Essa é uma versão dos Guias de Estudo desenvolvida para auxiliar o processo de impressão e para facilitar a leitura dos guias por programas que fornecem a leitura automatizada para suporte a todos os alunos que necessitem. Dessa forma, não apresentaremos ilustrações nesse arquivo.* *#ParaTodosLerem****.*

Módulo #1

Guias de estudo

Boas Vindas!

Olá, Mentoramer!
Você iniciou a sua jornada no curso de ***Python****! Tente aproveitar ao máximo o seu curso!*

Para ajudar os nossos alunos a consolidar o conhecimento adquirido, disponibilizamos esses ***guias de estudo*** *para cada um dos módulos do curso. Nesses guias você encontrará um pequeno resumo do que aprendeu durante as aulas e dicas extras para evoluir ainda mais!*

Lembre-se também que você pode ***tirar mais dúvidas e iniciar discussões produtivas com os nossos mentores lá na nossa comunidade no Discord. Participe!***

Aproveite a sua jornada de aprendizagem ao máximo e conte com o ***apoio completo do time Mentorama durante toda a sua trajetória! Bora estudar?***

Módulo #1

Nesse módulo, vocês aprenderam sobre ***a linguagem de programação Python****, entenderam o que são* ***variáveis****, conheceram os* ***tipos de operadores*** *que temos em programação e aprenderam também sobre estruturas de repetição. Agora, nada melhor do que uma boa revisão para fixarmos melhor todo este conteúdo. Pegue seu caderno de anotações e vamos começar!*

Imagine que estamos desenvolvendo um programa que pede para o usuário informar sua idade. Nesse programa, queremos que ele retorne se esta pessoa é maior de idade ou não. Para identificarmos essa informação é preciso acessar a idade que a pessoa informou. Para isso, precisamos então armazenar este dado na memória do nosso computador.

*Uma forma de armazenar dados na memória para usarmos no decorrer do programa é por meio do **uso de variáveis.** Nas variáveis podemos armazenar dados que podem variar, ou seja, posso armazenar a idade da Maria e, depois, sobrescrever na variável a idade do João. Lembrando que com esta estrutura de dados podemos armazenar apenas um valor por vez.*

Vale lembrar que as variáveis podem ser de diferentes tipo:

- *As **strings** são usadas para nomes, frases e etc..*
  - *Exemplo:*
    ```
    nome="Maria Silva Santos"
    ```

- *Os **inteiros** são usados para valores inteiros como idade, ano, quantidade de algo e etc...*
  - *Exemplo:*
    ```
    anoNascimento=1993
    ```

- *Os pontos **flutuantes** (**float**) são usados para valores que possuem casas decimais como peso, altura, preço e etc...*
  - *Exemplo:*
    ```
    salario=5000.78
    ```

- *Os valores **booleanos** (**bool**) são usados quando as variáveis possuem valor True ou False*
  - *Exemplo:*
    ```
    sexoFeminino=False
    sexoMaculino=True
    ```

Vamos voltar ao problema de criar um programa que informe se uma pessoa é maior de idade ou não, quer ver na prática como isso pode ser feito?

Primeiro, vamos criar um código que solicita a idade do usuário:

```
int(input('Por favor, informe sua idade: '))
```

*A função `input` cria uma caixinha de texto para o usuário informar nela sua idade. Tudo que é escrito dentro da caixinha, por mais que seja um valor inteiro, sempre vem no formato de **string**.*

*Por isso, usamos a função `int()` para converter de **string** para **inteiro**, pois é o tipo de dado correto para idade.*

Agora, queremos criar um código que vai verificar se o usuário é maior de idade. Observe que, com o programa acima, não é possível fazer a verificação porque não armazenamos na memória o valor informado pelo usuário. Logo, não podemos manipular este dado. Teremos que criar uma variável que vai receber a idade informada:

```
idade=int(input('Por favor, informe sua idade: '))
```

Como podemos identificar se o usuário é maior de idade? Para isso, verificamos se a idade que ele informou é maior ou igual a 18.
Se esta condição for verdadeira o usuário é maior de idade, caso contrário, o usuário é menor de idade.

```
if idade>=18:
 print('Usuário maior de idade')
else:
 print('Usuário menor de idade')
```

Observe que para o programa identificar se o usuário é maior de idade ele precisa verificar se a idade da pessoa é maior ou igual a 18, por isso foi criada a sentença "`if idade>=18:`".
Esta sentença retorna o valor True, caso positivo e False, caso negativo.
Operações como esta são muito utilizadas na programação. Vamos recapitular todas elas?

Operadores relacionais

***Operadores relacionais** são utilizados para comparar valores. Seus resultados serão sempre **TRUE** ou **FALSE**.*

==	**igual**	Comparar se duas sentenças ou valores são iguais
>	**maior**	Comparar se uma sentença ou valor é maior que outra(o)
<	**menor**	Comparar se uma sentença ou valor é menor que outra(o)
>=	**maior ou igual**	Comparar se uma sentença ou valor é maior ou igual a outra(o)
<=	**menor ou igual**	Comparar se uma sentença ou valor é menor ou igual a outra(o)
!=	**diferente**	Comparar se uma sentença ou valor é diferente de outra(o)

*Vamos agora ver uma possível aplicação de alguns **operadores relacionais**:*

Suponha um programa para lutas de boxe. As entradas do programa serão: nome, idade, nome da cidade em que reside, altura e peso dos pares de atletas que disputarão. O programa deve comparar se:

- *o primeiro nome deles é igual*
- *se os atletas moram em cidades diferentes*
- *se o peso do primeiro atleta informado é maior que do segundo*
- *se o primeiro atleta informado é mais baixo que o segundo*

Vamos codificar?
Primeiro criamos a etapa de solicitar os dados dos dois atletas:

```
#dados do primeiro atleta
NomeAtleta1=input('Informe o Primeiro Nome do Primeiro Atleta')
IdadeAtleta1=int(input('Informe a Idade do Primeiro Atleta'))
CidadeAtleta1=input('Informe a Cidade Onde Reside o Primeiro
Atleta')
AlturaAtleta1=float(input('Informe a Altura do Primeiro Atleta'))
PesoAtleta1=float(input('Informe o Peso do Primeiro Atleta'))

#dados do segundo atleta
NomeAtleta2=input('Informe o Primeiro Nome do Segundo Atleta')
IdadeAtleta2=int(input('Informe a Idade do Segundo Atleta'))
CidadeAtleta2=input('Informe a Cidade Onde Reside o Segundo Atleta')
AlturaAtleta2=float(input('Informe a Altura do Segundo Atleta'))
PesoAtleta2=float(input('Informe o Peso do Segundo Atleta'))
```

Agora verificamos as relações entre eles:

O primeiro nome deles é igual?
*O programa retornará **True** (caso positivo) ou **False** (caso negativo):* `NomeAtleta1 == NomeAtleta2`

Os atletas moram em cidades diferentes?
`CidadeAtleta1 != CidadeAtleta2`

O peso do primeiro atleta informado é maior que do segundo?
`PesoAtleta1 > PesoAtleta2`

O primeiro atleta informado é mais baixo que o segundo?
`AlturaAtleta1 < AlturaAtleta2`

Operadores relacionais

*Além dos operadores relacionais temos também os **operadores lógicos**, sendo os principais operadores lógicos usados na programação o **AND**, **OR** e **NOT**.*

*O **AND** retorna verdadeiro quando todas as sentenças são verdadeiras. Por exemplo:*

*Suponha que para ser efetivado em uma empresa o candidato deve ser maior de idade **e** deve também morar na cidade de São Paulo, então queremos criar um programa que vai identificar se o candidato atende os dois requisitos.*

*Observe que queremos fazer duas verificações, porém a resposta do programa deverá ser apenas uma: **True** (caso o candidato tenha idade maior ou igual a 18 anos e resida na cidade de São Paulo) ou **False** (caso as duas sentenças ou uma delas seja falsa).*

Primeiro criamos a etapa de solicitar os dados:

```
idade=int(input('Informe a idade do candidato: '))
cidade=input('Informe a cidade que o candidato reside: ')
```

*Agora vamos ver se o candidato atende os pré requisitos, se sim, a resposta do código a seguir será **True***

```
idade>=18 and cidade=='SP'
```

Sentença 1 `(idade>=18)`	***And***	*Sentença 2* `(cidade=='SP')`	*Resposta do programa*
TRUE		*TRUE*	*TRUE*
TRUE		*FALSE*	*FALSE*
FALSE		*TRUE*	*FALSE*
FALSE		*FALSE*	*FALSE*

O operador **OR** retorna verdadeiro quando todas ou uma das sentenças são verdadeiras. Diferente do **AND**, para o **OR** basta uma sentença ser verdadeira.

Por exemplo: para ser efetivado em uma empresa o candidato deve possuir ensino superior completo **ou** mais de 10 anos de experiência. Então, caso o candidato atenda um dos dois requisitos, ou os dois, ele está apto a trabalhar na empresa.

Primeiro criamos a etapa de solicitar os dados:

```
ensinoSuperior=input('Possui Ensino Superior Completo? Digite N-Não e S-Sim ')
anosExperiencia=int(input('Anos de experiencia: '))
```

*Agora vamos ver se o candidato atende os pré requisitos, se sim, a resposta do código a seguir será **True**:*

```
ensinoSuperior=='S' or anosExperiencia>10
```

Sentença 1 (`ensinoSuperior=='S'`)		*Sentença 2 (`anosExperiencia>10`)*	*Resposta do programa*
TRUE		*TRUE*	*TRUE*
TRUE	***OR***	*FALSE*	*TRUE*
FALSE		*TRUE*	*TRUE*
FALSE		*FALSE*	*FALSE*

*Já o operador **NOT** nega a resposta da sentença, modificando então seu status **TRUE** e **FALSE**. Ou seja:*

***NOT TRUE** é **FALSE** (não verdadeiro é falso)*
***NOT FALSE** é **TRUE** (não falso é verdadeiro)*

*Por exemplo: Queremos saber se o candidato não está apto a trabalhar na empresa, ou seja, se a candidatura dele foi reprovada. Então se ele não está apto a resposta do programa deve ser **TRUE**, caso ele esteja apto, a resposta deve ser **FALSE**.*

Recapitulando: Para ser contratado o candidato precisa: ser maior de idade e morar na cidade de São Paulo, além disso, possuir ensino superior completo ou mais de 10 anos de experiência.

Primeiro criamos a etapa de solicitar os dados

```
idade=int(input('Informe a idade do candidato: '))
cidade=input('Informe a cidade que o candidato reside: ')
ensinoSuperior=input('Possui Ensino Superior Completo? Digite N-Não e S-Sim ')
anosExperiencia=int(input('Anos de experiencia: '))
```

*Agora vamos ver se o candidato **não** está apto a trabalhar na empresa, se não estiver, a resposta do código a seguir será **True***

```
not ((idade>=18 and cidade=='SP') and (ensinoSuperior=='S' or anosExperiencia>10))
```

NOT	Sentença `(idade>=18 and cidade=='SP') and (ensinoSuperior=='S' or anosExperiencia>10)`	Resposta do programa
	TRUE	FALSE
	FALSE	TRUE

*É muito comum que em algumas partes dos programas que desenvolvemos precisamos retornar ou executar algo de acordo com algumas regras estabelecidas. Para isso existem as condicionais **IF**, **ELIF**, **ELSE**.*

Vamos entender melhor o que isso significa:

*Suponha que para ser aprovado em uma disciplina o aluno precisa de nota total >=60 e não possuir mais que 10 faltas. Então vamos construir um programa que, **de acordo com estes requisitos**, retorne a resposta "Aluno aprovado" **ou** "Aluno Reprovado". Observe que o programa precisa decidir qual das duas mensagens mostrar na tela.*

Primeiro criamos a etapa de solicitar os dados

```
nota=float(input('Informe a nota total do aluno: '))
faltas=int(input('Informe a quantidade de faltas do aluno: '))
```

Agora vamos construir as condicionais que retornarão "Aluno aprovado" ou "Aluno Reprovado" de acordo com a entrada informada

```
if nota>=60 and faltas<=10:
 print('Aluno Aprovado')


else:
 print('Aluno Reprovado')
```

*Se a sentença do **IF** é verdadeira, **Aluno Aprovado** é exibido na tela e o programa termina, pois a sentença verdadeira já foi encontrada. Caso contrário, o programa executa o que temos no **else**.*

*Em alguns casos, temos mais opções de sentenças, sendo assim, acrescentamos os **ELIF** na estrutura.*

*Vamos supor que se o aluno não tiver nota >=60 mas sua nota estiver entre 40 e 60 ele tenha a opção de fazer a reavaliação, neste caso ele também precisa ter 10 ou menos faltas. Então precisamos fazer mais uma análise e ter a opção de **Aluno em Reavaliação**.*

```
nota=float(input('Informe a nota total do aluno: '))
faltas=int(input('Informe a quantidade de faltas do aluno: '))

if nota>=60 and faltas<=10:
 print('Aluno Aprovado')
elif nota>=40 and nota<=60 and faltas<=10:
 print('Aluno em Reavaliação')
else:
 print('Aluno Reprovado')
```

*Lembrando que o **elif** pode ser adicionado em maior quantidade, já **if** e **else** teremos apenas um.*

Agora vamos falar sobre as estruturas de repetição. Como o próprio nome já diz, estas estruturas permitem repetir parte do nosso programa.
Okay, vamos entender melhor a necessidade disso: Imagine que queremos mostrar na tela números de 1 a 10. Uma opção seria um programa assim:

```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
print(7)
print(8)
print(9)
print(10)
```

Bom, agora quero um programa que mostre na tela números de 1 a 1.000.000. Imagina se tivermos que criar um milhão de prints? Esse processo seria inviável!

O que podemos fazer neste caso é criar estruturas que repetem os prints incrementando os números de 1 a 1 até que cheguemos no número 1.000.000.

*Existem duas opções de estrutura de repetição, o **while** e o **for**. Usamos **while** geralmente quando não sabemos quantas vezes queremos repetir, mas sabemos a condição de parada. O **for** é usado, geralmente, quando sabemos quantas vezes queremos repetir o código.*

*No exemplo citado vamos usar o **for** porque sabemos que vamos printar 1 milhão de números. O código seria assim:*

```
for x in range(1,1000001):
 print(x)
```

*A função **range** criará uma lista de números de 1 a 1000000, porque ela cria a lista que vai do primeiro número informado como parâmetro (que é o `1`) até menor que o segundo parâmetro (que é `1000001`).*

*A cada repetição do **for**, x receberá um destes valores em ordem. Ou seja, na primeira execução da repetição, x tem valor igual a 1, depois x vale 2, depois 3, 4... até 1000000, então basta printar x para termos todos os números na tela. Muito mais fácil do que criar um milhão de prints não é mesmo?*

*De acordo com o código que criamos acima, temos uma lista com uma quantidade finita de elementos, quando eles "acabam" a repetição para. Por isso, neste caso, usamos o **for**.*

*A lógica da repetição **while** muda um pouco, neste caso a repetição para apenas quando uma determinada situação acontece. Por exemplo:*

Suponha que queremos que o usuário informe números inteiros e vamos mostrar quanto é estes valores multiplicado por 5. Nós sabemos quantos números o usuário vai informar? Não! O programa repetirá esta ação até que o usuário não queira mais informar números. Neste caso, nunca saberemos quantas vezes a repetição vai acontecer, mas sabemos que a condição de parada é quando o usuário informar que não quer continuar.

Uma possível implementação seria a seguinte:

```
continuar='S'
while continuar=='S':
 numero=int(input('Informe um número:'))
 print(numero,'x 5 =',numero*5)
 continuar=input('Deseja continuar? Digite S para sim ou N para
não:')
```

Enquanto a variável `continuar` tem valor igual a S, ou seja, enquanto a sentença `continuar=='S'` é True a repetição é executada, quando a variável `continuar` muda seu valor para N a repetição para, pois a sentença `continuar=='S'` será False.

Lembre-se

Todos estes conceitos vistos no módulo 1 são muito importantes.

Dificilmente você desenvolverá um programa sem usá-los, por isso, é imprescindível que você os compreenda.

Uma boa maneira de fixar tudo isso que foi ensinado é praticando. ***Então, recomendamos que você faça exercícios de programação sempre que puder!***