeiro criamos nossa lista com alguns itens dentro

listaNomes = ['Geoclecio', 'Willams', 'Darllan', 'Rafa']
print("Resultado das listas! \n")

# Faremos um print para exibir toda nossa lista
print(listaNomes)

# Metodo 'remove'
listaNomes.remove("Willams")

# Exibindo como ficou a lista após a utilização do metodo
print(listaNomes)
```

```
Resultado das listas!

['Geoclecio', 'Willams', 'Darllan', 'Rafa']
['Geoclecio', 'Darllan', 'Rafa']
```

Figura : Exemplos de váriaveis

Nós aprendemos alguns comandos para remover e adicionar elementos em uma lista com a linguagem Python. Até agora utilizamos uma única lista para nossos exemplos, talvez já esteja na hora de aumentar nossas listas e ver o que mais é possível fazer com elas. já que vamos usar mais listas, será que dá pra gente pegar várias listas e transformar em uma só? Vamos testar se isso é possível, mão no código!

Soma de listas:

```
""" Utilizando a linguagem Python é possivel
manipular listas com operadores matematicos como
soma e multilicação, vamos ver como podemos fazer isso """
print("Resultado: \n")
print("Lista 1 \n", listaNomes)
listaNomes = ['Geoclecio', 'Willams', 'Darllan', 'Rafa']

print("Lista 2 \n", listaCidades)

listaCidades = ['Trindade', 'Paulista', 'Maranhão', 'Carpina']

print("Soma das listas 1 e 2")
soma = listaNomes + listaCidades

print(soma)
```

```
Resultado:

Lista 1
 ['Geoclecio', 'Willams', 'Darllan', 'Rafa']
Lista 2
 ['Trindade', 'Paulista', 'Maranhão', 'Carpina']
Soma das listas 1 e 2
['Geoclecio', 'Willams', 'Darllan', 'Rafa', 'Trindade', 'Paulista', 'Maranhão', 'Carpina']
```

Figura : Exemplos de váriaveis

Notemos que as listas foram concatenadas conforme a sua ordem de escrita, se nós trocarmos a ordem dos fatores o resultado da soma irá mudar as posições dos elementos.

```
""" Utilizando a linguagem Python é possivel
manipular listas com operadores matematicos como
soma e multilicação, vamos ver como podemos fazer isso """
print("Resultado: \n")
print("Lista 1 \n", listaNomes)
listaNomes = ['Geoclecio', 'Willams', 'Darllan', 'Rafa']

print("Lista 2 \n", listaCidades)

listaCidades = ['Trindade', 'Paulista', 'Maranhão', 'Carpina']

print("Soma das listas 1 e 2")
soma = listaCidades + listaNomes

print(soma)
```

```
Resultado:

Lista 1
 ['Geoclecio', 'Willams', 'Darllan', 'Rafa']
Lista 2
 ['Trindade', 'Paulista', 'Maranhão', 'Carpina']
Soma das listas 1 e 2
['Trindade', 'Paulista', 'Maranhão', 'Carpina', 'Geoclecio', 'Willams', 'Darllan', 'Rafa']
```

Figura : Exemplos de váriaveis

Multiplicação:

```
""" Utilizando a linguagem Python é possível
manipular listas com operadores matemáticos como
soma e multilicação, vamos ver como podemos fazer isso """

print("Resultado: \n")

listaNomes = ['Geoclecio', 'Willams', 'Darllan', 'Rafa']

print("Mutiplicação da lista de nomes")
mutiplicacao = 3 * listaNomes

print(mutiplicacao)
```

```
Resultado:

Mutiplicação da lista de nomes
['Geoclecio', 'Willams', 'Darllan', 'Rafa', 'Geoclecio', 'Willams', 'Darllan', 'Rafa', 'Geoclecio', 'Willams', 'Darllan', 'Rafa']
```

Figura : Exemplos de váriaveis

Podemos perceber que no caso da multiplicação, não podemos utilizar uma lista para multiplicar outra, só podemos utilizar números. Isso se dá pelo fato de que uma lista armazena tipos variados de dados e como já sabemos, não é possível multiplicar por exemplo, letras por números. Listas é um assunto que poderíamos fazer um curso exclusivo só para explorar suas possibilidades, mas para o objetivo desse curso de introdução, o que aprendemos até aqui já é satisfatório. Antes de irmos ao próximo assunto, que tal fazermos um exercício para praticar?

SubListas

Nosso próximo assunto será ‘sublistas’, tenho certeza de que iremos aprender bastante com as possibilidades que as sublistas podem nos proporcionar, espero que assim como eu vocês estejam entusiasmados e abertos para aprender um pouco mais sobre Python!

O que é uma sublistas? De forma simples, uma sublista nada mais é do que uma lista dentro de outra lista. Vamos ilustrar para uma melhor compreensão!

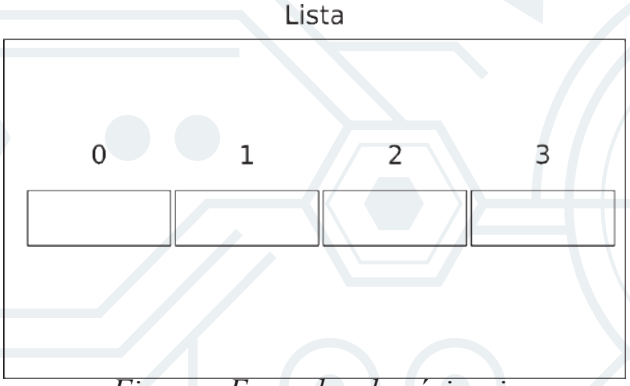

Figura : Exemplos de váriaveis

De forma abstrata uma lista é uma caixa onde colocamos coisas lá dentro, como já falamos lá no tópico “listas”, cada coisa recebe um lugar para ocupar, esses lugares nós chamamos de índices, estão representados na figura acima pelo retângulos.

Quando trabalhamos com sublistas, o que fazemos é adicionar uma caixinha dentro da caixa que tínhamos anteriormente, por sua vez essas caixinhas também irão abrigar coisas nelas e essas coisas também receberão seus próprios índices, está claro até aqui? Vamos ver outra imagem para melhorar o entendimento:

Figura : Exemplos de váriaveis

Como podemos observar as caixinhas que colocamos dentro da nossa caixa “Lista” agora irão armazenar os nossos dados. Isso serve para melhorar a organização e facilitar a busca dos objetos através dos índices.

Vejamos um exemplo de lista telefônica:

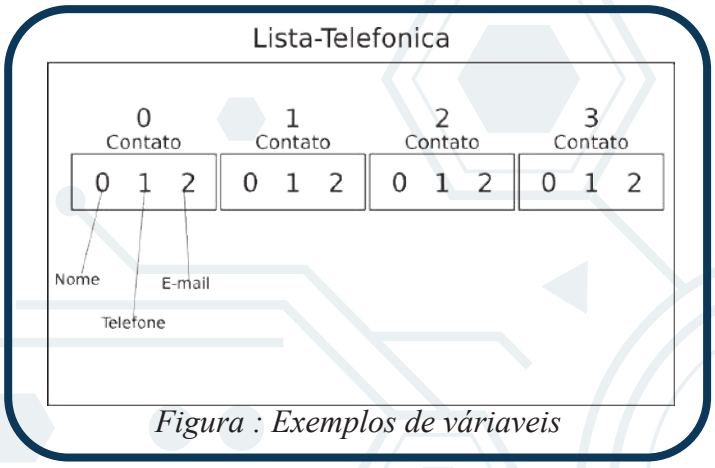

Figura : Exemplos de váriaveis

Creio que agora conseguimos perceber com maior clareza como as informações ficam melhor organizadas quando utilizamos sublistas. Mas para que haja maior aproveitamento vamos construir um código que simule uma lista telefônica. Mãos ao código!

```
# Primeiro vamos criar uma lista

listaTelefonica = []

# Definimos uma variavel de controle para manipular a estrutura de repetição

continuar = "S"

while (continuar == "S"):
  # Agora iremos declarar três variaveis com um input
  name = str (input("Escreva um nome: "))
  telefone = str (input("Digite o telefone: "))
  email = str (input("Informe seu E-mail: "))
  # É hora de criar sublistas dentro da lista alunos
  listaTelefonica.append([name, telefone, email])

  continuar = str.upper(input("Deseja continuar? Digite 'S' para sim ou 'N' para não " ))

print(listaTelefonica)
```

Figura : Exemplos de váriaveis

O código acima possui uma estrutura que nos permite criar uma lista de contatos. O que podemos fazer para acessar um contato dessa lista e obter informações de que iremos precisar.

Vamos escrever um trecho de código que busca em toda a lista e exibe os nomes de todos os nossos contatos, você pode se basear pelo exemplo para criar outros programas.

```
for i in range(len(listaTelefonica)):
  print(listaTelefonica[i][0])
```

```
Geoclecio
Willams
```

Figura : Exemplos de váriaveis

Repare que estamos utilizando o ‘for’ para percorrer a lista, também estamos fazendo uso do comando ‘len’, esse comando serve para descobrirmos qual é o tamanho da nossa lista.

Essa foi uma longa jornada e estamos bem próximos do fim, espero que tenha gostado de estar conosco até aqui, tudo o que precisamos fazer agora é resolver os exercícios e revisar o material sempre que alguma dúvida surgir. Lembre-se que para se tornar um bom programador é preciso praticar bastante, por isso sempre que puderem… Mãos ao código.

Referências

Siebra, Sandra de Albuquerque. **Introdução a Programação**. Volume 1. Recife: UFRPE, 2010.

Banin, Sérgio Luiz. **Python 3: uma abordagem didática**. São Paulo: Érica, 2018.