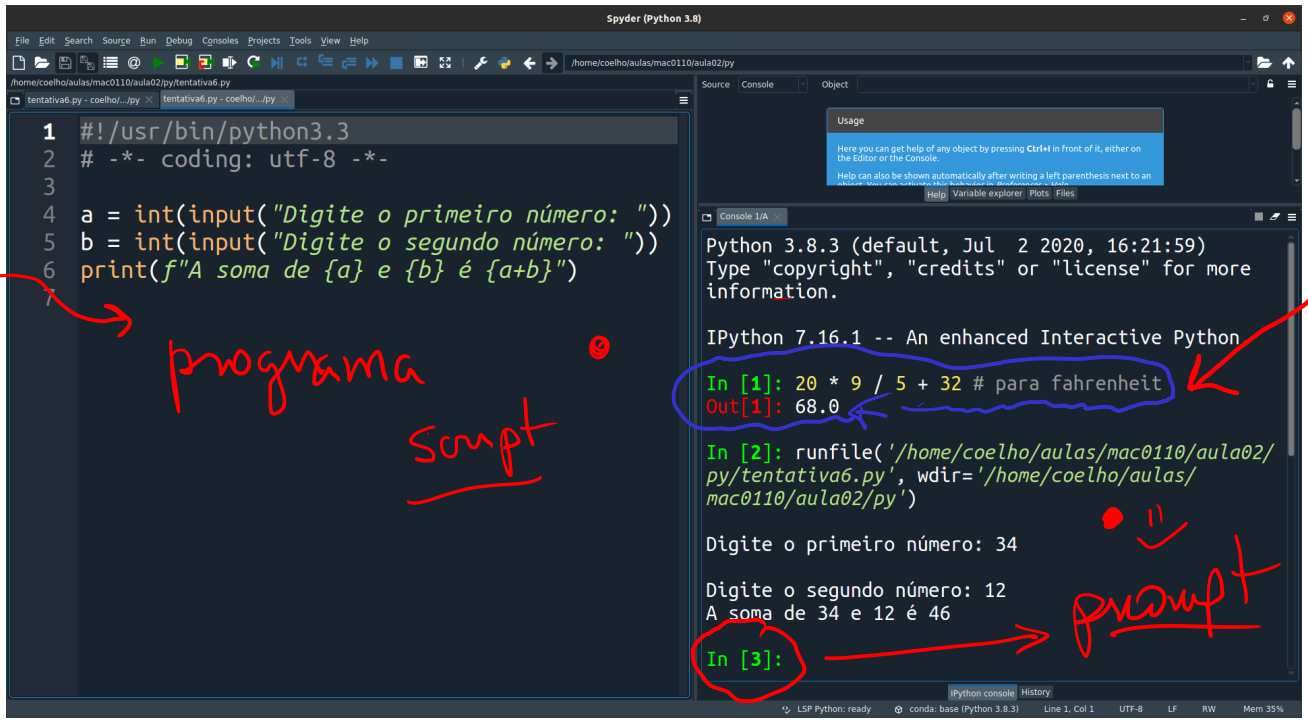
3 Aula 03: 08/SET/2020

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
- funções de conversão `int()`, `float()`
- Spyder



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

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

```
12    17    4    -6    8    0
```

o seu programa deve escrever o número 35.

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
```

```
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
```

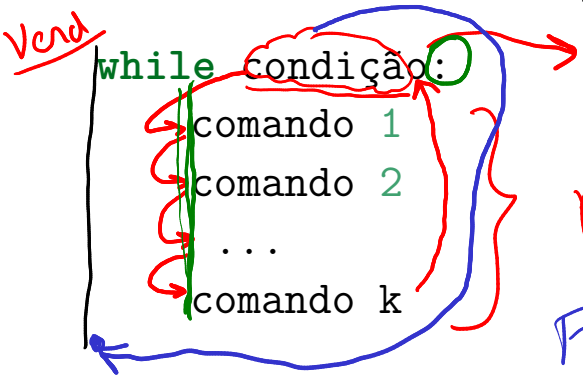
3.4.2 Solução

```
soma = 0          # atribuicao
num = int(input("Digite um inteiro: "))
# print("Número lido =", num)
while num != 0:
    soma = soma + num   # atribuicão
    num = int(input("Digite um inteiro: "))
    # print("Número lido =", num)
print(f"A soma é {soma}")
```

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)
```

imprime:

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

O trecho de código:

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

imprime:

```
num = 3
num = 3
num = 3
...
```

Qual é a saída? O que imprime?

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.

3.9 Exercício: primeiros ímpares

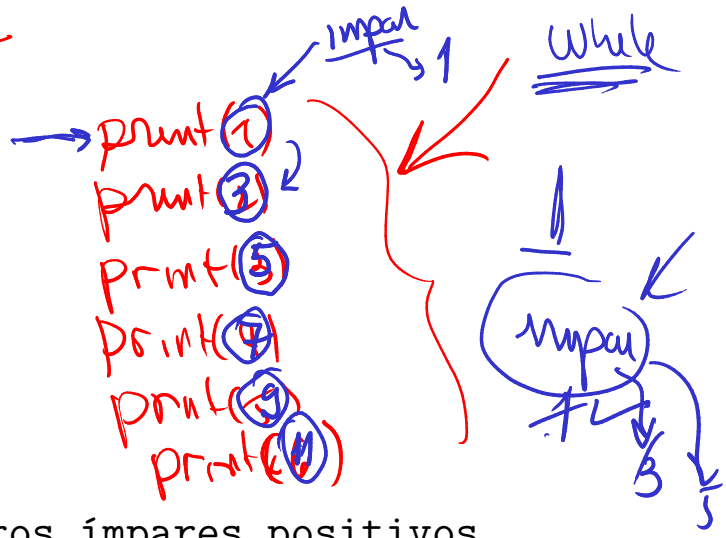
Dado um número inteiro positivo `n`, imprimir os `n` primeiros naturais ímpares.

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
```



```
Gerador dos n primeiros números ímpares positivos

Digite o valor de n: 3
1
3
5
```

3.9.2 Solução

```
'''
Programe 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 impares, 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
```