20 Reunião 20: 05/NOV/2020

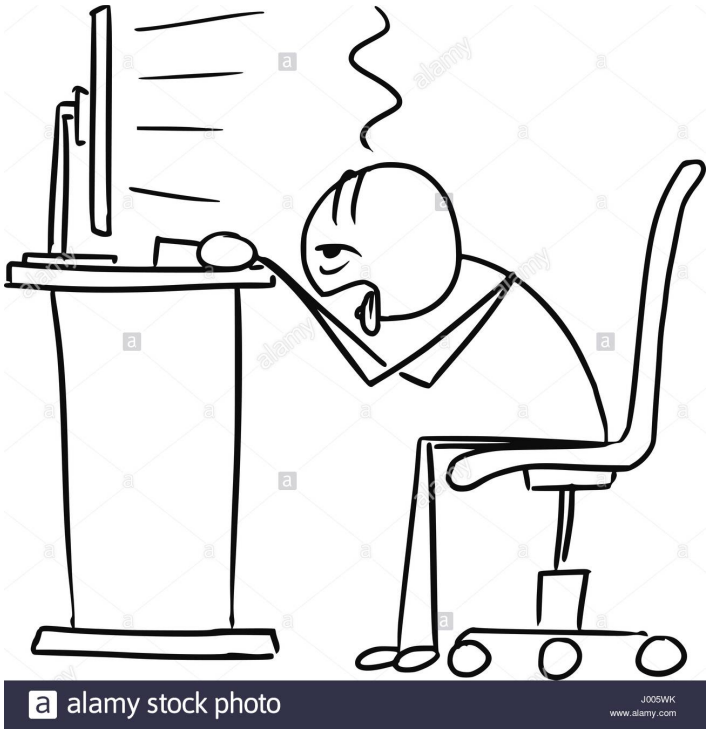


Figure 1: Fonte: https://www.alamy.com/

20.1 Comando `for ... in range()`

```
for i in range(2, 10, 3):
    print(i)
```

tem o mesmo efeito que

```
ini = 2
fim = 10
passo = 3
i = ini
while i < fim:
    print(i)
    i += passo
```

20.2 Fatias

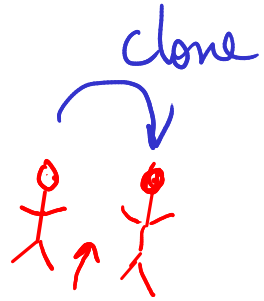
Considere a lista

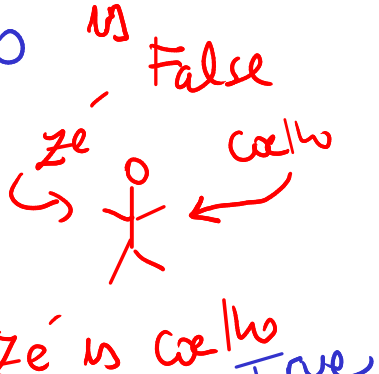
```
lstA = [True, 5, 3.14, None, 'fim']
lstB = lstA
n = len(lstA)
lstC = lstA[0 : n : 1]   # fatia [ ini : fim : passo]
lstD = lstA[0 : n]       # fatia [ ini : fim ]  passo é 1
lstE = lstA[0 : n]       # fatia [ : fim ]  ini é 0 e passo
lstF = lstA[  :  ]       # fatia [ : ]  ini é 0, fim é len(
lstG = lstA[0 : n : 2]   # ...
```

lstG → [True, 3.14, 'fim']

Apelidos × Clones

- Fatias são clones/cópias
- Atribuição só cria apelidos ou modifica





- Listas são objeto/coisa mutáveis: as componentes podem ser alterado (EPOG!)
- Antes de testar uma posição em uma lista tenha certeza que ela existe

20.3 Apelidos

Igualdade de coisas.

`lstA` e `lstB` são *apelidos*, nomes associados a uma mesma lista.

```
if lstA == lstB:
   print("listas têm os mesmos valores, na mesma ordem")
else:
   print("listas tem valores diferentes")
```

Operador `is`

```
if lstA == lstB:
   if lstA is lstB:
       print("Uau, lstA e lstB são a mesma coisa!")
   else:
       print("lstA e lstB têm os mesmos valores, mas não são a
else:
    print("listas tem valores diferentes")
```

20.4 Operador `in list`

O operador `in` pode ser usado para verificar se um item esta na lista

```
item in lst
```

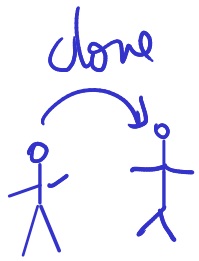
é `True` se `item` é um elemento da lista `lst` e `False` em caso contrário. Por exemplo

```
5 in [2, 3, 5, 7, 11]  == True
True in [True]  == True
True in [False] == False
True in [1, 2, 5, None] == False
None in [1, 2, 5, None] == True
```

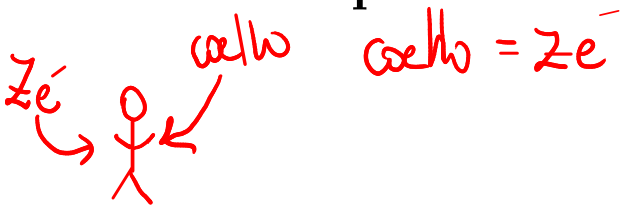
```
if 5 in [3, 2, 7, 5, True]:
    print('5 está na lista')
else:
    print('5 não está na lista')


if False not in [3, 2, 7, 5, True]:
    print('False não está na lista')
else:
    print('False está na lista')
```

20.5 Hoje



- Mutabilidade
- Listas são bichos *mutáveis*
- Fatias são **clones**
- atribuições só **criam apelidos** ou **modificam apelidos**
- atribuições **não** criam clones



20.6 Exercício: crivo de Eratóstenes

Escreva um programa que leia um número natural `n` e imprima todos os primos menores ou iguais a `n`.

20.6.1 Exemplos

```
Programa que imprime todos os primos menores que ou igual a n
Digite n: 20
Primos: 2 3 5 7 11 13 17 19

Programa que imprime todos os primos menores que ou igual a n
Digite n: 100
Primos: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67
        71 73 79 83 89 97
```

20.6.2 Crivo de Erastótenes

Se você está em busca de inspiração, talvez o que segue ajude.

No século três A.C., o astrônomo grego Eratóstenes desenvolveu um algoritmo para determinar todos os números primos até um dado número inteiro positivo `n`. Para aplicar o algoritmo, inicialmente, escrevemos a lista dos inteiros entre 2 e `n`. Por exemplo, se `n` fosse 20 teríamos a lista

```
2 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
```

Em seguida, selecione o primeiro número da lista indicando que encontramos um primo. Agora, percorremos a lista riscando todos os múltiplos do número selecionado, já que nenhum deles é primo.

Após executarmos o primeiro passo do algoritmo teríamos selecionado 2 e riscado todos os múltiplos de 2:

```
2 3  X  5  X  7  X  9  XX  11  XX  13  XX  15  XX  17  XX  19  XX
```

Agora simplesmente repetimos o processo selecionando o primeiro número da lista que não tenha sido selecionado e que não tenha sido riscado. No caso selecionamos e riscamos todos os múltiplos desse número. No exemplo, selecionamos 3 e riscamos os seus múltiplos obtendo:

```
2 3  X  5  X  7  X  X  XX  11  XX  13  XX  XX  XX  17  XX  19  XX
```

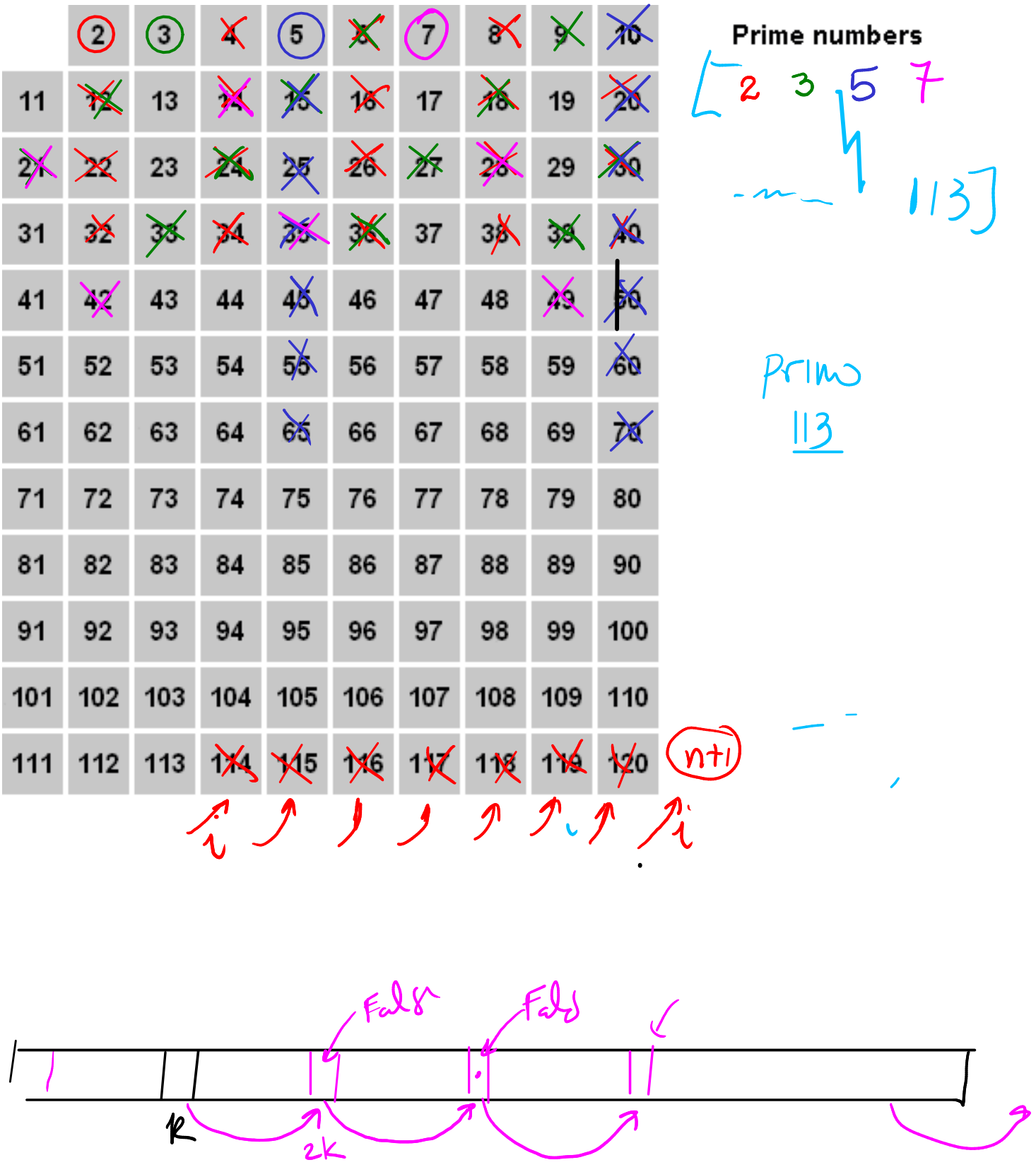
Repetindo esse processo até que todo número na lista tenha sido selecionado ou tenha sido riscado chegamos a:

```
2 3  X  5  X  7  X  X  XX  11  XX  13  XX  XX  XX  17  XX  19  XX
```

Os números que não foram riscados são primos e os demais são compostos.

Esse algoritmo para gerar essa lista de primos é chamado de **Crivo de Eratóstenes**.

Veja a seguir uma animação desse algoritmo copiada da página Crivo de Eratóstenes na Wikipédia.

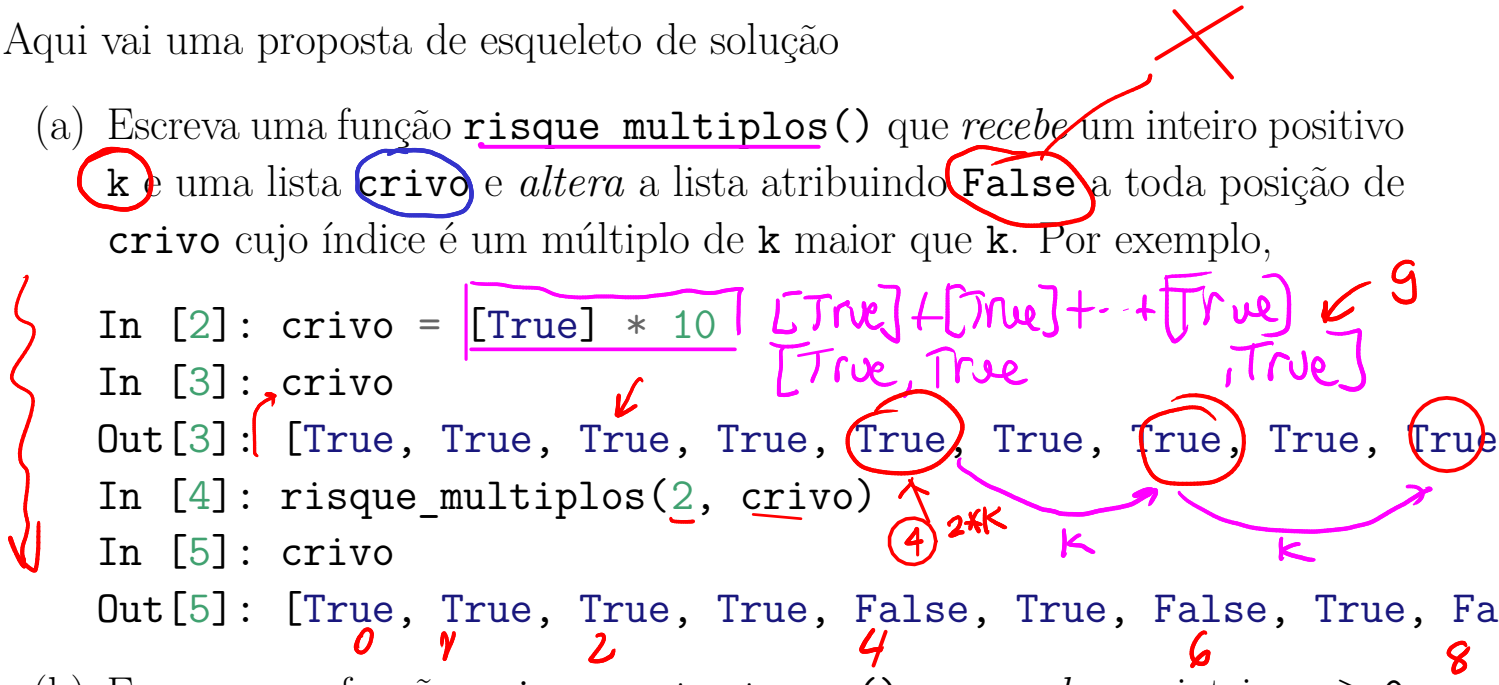

Prime numbers
2 3 5 7
113
prime
113
n+1

False
False
k
2k

20.6.3 Esqueleto de solução

Aqui vai uma proposta de esqueleto de solução

(a) Escreva uma função `risque multiplos()` que *recebe* um inteiro positivo `k` e uma lista `crivo` e *altera* a lista atribuindo `False` a toda posição de `crivo` cujo índice é um múltiplo de `k` maior que `k`. Por exemplo,



```
In [2]: crivo = [True] * 10
In [3]: crivo
Out[3]: [True, True, True, True, True, True, True, True, True
In [4]: risque_multiplos(2, crivo)
In [5]: crivo
Out[5]: [True, True, True, True, False, True, False, True, Fa
```

(b) Escreva uma função `crivo_eratostenes()` que *recebe* um inteiro `n` $\geq$ `0` e *retorna* uma lista com todos os números primos até `n` inclusive. Os primos na lista devem estar em ordem crescente. Por exemplo,

```
In [6]: crivo_eratostenes(29)
Out[6]: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
In [7]: crivo_eratostenes(20)
Out[7]: [2, 3, 5, 7, 11, 13, 17, 19]
```

(c) Escreva uma função `main()` que *lê* um inteiro `n` $\geq$ `0` e *imprime* todos os números primos números primos até `n` inclusive.

20.6.4 Solução

```
def main():
    '''
    Programa que lê um número natural n e imprime todos os
    primos menores ou igual a n.

    O programa é uma implementação do Crivo de Eratóstenes.

    Está função main() serve para testar suas funções.

    Utilize e altere como desejar.

    Exemplos:

    >>> main()
    Programa que imprime todos os primos menores que ou igual a
    Digite n: 120
    Primos: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67
    >>> main()
    Programa que imprime todos os primos menores que ou igual a
    Digite n: 20
    Primos: 2 3 5 7 11 13 17 19
    >>>
    '''
    print("Programa que imprime todos os primos menores que ou ig
    n = int(input("Digite n: "))
    print("Primos: ", end="")
    primos = crivo_eratostenes(n)
    for i in range(len(primos)):
        print(f"{primos[i] }", end="") # end="" para não mudar de
    print()
```

```
#-----------------------------------------------------------------
def risque_multiplos(k, crivo):
    '''(int, list) -> None

    RECEBE um inteiro positivo `k` e uma lista `crivo`.
    ALTERA a lista  atribuindo False a toda posição de `crivo`
    cujo índice é um múltiplo de k maior que k.

    Em palavras, essa função risca os múltiplos de k no crivo.

    Exemplos:

    >>> crivo = [True] * 10
    >>> crivo
    [True, True, True, True, True, True, True, True, True, True]
    >>> risque_multiplos(2, crivo)
    >>> crivo
    [True, True, True, True, False, True, False, True, False, Tr
    >>> crivo = [True] * 12
    >>> risque_multiplos(4, crivo)
    >>> crivo
    [True, True, True, True, True, True, True, True, False, True
    >>> crivo = [1] * 12
    >>> crivo
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
    >>> risque_multiplos(1, crivo)
    >>> crivo
    [1, 1, False, False, False, False, False, False, False, Fals
    >>> crivo = [0] * 16
    >>> crivo
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    >>> risque_multiplos(4, crivo)
    >>> crivo
```

```
    [0, 0, 0, 0, 0, 0, 0, 0, False, 0, 0, 0, False, 0, 0, 0]
    >>> risque_multiplos(5, crivo)
    >>> crivo
    [0, 0, 0, 0, 0, 0, 0, 0, False, 0, False, 0, False, 0, 0, Fa
    >>>
    '''
    n = len(crivo)
    i = k+k
    while i < n:
        crivo[i] = False
        i += k


#-----------------------------------------------------------------
def crivo_eratostenes(n):
    '''(int) -> list

    RECEBE um inteiro `n` >= 0.
    RETORNA uma lista com todos os números primos até `n` inclus

    Os primos na lista devem estar em ordem crescente.

    Esta função deve ser uma implementação do Crivo de Eratósten
    deve utilizar obrigatoriamente a função risque_multiplos().

    Exemplos: atenção, return não é print()

    >>> crivo_eratostenes(10)
    [2, 3, 5, 7]
    >>> crivo_eratostenes(120)
    [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,
    >>> crivo_eratostenes(5)
    [2, 3, 5]
```

```
    >>>
    '''
    primos = []
    crivo = [True]*(n+1)
    if n > 1:
        crivo[0] = False
        crivo[1] = False
    primo = 2
    while primo <= n:
        # coloque o primo na lista
        primos += [primo]

        # risque os múltiplos do primo
        risque_multiplos(primo, crivo)

        # encontre o próximo primo
        i = primo + 1
        achou_primos = False
        while i < n+1 and not crivo[i]:
            i += 1

        # i é o próximo primo menor ou igual a n ou n+1
        primo = i

    return primos

#----------------------------------------------------------------
if __name__ == "__main__":
    main()
```