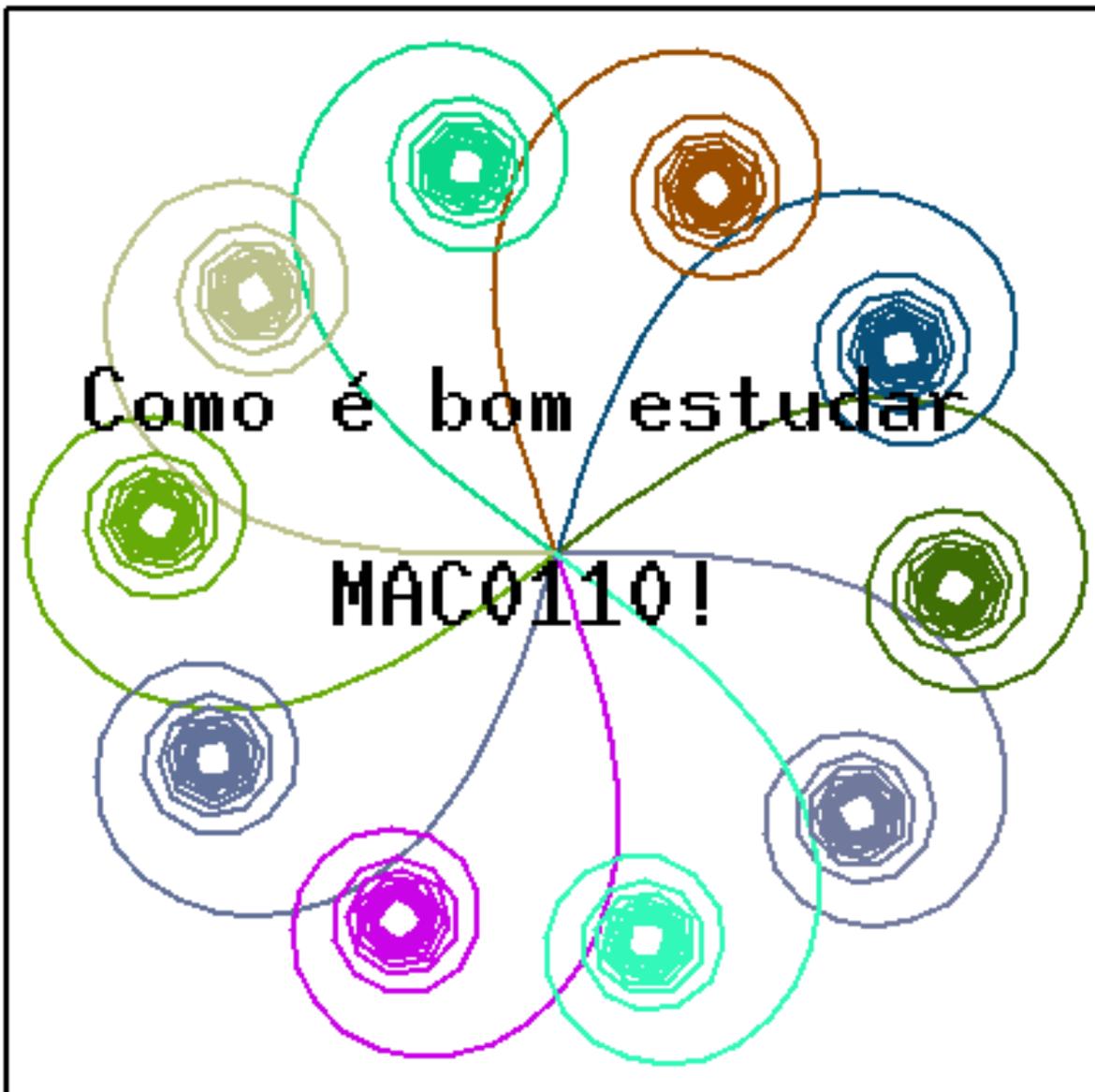


7 Reunião 07: 22/SET/2020

7.1 Aulas passadas

- execução condicional if: *conta pares e ímpares, notas finais*
- execução alternativa if-else: *conta pares e ímpares, estatística de notas*
- execução condicional em cadeia if-elif-else: *estatística de notas*



7.2 Exercício: múltiplos de i ou j

Dados n e dois números inteiros positivos i e j diferentes de 0, imprimir em **ordem crescente** os n primeiros naturais que são múltiplos de i ou de j ou de ambos.

Para os dados $n == 8$, $i == 2$ e $j == 3$ o programa deve imprimir

0 2 3 4 6 8 9 10

7.2.1 Exemplo

```

Digite n: 8
Digite i: 2
Digite j: 3
0
2
3
4
6
8
9
10
Fim
    
```

$\left\{ \begin{array}{l} \text{mult} \% i == 0 \\ \text{mult} \% j == 0 \end{array} \right.$

print(f"{_}")

$\text{mult} \rightarrow$ $\text{mult} \leftarrow \text{mult} + 1$

Saida 0 2 3 4

mult = mult + 1

engarrafado

#verifique se mult é múltiplo de i ou j
 # mult dá um passo a frente

$\sqrt{-1} \rightarrow \mathbb{R}$

7.2.2 Rascunho

multi
5

multi
3



3
2

while $k < n$:



1000000
1000000

0 1000000 1000000

n
8

i
5

j
3

multi
5

multi
3



ideia



spudo
progru

9
6
5
3
20
17

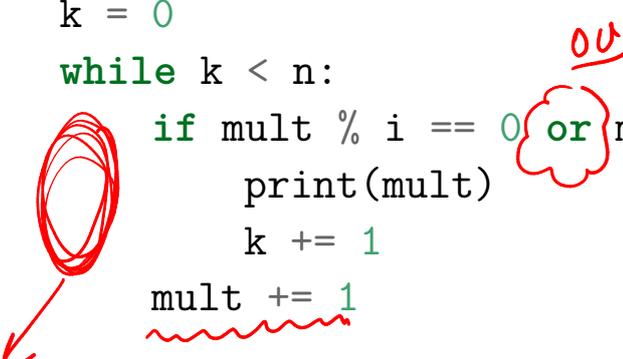
0 3 5 6 9 10

7.2.3 Solução *direta*

Esta solução é uma oportunidade para trabalharmos **and** e **or** e precedência de operadores. Podemos também apresentar *not*. Vixe, muita coisa.

```
n = int(input("Digite n: "));
i = int(input("Digite i: "));
j = int(input("Digite j: "));

mult = 0
k = 0
while k < n:
    if mult % i == 0 ou mult % j == 0:
        print(mult)
        k += 1
    mult += 1
```



7.2.4 Outra solução

A próxima solução explora execução em cadeia. Acho que vou deixar abreviaturas para a próxima aula.

```
n = int(input("Digite n: "));  
i = int(input("Digite i: "));  
j = int(input("Digite j: "));
```

```
mult_i = 0  
mult_j = 0
```

```
k = 0  
while k < n:
```

```
    if mult_i == mult_j:
```

```
        print(mult_i)
```

```
        mult_i += i
```

```
        mult_j += j
```

```
    elif mult_i < mult_j:
```

```
        print(mult_i)
```

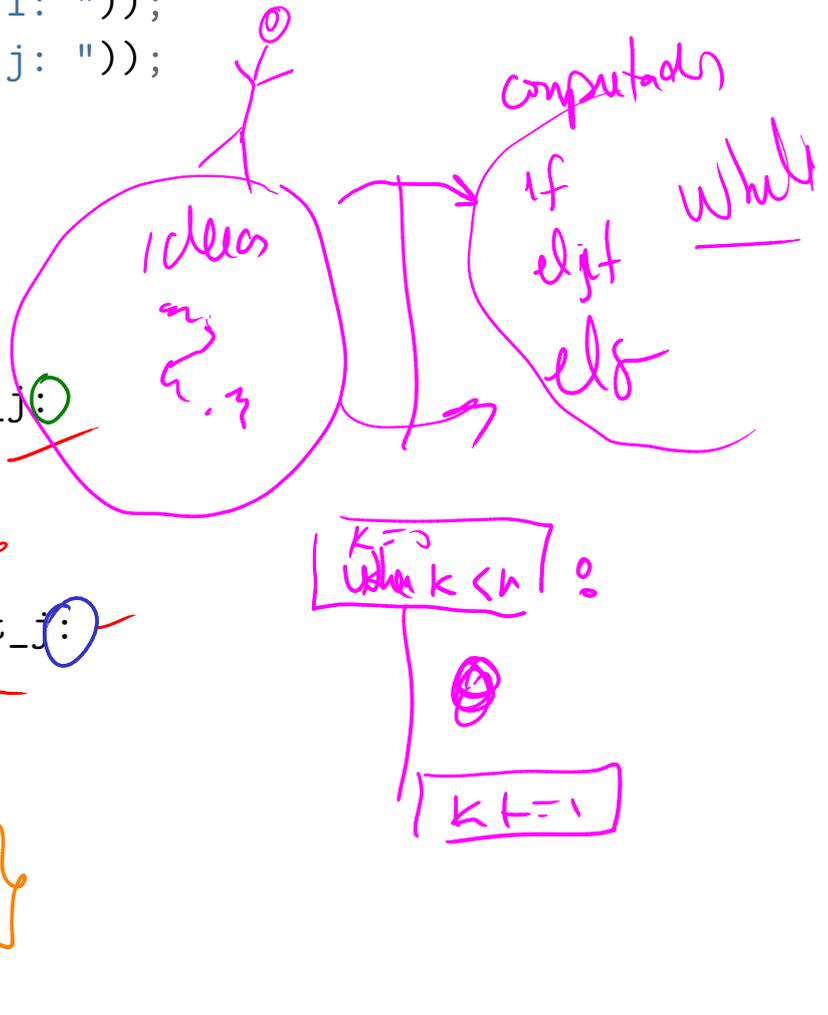
```
        mult_i += i
```

```
    else:
```

```
        print(mult_j)
```

```
        mult_j += j
```

```
    k += 1
```



7.3 Operadores e expressões lógicas

Existem três operadores lógicos: and, or e not

O resultados de condições é **True** ou **False**

Operadores or e and

Prioridades:

$4 + 3 * 2$
 $4 + 6 = 10$

()
**
* /
+ -
< > <= >=
!= ==
not
and
or

$1 < 2$ True
 $1 > 5$ False
 $1 > 2$ bool True False
 $mult \% 1 == 0$

aritmeticos
 + - * /
 // %
 relacionar
 < > =
 bool True False
 $n > 0$ } True
 $i < k$ } False

Prioridade igual: da esquerda para a direita, exceto **

7.3.1 Exemplos

True and True é ?

True and False é ?

True or False é

False or True é

True or False and False é

True or False and False é

7.4 Abreviaturas

$i += 1$

$i += 1$ é equivalente a $i = i + 1$

$+=$ $*=$ $/=$ $-=$ $\%=$

$i += j$ # equivale a $i = i + j$;

$i += (j+k)*d$ # equivale a $i = i + ((j+k)*d)$

$i *= j+k$ # equivale a $i = i*(j+k)$

a op = expressão equivale a $x = x$ op (expressão).

↑
↑
+
- *