

3 Aula 03: 08/SET/2020



3.1 Aula passada

- variáveis ✓
- expressões e operadores aritméticos ✓
- atribuição ✓
- escrita na tela (= `print()`) ✓
- leitura via teclado (= `input()`) ✓
- tipos `str`, `int` e `float`
- f-strings ← *sc x y*
- funções de conversão `int()`, `float()`
- Spyder

```
#!/usr/bin/python3.3
# -*- coding: utf-8 -*-
3
4 a = int(input("Digite o primeiro número: "))
5 b = int(input("Digite o segundo número: "))
6 print(f"A soma de {a} e {b} é {a+b}")
7
```

Python 3.8.3 (default, Jul 2 2020, 16:21:59)
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python

In [1]: $20 * 9 / 5 + 32$ # para fahrenheit
Out[1]: 68.0

In [2]: runfile('/home/coelho/aulas/mac0110/aula02/py/tentativa6.py', wdir='/home/coelho/aulas/mac0110/aula02/py')

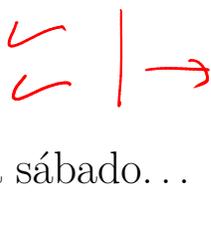
Digite o primeiro número: 34
Digite o segundo número: 12
A soma de 34 e 12 é 46

In [3]:

3.2 Hoje

- **mais** variáveis
 - **mais** atribuição
 - **mais** escrita na tela (= `print()`)
 - **mais** leitura via teclado (= `input()`)
 - **mais** tipos `str`, `int` e `float`
 - **mais** funções de conversão `int()`, `float()`
 - **mais** expressões e operadores relacionais
 - **mais** Spyder
 - **tópico novo**: comando de repetição `while`
 - **tópico novo**: operadores relacionais
 - **tópico novo**: tipo `bool`
 - **tópico novo**: função `type()`
- 

3.3 Avisos

- EP01 deve ser entregue até 14/09
 - EP02 deve ser entregue até 21/09
 - provinha **p02** disponível de sexta a sábado...
- 

3.4 Exercício: soma de inteiros

especificação

Dada uma seqüência de números inteiros diferentes de zero, terminada por um zero, calcular a sua soma. Por exemplo, para a seqüência

12 17 4 -6 8 0

resposta

o seu programa deve escrever o número 35.

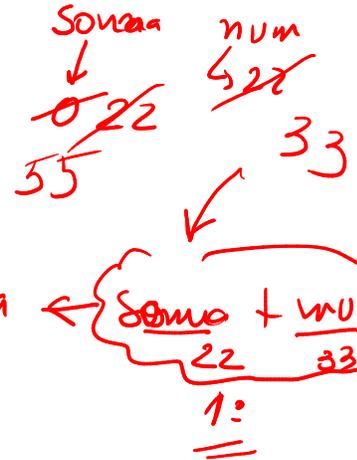
1ª aula

3.4.1 Exemplos

Digite um inteiro: 10
Digite um inteiro: 23
Digite um inteiro: 5
Digite um inteiro: 4
Digite um inteiro: -15
Digite um inteiro: 0
A soma é 27

ENTER

Parar



Digite um inteiro: 1
Digite um inteiro: 2
Digite um inteiro: 3
Digite um inteiro: 4
Digite um inteiro: 5
Digite um inteiro: 6
Digite um inteiro: 0
A soma é 21

$$1 + 2 + 3 + \dots + 6 = 21$$

3.4.2 Solução

```
soma = 0 # atribuição
```

```
num = int(input("Digite um inteiro: "))
```

```
# print("Número lido =", num)
```

```
while num != 0:
```

```
    soma = soma + num # atribuição soma ← soma + num
```

```
    num = int(input("Digite um inteiro: "))
```

```
    # print("Número lido =", num)
```

```
print(f"A soma é {soma}")
```

f" { } "

↑

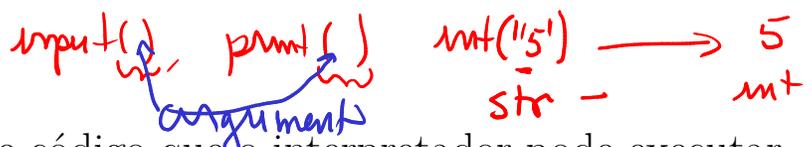
↓

valor

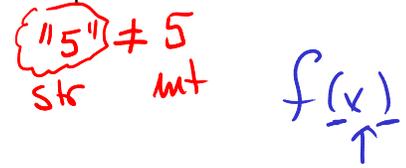
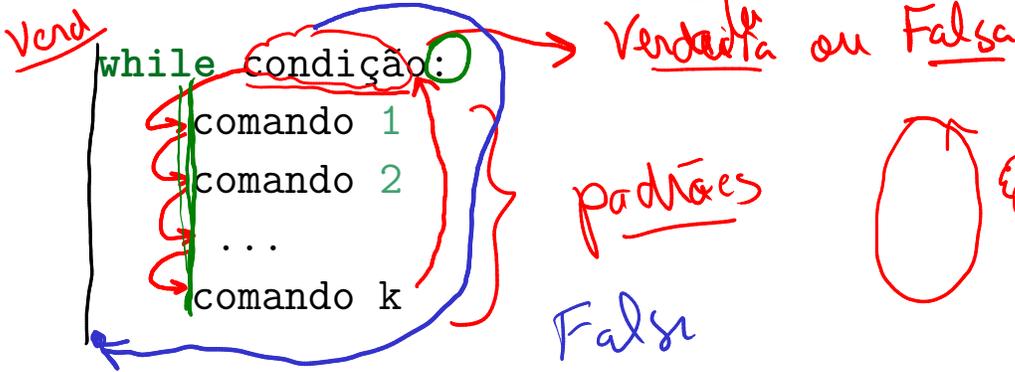
f-string

3.5 Comando

Um **comando** é uma unidade de código que o interpretador pode executar.



3.6 Comando de repetição while



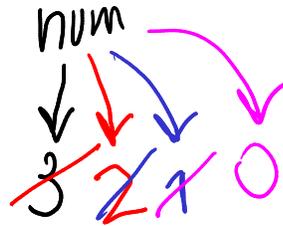
3.6.1 Significado

Enquanto a **condição** é verdadeira executa os comandos.

Os comandos a serem repetidos **devem** ter a mesma tabulação.

3.6.2 Exemplos

O trecho de código:



```
num = 3
while num > 0:
    num = num - 1 # num ← (num - 1)
    print("num =", num)
```

Saída
num = 2
num = 1
num = 0

imprime:

```
num = 2
num = 1
num = 0
```

O trecho de código:

```

num = 3
while num > 0:
    num = num - 1
print("num =", num)

```

Handwritten annotations:
 - "falso" (false) written in blue next to the while loop.
 - "verdadeiro" (true) written in red above the loop.
 - A diagram shows the variable 'num' starting at 3, then decreasing to 2, 1, and finally 0, which is underlined. Arrows indicate the flow of the loop.

imprime:

```

num = 0
num = 3
num = 3
num = 3
...

```

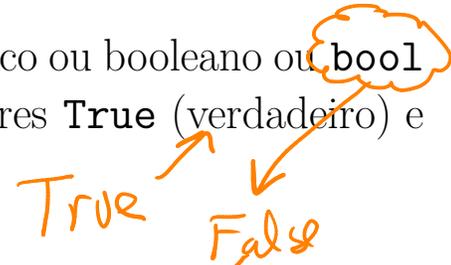
Handwritten annotations:
 - The first line 'num = 0' is written in red.
 - The subsequent lines are crossed out with a red 'X'.
 - A blue arrow points downwards from the first line.

Qual é a saída?
O queo imprime?

Handwritten flow: -1 0 2 27 → input() → "15" str → int() → 15 int

3.7 Tipo bool

Além de **int**, **str** e **float**, Python tem como o tipo lógico ou booleano ou **bool** como nativo. **bool** é um tipo que tem apenas dois valores **True** (verdadeiro) e **False** (falso).



3.8 Expressões lógicas

Condições ou **expressões lógicas** são expressões cujo valor é **True** ou **False** e usam os operadores relacionais:

- **>** maior;
- **>=** maior ou igual;
- **<** menor;
- **<=** menor ou igual;
- **==** igual; ou
- **!=** diferente.

Handwritten notes:

- "condição" written in blue above a bracket grouping the list items.
- "5 == 5" with arrows pointing to the equals signs, and a note "é atribuição" (is assignment) written in orange.
- "while condição:" written in orange, with "condição" circled in a cloud shape.
- A crossed-out "!=" symbol is also present.

3.9 Exercício: primeiros ímpares

Dado um número inteiro positivo n , imprimir os n primeiros naturais ímpares.

Exemplo: Para $n=4$ a saída deverá ser 1, 3, 5, 7.

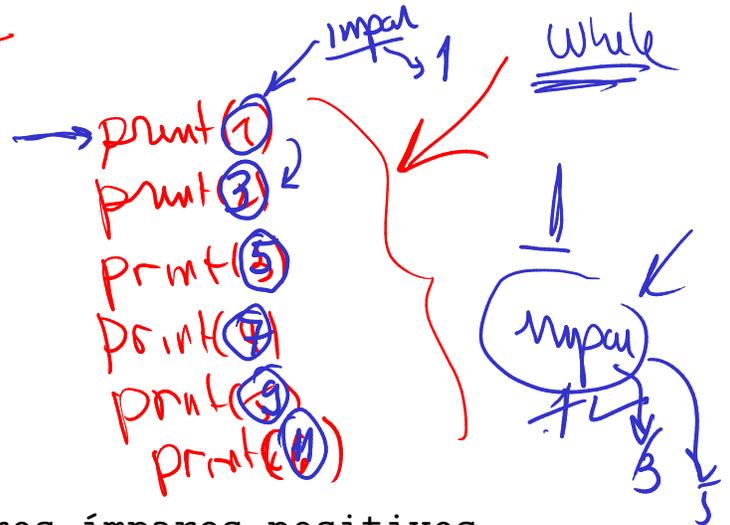
3.9.1 Exemplos

Gerador dos n primeiros números ímpares positivos

Digite o valor de n : 6

1
3
5
7
9
11

~~print(2)~~



Gerador dos n primeiros números ímpares positivos

Digite o valor de n : 3

1
3
5

3.9.2 Solução

```
'''
```

Programa que lê um inteiro positivo n e imprime os n primeiros inteiros ímpares positivos.

Exemplo de execução:

Digite o valor de n: 3

1

3

5

>>>

```
'''
```

```
print("Gerador do n primeiros números ímpares positivos\n")
```

```
# 1 leia o valor de n
```

```
n = int(input("Digite o valor de n: "))
```

```
# 2 contador de ímpares impressos
```

```
i = 0
```

```
# 3 primeiro número ímpar
```

```
impar = 1
```

```
# 4 imprima os n primeiros ímpares, um por linha
```

```
while i < n:
```

```
    # 4.1 imprima o próximo número ímpar
```

```
    print(impar)
```

```
    # 4.2 incremente p contador
```

```
    i = i + 1
```

```
    # 4.3 determine o próximo ímpar
```

```
    impar = impar + 2
```