1. No Python, toda variável é um objeto.
   1. Um objeto é um pouco mais do que uma variável.
2. nome = input("Informe seu nome: ")

#print ("Prazer em te conhecer, **{:20}**!".format(nome))

#No exemplo acima, ele vai escrever o nome num espaço de 20 caracteres

# Vai ficar: "Prazer em te conhecer, antonio             !"

#print ("Prazer em te conhecer, **{:>20}**!".format(nome))

#Acima, com ">", vai alinhar o nome à direita

# Vai ficar: "Prazer em te conhecer,                 João!"

#print ("Prazer em te conhecer, **{:<20}**!".format(nome))

#Com "<", vai alinhar à esquerda

#print ("Prazer em te conhecer, **{:^20}**!".format(nome))

#Com "^", vai alinhar ao centro

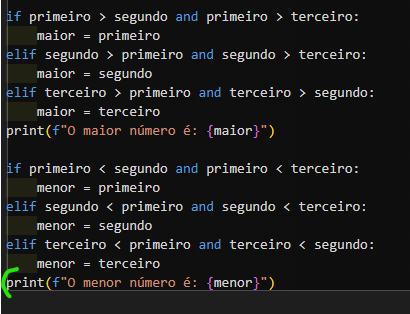
#print ("Prazer em te conhecer, **{:=^20}**!".format(nome))

# No modelo acima, vai alinhar ao centro e imprimir "=" nos espaços vazios

# Vai ficar: "Prazer em te conhecer, ========João========!"

# MODELO MAIS NOVO:

print(f'Prazer em te conhecer, **{nome:=^20}**!'.upper())

1. **O Pyhton aceita variáveis com acentos!**
2. Só usar variável quando esse valor precisar ser resgatado futuramente. Se for só para fins matemáticos, fazer a conta direto no programa para economizar memória.
   1. Pode parecer pouco em programas simples, mas a memória poupada faz diferença em programas maiores e mais complexos.
3. Para importar uma biblioteca, basta digitar “import” e o nome da biblioteca.
   1. Ex: **import math**.
   2. Também podemos importar só um elemento da biblioteca, como: **from math import sqr**t.
   3. Para consultar a biblioteca do Python: <https://docs.python.org/pt-br/3/library/index.html>
   4. Para baixar bibliotecas extras: <https://pypi.org/>
4. Para usar emojis no Python, pegar os códigos de: <https://apps.timwhitlock.info/emoji/tables/unicode>
   1. Substituir o “+” do código unicode por “000” e colocar uma “\” antes de tudo. Então, “U+1F601” deve ficar “\U0001F601”.
5. Cadeia de texto é qualquer string, que é reconhecida pelo Python como texto.
   1. Ex: **frase = “curso de Python”**.
   2. Cada caractere da string, incluindo os espaços, ocupa um micro-espaço na memória do computador.
   3. Ex: 
   4. Com essa característica é possível fatiar a string e manipulá-la. Ex: 
      1. Ao imprimir “**frase[9]**”, o programa vai mostrar o caractere de número 9 dessa string. No caso, será a letra “V”, conforme a divisão mais acima.
   5. 
      1. Ao escrever **frase[9:13]**, o programa vai pegar do 9 até o 13 e excluir o 13. Ou seja, vai imprimir só 9, 10, 11 e 12. É sempre um a menos no final. O começo é certinho.
   6. 
      1. Por outro lado, para conseguir pegar do 9 ao 20, devemos pedir para o programa imprimir do 9 ao 21; embora nossa string vá apenas até 20.
   7. 
      1. No exemplo acima, a instrução significa que o programa vai pegar do 9 ao 21 pulando de dois em dois. Ou seja, neste caso só os espaços ímpares serão considerados.
   8. 
      1. Quando não especificamos o início, o programa começa de 0. No exemplo acima, então, **frase[:5]** é o mesmo que escrever frase**[0:5]**.
   9. 
      1. Da mesma forma, ao não indicar o final, o programa segue até o último elemento. Então, **frase[15:]** seria o mesmo que **frase[15:21]**. Porém, é melhor usar essa forma com o final não especificado do que colocar até um número acima do limite.
   10. 
       1. Seguindo a mesma lógica, o exemplo acima vai do elemento 9 até o final, mostrando de 3 em 3.
   11. Se colocar frase [::2], ele vai pegar do primeiro ao último elemento de 2 em 2.
   12. Também podemos atribuir isso às listas. Ex de lista: [‘João’, ‘Carlos’, ‘Roberto’].
       1. print[0] vai mostrar o primeiro item da lista, que é ‘João’.
       2. Print[0][2] vai mostrar o terceiro caractere do primeiro item da lista, que é “ã”.
6. Para imprimir textos grandes com quebra de linha, basta usar o **print** e “””(três aspas duplas) no início e mais “”” onde deve ser o final.
7. Estruturas condicionais:
   1. Simples: só com um if (pode acontecer ou não).
   2. Composta: mais de um if e/ou um else (acontece uma coisa ou outra)
   3. O elemento que estiver deslocado para a direita, abaixo da condição, será executado se a condição for cumprida. O que estiver alinhado à esquerda será executado independentemente disso. Ex:
   4. 
8. While e for: são o que chamamos de laços/repetições/iterações.