



 [Asabeneh](#) / [30 dias de Python](#) Público[Código](#) [Problemas 16](#) [Solicitações pull 83](#) [Ações](#) [Projetos](#) [Segurança](#) [Perc](#) [mestre](#) [30 dias de Python / 09_Dia_Condicionais](#)
[/ 09_conditionals.md](#) [Asabeneh](#) 2 years ago

278 lines (217 loc) · 7.91 KB

[Visualização](#)[Código](#)[Culpa](#)


30 dias de Python: Dia 9 - Condicionais

[Follow @asabeneh](#)Autor: [Asabeneh Yetayeh](#)

Segunda edição: julho de 2021

[<< Dia 8](#) | [Dia 10 >>](#)

- [Dia 9](#)
 - [Condicionais](#)
 - [Se condição](#)
 - [Se mais](#)
 - [Se Elif mais](#)
 - [Forma abreviada](#)
 - [Condições aninhadas](#)

- [Se Condição e Operadores Lógicos](#)
- [Operadores Lógicos Se e Ou](#)
-  [Exercícios: Dia 9](#)
 - [Exercícios: Nível 1](#)

Dia 9

Condicionais

Por padrão, as instruções no script Python são executadas sequencialmente de cima para baixo. Caso a lógica de processamento assim o exija, o fluxo sequencial de execução pode ser alterado de duas maneiras:

- Execução condicional: um bloco de uma ou mais instruções será executado se uma determinada expressão for verdadeira
- Execução repetitiva: um bloco de uma ou mais instruções será executado repetidamente enquanto uma determinada expressão for verdadeira. Nesta seção, abordaremos as instruções *if*, *else* e *elif*. A comparação e os operadores lógicos que aprendemos nas seções anteriores serão úteis aqui.

Se condição

Em python e outras linguagens de programação, a palavra-chave *if* é usada para verificar se uma condição é verdadeira e para executar o código do bloco. Lembre-se do recuo após os dois pontos.

```
# syntax
if condition:
    this part of code runs for truthy conditions
```



Exemplo 1

```
a = 3
if a > 0:
    print('A is a positive number')
# A is a positive number
```



Como você pode ver no exemplo acima, 3 é maior que 0. A condição era verdadeira e o código do bloco foi executado. Porém, se a condição for falsa, não vemos o resultado. Para ver o resultado da condição falsa, devemos ter outro bloco, que será *else*.

Se mais

Se a condição for verdadeira, o primeiro bloco será executado; caso contrário, a condição *else* será executada.

```
# syntax
if condition:
```



```
    this part of code runs for truthy conditions
else:
    this part of code runs for false conditions
```

****Exemplo: ****

```
a = 3
if a < 0:
    print('A is a negative number')
else:
    print('A is a positive number')
```



A condição acima é falsa, portanto o bloco `else` foi executado. Que tal se nossa condição for maior que dois? Poderíamos usar `_ elif _`.

Se Elif mais

Em nossa vida diária, tomamos decisões diariamente. Tomamos decisões não verificando uma ou duas condições, mas múltiplas condições. Assim como a vida, a programação também é cheia de condições. Usamos *elif* quando temos múltiplas condições.

```
# syntax
if condition:
    code
elif condition:
    code
else:
    code
```



****Exemplo: ****

```
a = 0
if a > 0:
    print('A is a positive number')
elif a < 0:
    print('A is a negative number')
else:
    print('A is zero')
```



Forma abreviada

```
# syntax
code if condition else code
```



****Exemplo: ****

```
a = 3
print('A is positive') if a > 0 else print('A is negative') # first condition met, 'A
```



Nested Conditions

Conditions can be nested

```
# syntax
if condition:
    code
    if condition:
        code
```



Example:

```
a = 0
if a > 0:
    if a % 2 == 0:
        print('A is a positive and even integer')
    else:
        print('A is a positive number')
elif a == 0:
    print('A is zero')
else:
    print('A is a negative number')
```



We can avoid writing nested condition by using logical operator *and*.

If Condition and Logical Operators

```
# syntax
if condition and condition:
    code
```



Example:

```
a = 0
if a > 0 and a % 2 == 0:
    print('A is an even and positive integer')
elif a > 0 and a % 2 != 0:
    print('A is a positive integer')
elif a == 0:
    print('A is zero')
else:
    print('A is negative')
```



If and Or Logical Operators

```
# syntax
if condition or condition:
    code
```



Example:

```
user = 'James'
access_level = 3
if user == 'admin' or access_level >= 4:
    print('Access granted!')
else:
    print('Access denied!')
```



👉 You are doing great. Never give up because great things take time. You have just completed day 9 challenges and you are 9 steps a head in to your way to greatness. Now do some exercises for your brain and muscles.

Exercises: Day 9

Exercises: Level 1

1. Get user input using input("Enter your age: "). If user is 18 or older, give feedback: You are old enough to drive. If below 18 give feedback to wait for the missing amount of years. Output:

```
Enter your age: 30
You are old enough to learn to drive.
Output:
Enter your age: 15
You need 3 more years to learn to drive.
```



2. Compare the values of my_age and your_age using if ... else. Who is older (me or you)? Use input("Enter your age: ") to get the age as input. You can use a nested condition to print 'year' for 1 year difference in age, 'years' for bigger differences, and a custom text if my_age = your_age. Output:

```
Enter your age: 30
You are 5 years older than me.
```



3. Get two numbers from the user using input prompt. If a is greater than b return a is greater than b, if a is less b return a is smaller than b, else a is equal to b. Output:

```
Enter number one: 4
Enter number two: 3
4 is greater than 3
```



Exercises: Level 2



1. Write a code which gives grade to students according to their scores:

```
80-100, A
70-89, B
60-69, C
50-59, D
0-49, F
```



2. Check if the season is Autumn, Winter, Spring or Summer. If the user input is: September, October or November, the season is Autumn. December, January or February, the season is Winter. March, April or May, the season is Spring June, July or August, the season is Summer
3. A lista a seguir contém algumas frutas:

```
fruits = ['banana', 'orange', 'mango', 'lemon']
```



Se não existir uma fruta na lista, adicione a fruta à lista e imprima a lista modificada. Se a fruta existir print('Essa fruta já existe na lista')

Exercícios: Nível 3

4. Aqui temos um dicionário pessoal. Sinta-se à vontade para modificá-lo!

```
person={
    'first_name': 'Asabeneh',
    'last_name': 'Yetayeh',
    'age': 250,
    'country': 'Finland',
    'is_marred': True,
    'skills': ['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address': {
        'street': 'Space street',
        'zipcode': '02210'
    }
}
```



* Check if the person dictionary has skills key, if so print out the middle skill in the skills list.



* Check if the person dictionary has skills key, if so check if the person has 'Python' skill and print out the result.

* If a person skills has only JavaScript and React, print('He is a front end developer'), if the person skills has Node, Python, MongoDB, print('He is a backend developer'), if the person skills has React, Node and MongoDB, Print('He is a fullstack developer'), else print('unknown title') - for more accurate results more conditions can be nested!

* If the person is married and if he lives in Finland, print the information in the following format:



Asabeneh Yetayeh lives in Finland. He is married.



🎉 PARABÉNS! 🎉

<< Dia 8 | Dia 10 >>