

Variáveis, Tipos de Variáveis e Estruturas Condicionais

Sumário

- Variáveis
- Tipos de variáveis
- Exercício 1 - Variáveis
- Exercício 2 - Compra de supermercado
- Conversão entre tipos de variáveis
- Entrada e saída de dados
- Exercício 3 - Entrada e saída
- Estruturas condicionais
- Operadores de comparação
- Exercício 4 - Compra de supermercado com entrada e saída e condicional
- Exercício 5 - Menu
- Estrutura condicional com múltiplas condições (elif)
- Exercício 6 - Senha

Trabalhando com variáveis

Variável - motivação

Em muitas situações, é útil armazenarmos valores que serão utilizados em cálculos futuros:

Exemplo 1:

$x \leftarrow 5$ (x recebe o valor 5)
 $y \leftarrow 2 * x + 7$ (y recebe o valor $2 * 5 + 7$)

Exemplo 2:

Preço produto 1 \leftarrow R\$12,45

Preço produto 2 \leftarrow R\$16,23

Preço produto 3 \leftarrow R\$8,37

Preço total \leftarrow Preço produto 1 + Preço produto 2 + Preço produto 3

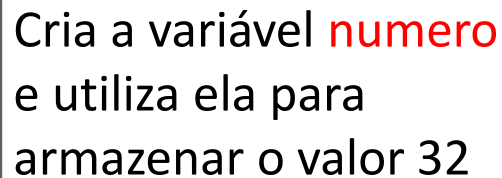
Variável

- Nome dado a valores que são manipulados em um programa
- Em Python, é definida através de um sinal de igualdade:

=

- Exemplo:

```
numero = 32
```



Cria a variável **numero**
e utiliza ela para
armazenar o valor 32

Variável

- O uso do símbolo = para definir uma variável costuma causar confusão, pois seu significado matemático é de igualdade.
- Mas esse é o padrão na maioria das linguagens de programação. Para reduzir a confusão, basta pensarmos que

numero = 32

pode ser entendido como

numero \leftarrow 32

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve **começar** com números (mas pode haver números no meio do nome).

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve **começar** com números (mas pode haver números no meio do nome).

Certo	Errado
numero, nome, asdf, NoMeRaDiCaL, n1	1numero, 50nomes

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve começar com números (mas pode haver números no meio do nome).
 2. Sublinha (também conhecido como *underscore*) pode ser utilizado

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve começar com números (mas pode haver números no meio do nome).
 2. Sublinha (também conhecido como *underscore*) pode ser utilizado

Exemplos:

primeiro_numero

valor_1_esperado

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve começar com números (mas pode haver números no meio do nome).
 2. Sublinha (também conhecido como *underscore*) pode ser utilizado.
 3. Acentos não devem ser utilizados (pois funcionam apenas em alguns casos).

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve começar com números (mas pode haver números no meio do nome).
 2. Sublinha (também conhecido como *underscore*) pode ser utilizado.
 3. Acentos não devem ser utilizados (pois funcionam apenas em alguns casos).
 4. Há diferença entre letras maiúsculas e minúsculas

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve começar com números (mas pode haver números no meio do nome).
 2. Sublinha (também conhecido como *underscore*) pode ser utilizado.
 3. Acentos não devem ser utilizados (pois funcionam apenas em alguns casos).
 4. Há diferença entre letras maiúsculas e minúsculas
Por exemplo, os nomes casa, Casa e CASA são diferentes

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve começar com números (mas pode haver números no meio do nome).
 2. Sublinha (também conhecido como *underscore*) pode ser utilizado.
 3. Acentos não devem ser utilizados (pois funcionam apenas em alguns casos).
 4. Há diferença entre letras maiúsculas e minúsculas.
 5. Alguns nomes são reservados pelo Python, e não podem ser usados como nome de variável

Variável

Palavras reservadas da linguagem Python:

False	def	del	raise
True	if	import	return
None	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

Variável

- Nomes de variáveis possuem algumas regras que devem ser obedecidas:
 1. O nome não deve começar com números (mas pode haver números no meio do nome).
 2. Sublinha (também conhecido como *underscore*) pode ser utilizado.
 3. Acentos não devem ser utilizados (pois funcionam apenas em alguns casos).
 4. Há diferença entre letras maiúsculas e minúsculas.
 5. Alguns nomes são reservados pelo Python
 6. Não podem conter espaços em branco

Variável

- É importante que nomes de variáveis sejam intuitivos:

Suponha que precisamos armazenar o nome de uma pessoa, a rua onde ela mora, o número da casa e o número de filhos que ela possui.

Poderíamos usar as variáveis :

a, b, c, d

Essa nomenclatura causaria diversas confusões em códigos mais complexos. Melhor abordagem:

nome_pessoa, nome_rua, num_casa, num_filhos

Variável

- É importante que nomes de variáveis sejam intuitivos:

Suponha que precisamos armazenar o nome de uma pessoa, a rua onde ela mora, o número da casa e o número de filhos que ela possui.

Poderíamos usar as variáveis :

a, b, c, d

Essa nomenclatura causaria diversas confusões em códigos mais complexos. Melhor abordagem:

nome_pessoa, nome_rua, num_casa, num_filhos

Outra opção:

nomePessoa, nomeRua, numCasa, numFilhos

Tipos de Variáveis

Tipos de variáveis

- Vamos trabalhar com 4 tipos de variáveis:

Tipo	Descrição	Exemplos
Inteiro	Armazena números inteiros	3, 67, -5, -200
Ponto flutuante	Armazena números racionais	0.3, -5.6, 3., 0.0004
String	Usado para manipular texto	"casa", 'carro', "Ele vai sair"
Booleano	Usado em testes de lógica, possui apenas 2 valores possíveis	True, False

Tipos de variáveis - Números

- Existem dois tipos de números em Python:
 - Inteiro: utilizado para representar números inteiros
 - Ponto flutuante: utilizado para representar números reais ("números com vírgula")
- Em Python, ao invés de vírgula utilizamos o caractere . (ponto) para indicar a parte decimal do número.

```
numero1 = 5  
numero2 = 3.45 (representa o número 3,45)
```

Tipos de variáveis - Números

- Ao somarmos um número inteiro com um número ponto flutuante, o resultado é um número ponto flutuante:

```
numero1 = 5  
numero2 = 3.45  
numero3 = numero1 + numero2
```

- **numero3** possuirá o valor 8.45

Tipos de variáveis - Strings

- Outro tipo de variável é a string, utilizada para representar textos.
- Uma string é definida por uma sequência de caracteres envolta em aspas simples ou dupla

```
nome1 = "Luis"  
nome2 = 'Carla'
```

Tipos de variáveis - Strings

CUIDADO! É muito comum que seja feito uma confusão entre uma string e um nome de variável. Os dois comandos abaixo são completamente distintos:

Atribui à variável **nome** a string "Carla":

```
nome = "Carla"
```

Atribui à variável **nome** o conteúdo da variável **Carla**. Nesse caso, a variável **Carla** deve estar definida em algum lugar do código

```
nome = Carla
```


Tipos de variáveis - Strings

- CUIDADO! Um número representado como uma string é diferente de um número inteiro ou ponto flutuante:

```
>>> numero = 3
>>> string = "10"
>>> numero + string
```

Traceback (most recent call last):

```
File "<pyshell#42>", line 1, in <module>
    numero + string
```

TypeError: unsupported operand type(s) for +: 'int' and 'str'

```
>>>
```

Tipos de variáveis - Booleano

- Uma variável booleana pode possuir apenas dois valores: True (verdadeiro) ou False (falso)
- Note que True e False devem começar com letra maiúscula

```
val1 = True  
val2 = False
```

- Esse tipo de variável será útil quando trabalharmos com operações lógicas

Exercício 1 - Variáveis

Defina nomes de variáveis apropriados para armazenar as informações da seguinte tabela:

	Nome	Idade, em anos	Salário	Está empregado
Pessoa 1	João	23	2400.00	Sim
Pessoa 2	Amanda	34	3200.00	Sim
Pessoa 3	Vitor	22	0.00	Não

Solução

Definimos uma variável para cada valor:

```
nome1 = "João"  
nome2 = "Amanda"  
nome3 = "Vitor"  
idade1 = 23  
idade2 = 34  
idade3 = 22  
salario1 = 2400.0  
salario2 = 3200.0  
salario3 = 0.0  
esta_emp1 = True  
esta_emp2 = True  
esta_emp3 = False
```

Exercício 2 - Compra de supermercado

- Faça um programa que calcule o preço total de uma compra de supermercado contendo 3 tipos de produtos. O programa deve executar as seguintes tarefas:
 - VARIÁVEIS:
 1. Número de barras de chocolate compradas
 2. Preço de cada barra de chocolate
 3. Número de potes de sorvete comprados
 4. Preço de cada pote de sorvete
 5. Número de pacotes de bolacha comprados
 6. Preço de cada pacote de bolacha
 - Calcular o preço total a ser pago
 - SAÍDA:
 1. Imprimir na tela o preço total calculado

Solução

```
# Quantidades dos produtos
```

```
q_chocolate = 2
```

```
q_sorvete = 1
```

```
q_bolacha = 3
```

```
# Preço dos produtos
```

```
p_chocolate = 12.3
```

```
p_sorvete = 35.2
```

```
p_bolacha = 15.3
```

```
# Valor total
```

```
total = q_chocolate*p_chocolate + q_sorvete*p_sorvete + q_bolacha*p_bolacha
```

```
print(total)
```

Conversão entre tipos de variáveis

Podemos converter uma variável de um tipo para outro. Para isso, utilizamos uma das funções abaixo:

Nome	Descrição
int(x)	Converte x para um número inteiro
float(x)	Converte x para um número de ponto flutuante
str(x)	Converte x para uma string

Conversão entre tipos de variáveis

Exemplos:

```
>>> float(3)
```

```
3.0
```

```
>>> int(3.0)
```

```
3
```

```
>>> str(3)
```

```
'3'
```

```
>>> str(3.0)
```

```
'3.0'
```

```
>>> int('3')
```

```
3
```

```
>>>
```


Entrada e saída de dados

Entrada e saída

- Entrada e saída de dados é a forma utilizada para interagirmos com o programa.
- A maioria dos programas funciona da seguinte forma:
 1. O usuário entra com os dados;
 2. O programa processa os dados, obtendo uma resposta;
 3. O programa mostra na tela o resultado.
- Em Python, a entrada de dados pelo usuário é feita através da função **input()** e a saída através da função **print()**

Função **print**

Função print

- Em Python, valores podem ser mostrados na tela através da função **print()**
- Em geral, passamos para a função **print()** uma string contendo a mensagem a ser impressa.

Função print

Usamos a função print da seguinte forma:

```
print("mensagem a ser impressa")
```

Exemplo:

```
print("Olá, tudo bem?")
```

Função print

A mensagem deve estar entre aspas duplas porque ela é uma string. Inclusive, é possível criar uma variável com a mensagem e depois imprimi-la:

```
mensagem = "Olá, tudo bem?"  
print(mensagem)
```

Função print

A função `print()` também pode ser usada para imprimir valores de variáveis:

```
numero = 32  
print(numero)
```

Função print

Em muitos casos, precisamos imprimir uma mensagem e também o valor de uma variável. Isso é feito da seguinte forma:

```
valor1 = 12  
valor2 = 73.4
```


```
print(f"valor1 é igual a {valor1} e valor2 é igual a {valor2}")
```


Função print

Em muitos casos, precisamos imprimir uma mensagem e também o valor de uma variável. Isso é feito da seguinte forma:

```
valor1 = 12  
valor2 = 73.4
```

```
print(f"valor1 é igual a {valor1} e valor2 é igual a {valor2}")
```

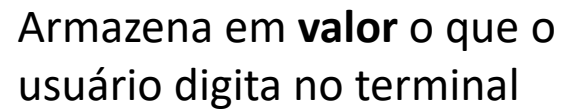


Isto é, para imprimir o valor de uma variável, utilizamos a sintaxe {nome variável}. Também não podemos esquecer de adicionar o caractere **f** antes das primeiras aspas.

Função input

Entrada e saída

```
valor = input()  
print(valor)
```



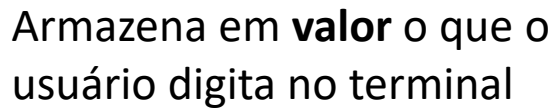
Armazena em **valor** o que o usuário digita no terminal

Notem que o valor retornado pela função **input()** é **sempre uma string!**

Mas e se quisermos que **valor** seja um número?

Entrada e saída

```
valor = input()  
print(valor)
```



Armazena em **valor** o que o usuário digita no terminal

Notem que o valor retornado pela função **input()** é **sempre uma string!**

Mas e se quisermos que **valor** seja um número?
Usamos a função `int()` ou `float()`

Entrada e saída

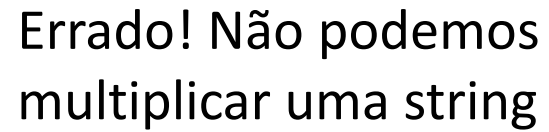
Exemplo de entrada utilizando números:

```
valor = input()  
print(valor*valor)
```

Entrada e saída

Exemplo de entrada utilizando números:

```
valor = input()  
print(valor*valor)
```



Errado! Não podemos multiplicar uma string

Entrada e saída

Exemplo de entrada utilizando números:

```
valor = input()  
print(valor*valor)
```

Errado! Não podemos multiplicar uma string

```
valor = input()  
valor = int(valor)  
print(valor*valor)
```

Forma correta

Importante! O comando `valor = int(valor)` pode parecer um pouco estranho. O significado dele pode ser descrito como:

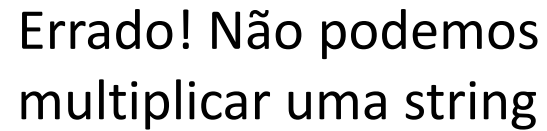
crie uma nova variável chamada `valor` do tipo inteiro e atribua a ela o resultado da conversão para inteiro da variável `valor`, do tipo string.

A variável `valor` do tipo string deixará de existir após o comando.

Entrada e saída

Exemplo de entrada utilizando números:

```
valor = input()  
print(valor*valor)
```



Errado! Não podemos multiplicar uma string

Utilizando int()

```
valor = input()  
valor = int(valor)  
print(valor*valor)
```

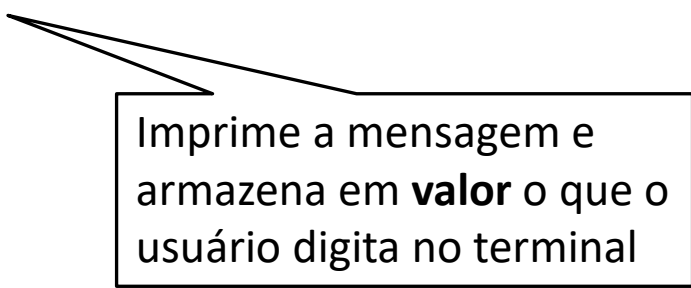
Utilizando float()

```
valor = input()  
valor = float(valor)  
print(valor*valor)
```


Entrada e saída

É possível usar a função **input** para imprimir na tela instruções sobre o que o usuário deve digitar. Para isso, basta passarmos a mensagem a ser impressa como uma string:

```
valor = input("Digite um valor: ")  
print(valor)
```



Imprime a mensagem e armazena em **valor** o que o usuário digita no terminal

Exercício 3 - Entrada e saída

Faça um programa que solicite que o usuário digite um valor qualquer. O programa imprime na tela o valor digitado elevado ao quadrado e ao cubo.

Exemplo:

```
>>> Digite um valor: 4  
16  
64
```

* Não esqueçam de converter o valor digitado (uma string) para inteiro

Solução

```
valor = input("Digite um valor:")  
valor = int(valor)  
res1 = valor*valor  
res2 = valor*valor*valor  
  
print(f"Valor ao quadrado: {res1}")  
print(f"Valor ao cubo: {res2}")
```

Solução 2

Em Python, podemos usar `**` para elevar um número a uma potência. Por exemplo, a expressão `4**2` dará o resultado 16.

```
valor = input("Digite um valor:")
valor = int(valor)
res1 = valor**2
res2 = valor**3

print(f"Valor ao quadrado: {res1}")
print(f"Valor ao cubo: {res2}")
```

Estruturas condicionais

Estrutura condicional

- Utilizada quando os comandos a serem realizados dependem de uma condição ser verdadeira ou falsa

Estrutura condicional

- Utilizada quando os comandos a serem realizados dependem de uma condição ser verdadeira ou falsa
- Possui a sintaxe:

```
if condição:  
    comando 1  
    comando 2  
    .  
    .  
    .  
else:  
    comando 1  
    comando 2  
    .  
    .  
    .
```

Estrutura condicional

- Utilizada quando os comandos a serem realizados dependem de uma condição ser verdadeira ou falsa
- Possui a sintaxe:

```
if condição:  
    comando 1  
    comando 2
```

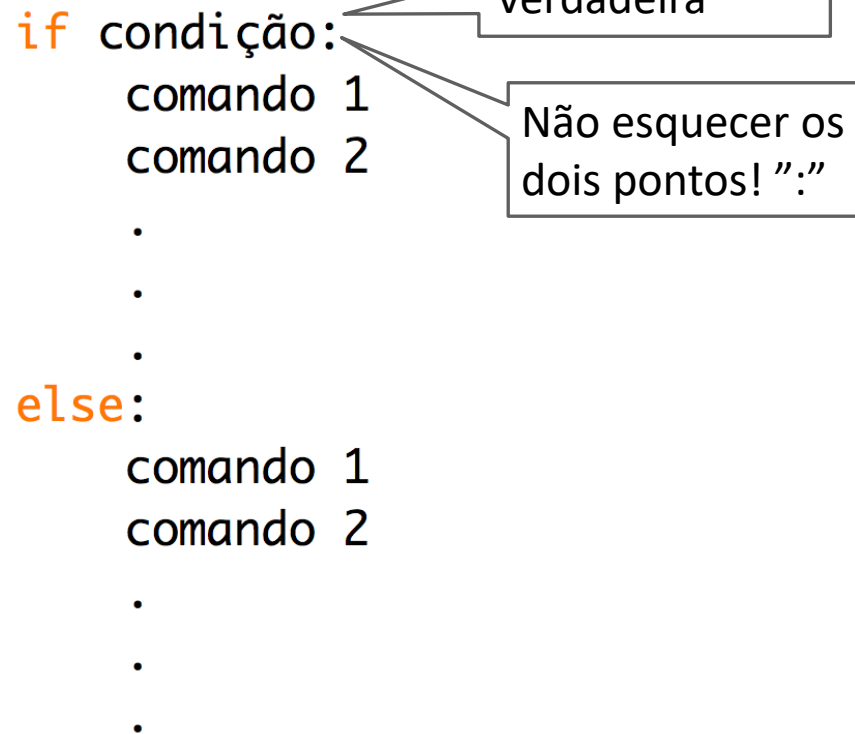
Se condição for verdadeira

```
    .  
    .  
    .  
else:  
    comando 1  
    comando 2  
    .  
    .  
    .
```


Estrutura condicional

- Utilizada quando os comandos a serem realizados dependem de uma condição ser verdadeira ou falsa
- Possui a sintaxe:

```
if condição:  
    comando 1  
    comando 2  
    .  
    .  
    .  
else:  
    comando 1  
    comando 2  
    .  
    .  
    .
```




Se condição for verdadeira

Não esquecer os dois pontos! ":"

Estrutura condicional

- Utilizada quando os comandos a serem realizados dependem de uma condição ser verdadeira ou falsa
- Possui a sintaxe:

```
if condição:
    comando 1
    comando 2
    .
    .
    .
else:
    comando 1
    comando 2
    .
    .
    .
```



Comandos a serem executados

Estrutura condicional

- Utilizada quando os comandos a serem realizados dependem de uma condição ser verdadeira ou falsa
- Possui a sintaxe:

```
if condição:  
    comando 1  
    comando 2  
    .  
    .  
    .  
else:  
    comando 1  
    comando 2  
    .  
    .  
    .
```

Se a condição for falsa,
execute os comandos abaixo

Estrutura condicional

- Indentação é fundamental em Python!!!!

```
if condição:
    comando 1
    comando 2
    .
    .
    .
else:
    comando 1
    comando 2
    .
    .
    .
```

Primeira coluna
do arquivo

Estrutura condicional

- Indentação é fundamental em Python!!!!

if condição:

```
comando 1  
comando 2  
.  
.  
.
```

else:

```
comando 1  
comando 2  
.  
.  
.
```

- Segundo nível de indentação. Pode ser qualquer número de espaços, mas **o número de espaços deve ser igual para todas as linhas**
- Exemplo: 4 espaços, 1 tab, etc

Estrutura condicional

- A estrutura **if** pode ser utilizada sem um **else**

```
if condição:  
    comando 1  
    comando 2  
    .  
    .  
    .
```

Operadores de comparação

Operador	Significado
==	Igual a
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a
!=	Não é igual a


Atribuição ou igualdade?

A diferença entre os dois comandos abaixo é muito importante:


`variavel1 = variavel2`

`variavel1 == variavel2`

Atribui o valor que está em
variavel2 para **variavel1**



Verifica se o valor de **variavel1** é
igual ao valor de **variavel2**



Operador and

Quando precisamos testar mais de uma condição ao mesmo tempo, podemos utilizar o operador lógico **and**. Por exemplo:

```
if var1==var2 and var3>var4:  
    print('Ambas as condições são verdadeiras')  
else:  
    print('Uma das condições é falsa')
```

Exercício 4 - Compra de supermercado com entrada e saída e condicional

Faça um programa que imprima diferentes tipos de mensagens na tela dependendo do valor total de uma compra de supermercado. O mercado possui apenas 3 produtos: chocolate, sorvete e bolacha.

- Os preços dos produtos são definidos por variáveis no próprio programa.
- ENTRADA:
 - Quantidade de chocolates, sorvetes e bolachas compradas
- Calcular o preço total a ser pago
- SAÍDA:
 - Se o preço total for menor que R\$ 50.00, imprimir “Pagar no pix”
 - Se o preço total for maior ou igual a R\$ 50.00, e menor que R\$ 150, imprimir “Pagar no cartão”
 - Se o preço for maior ou igual a R\$ 150.00, imprimir “Fazer empréstimo”

Solução

```
# Preço dos produtos
```

```
p_chocolate = 12.5
```

```
p_sorvete = 35.2
```

```
p_bolacha = 15.3
```

```
# Quantidades dos produtos
```

```
q_chocolate = int(input("Quantidade de chocolate: "))
```

```
q_sorvete = int(input("Quantidade de sorvete: "))
```

```
q_bolacha = int(input("Quantidade de bolacha: "))
```

```
# Valor total
```

```
total = q_chocolate*p_chocolate + q_sorvete*p_sorvete + q_bolacha*p_bolacha
```

```
print("Total da compra:", total)
```

```
if total<50:
```

```
    print("Pagar no pix")
```

```
if total>=50 and total<150:
```

```
    print("Pagar no cartão")
```

```
if total>=150:
```

```
    print("Fazer empréstimo")
```

Exercício 5 - Menu

Faça um programa que imprima o seguinte menu de opções:

Menu

1. X-burguer
2. X-bacon
3. X-tudo

O programa então solicita que o usuário digite uma opção. Uma mensagem específica para cada opção é impressa na tela.

Solução

```
print('Menu')
print('-----')
print('1. X-burguer')
print('2. X-bacon')
print('3. X-tudo')
opcao = int(input('Escolha uma opção: '))

if opcao==1:
    print('Opção escolhida: X-burguer')
if opcao==2:
    print('Opção escolhida: X-bacon')
if opcao==3:
    print('Opção escolhida: X-tudo')
```

Estrutura condicional com múltiplas condições

Estrutura condicional com múltiplas condições

Como podemos imprimir na tela uma mensagem de erro caso o usuário digite um valor diferente de 1, 2 ou 3?

```
print("Menu")
print("-----")
print("1. X-burguer")
print("2. X-bacon")
print("3. X-tudo")
opcao = int(input("Escolha uma opção: "))

if opcao==1:
    print("Opção escolhida: X-burguer")
if opcao==2:
    print("Opção escolhida: X-bacon")
if opcao==3:
    print("Opção escolhida: X-tudo")
```

Estrutura condicional com múltiplas condições

Podemos adicionar um *if* ao final:

```
print("-----")
print("1. X-burguer")
print("2. X-bacon")
print("3. X-tudo")
opcao = int(input("Escolha uma opção: "))

if opcao==1:
    print("Opção escolhida: X-burguer")
if opcao==2:
    print("Opção escolhida: X-bacon")
if opcao==3:
    print("Opção escolhida: X-tudo")
if opcao!=1 and opcao!=2 and opcao!=3:
    print("Opção inválida")
```

Mas e se o menu tivesse 20 opções?

Estrutura condicional com múltiplas condições

Outra opção é adicionar um ***else***:

```
print("Menu")
print("-----")
print("1. X-burguer")
print("2. X-bacon")
print("3. X-tudo")
opcao = int(input("Escolha uma opção: "))

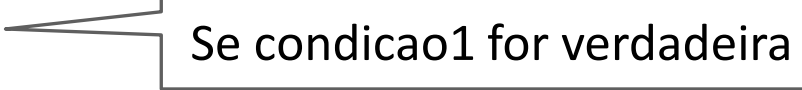
if opcao==1:
    print("Opção escolhida: X-burguer")
if opcao==2:
    print("Opção escolhida: X-bacon")
if opcao==3:
    print("Opção escolhida: X-tudo")
else:
    print("Opção inválida")
```

Mas esse código está errado! Por exemplo, se o usuário digitar 1, o programa imprimirá 'Opção escolhida: X-burguer' e também imprimirá 'Opção inválida'

Estrutura condicional com múltiplas condições

- Sintaxe para testar diversas condições

```
if condicao1:  
    comandos  
    ...  
elif condicao2:  
    comandos  
    ...  
elif condicao3:  
    comandos  
    ...  
else:  
    comandos  
    ...
```



Se condicao1 for verdadeira

Estrutura condicional com múltiplas condições

- Sintaxe para testar diversas condições

```
if condicao1:  
    comandos
```

```
...
```

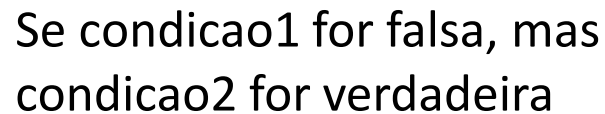
```
elif condicao2:  
    comandos
```

```
...
```

```
elif condicao3:  
    comandos
```

```
...
```

```
else:  
    comandos  
...
```

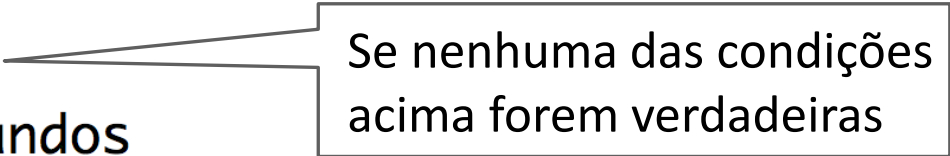


Se condicao1 for falsa, mas
condicao2 for verdadeira

Estrutura condicional com múltiplas condições

- Sintaxe para testar diversas condições

```
if condicao1:  
    comandos  
    ...  
elif condicao2:  
    comandos  
    ...  
elif condicao3:  
    comandos  
    ...  
else:  
    comandos  
    ...
```



Se nenhuma das condições acima forem verdadeiras

Estrutura condicional com múltiplas condições

O código de menu fica assim:

```
print("Menu")
print("-----")
print("1. X-burguer")
print("2. X-bacon")
print("3. X-tudo")
opcao = int(input("Escolha uma opção: "))

if opcao==1:
    print("Opção escolhida: X-burguer")
elif opcao==2:
    print("Opção escolhida: X-bacon")
elif opcao==3:
    print("Opção escolhida: X-tudo")
else:
    print("Opção inválida")
```

Estrutura condicional com múltiplas condições

- Importante! Notem a diferença entre os dois códigos abaixo

```
if opcao==1:  
    print('Opção escolhida: X-burguer')  
if opcao==2:  
    print('Opção escolhida: X-bacon')  
if opcao==3:  
    print('Opção escolhida: X-tudo')  
else:  
    print('Opção inválida')
```

```
if opcao==1:  
    print('Opção escolhida: X-burguer')  
elif opcao==2:  
    print('Opção escolhida: X-bacon')  
elif opcao==3:  
    print('Opção escolhida: X-tudo')  
else:  
    print('Opção inválida')
```

Estrutura condicional com múltiplas condições

- Importante! Notem a diferença entre os dois códigos abaixo

```
if opcao==1:  
    print('Opção escolhida: X-burguer')  
if opcao==2:  
    print('Opção escolhida: X-bacon')  
if opcao==3:  
    print('Opção escolhida: X-tudo')  
else:  
    print('Opção inválida')
```

Se **opcao** for 1 ou 2, vai imprimir
'Opção inválida'

```
if opcao==1:  
    print('Opção escolhida: X-burguer')  
elif opcao==2:  
    print('Opção escolhida: X-bacon')  
elif opcao==3:  
    print('Opção escolhida: X-tudo')  
else:  
    print('Opção inválida')
```

Estruturas condicionais aninhadas

Estruturas condicionais aninhadas

É comum termos que incluir estruturas condicionais dentro de outras estruturas condicionais!

Podemos incluir uma nova estrutura condicional em qualquer linha da primeira condicional:

```
if condicao1:
    print('condicao1 verdadeira!')
    if condicao2:
        print('condicao1 e condicao2 verdadeiras!')
    else:
        print('condicao1 verdadeira e condicao2 falsa!')
else:
    print('condicao1 falsa!')
```

Estruturas condicionais aninhadas

Mas é importante lembrarmos da **identação**

Os comandos executados dentro da segunda condicao devem ter um espaçamento adicional no começo da linha

```
if condicao1:  
    → print('condicao1 verdadeira!')  
    → if condicao2:  
        → print('condicao1 e condicao2 verdadeiras!')  
    → else:  
        → print('condicao1 verdadeira e condicao2 falsa!')  
else:  
    → print('condicao1 falsa!')
```

Exercício 6 - Senha

Faça um programa que execute as seguintes tarefas:

- O programa solicita que o usuário digite a opção "entrar" ou "sair"
- Se o usuário digitar "entrar", o programa então solicita que seja digitado um nome de usuário e senha. Caso os dados estejam corretos, o programa imprime uma mensagem de sucesso. Uma mensagem de erro é impressa caso contrário.
- Se o usuário digitar "sair", é impressa uma mensagem de despedida.

Solução 1

```
# Valores constantes definidos no código. Eles
# são comparados com o que o usuário digitar
USUARIO = "paulo"
SENHA = "ghet45"

operacao = input("Digite entrar ou sair: ")

if operacao=="entrar":
    usuario = input("Usuário: ")
    senha = input("Senha: ")
    if usuario==USUARIO and senha==SENHA:
        print("Você entrou no sistema!")
    else:
        print("Usuário ou senha incorreto!")
elif operacao=="sair":
    print("Tchau!")
else:
    print("Opção inválida")
```

Solução 2

```
# Valores constantes definidos no código. Eles
# são comparados com o que o usuário digitar
USUARIO = "paulo"
SENHA = "ghet45"

operacao = input("Digite entrar ou sair: ")

if operacao=="entrar":
    usuario = input("Usuário: ")
    if usuario==USUARIO:
        senha = input("Senha: ")
        if senha==SENHA:
            print("Você entrou no sistema!")
        else:
            print("Senha incorreta")
    else:
        print("Usuário incorreto")
elif operacao=="sair":
    print("Tchau!")
else:
    print("Opção inválida")
```