Fundamentos da Programação
LEIC/LETI

Exemplos Listas

Aula 11

Alberto Abad, Tagus Park, IST, 2021-22

Exemplo: Verificar IMEI válido

- O IMEI (International Mobile Equipment Identity) é um número de 15 dígitos, geralmente exclusivo, para identificar telefones móveis (marcar *#06# para o obter). O dígito mais à direita é um dígito de verificação ou *checksum*.
- O algoritmo Luhn ou a fórmula Luhn, também conhecido como algoritmo *módulo 10*, é uma fórmula de *checksum* simples usada para validar uma variedade de números de identificação, como números de cartão de crédito, números IMEI, etc.
- O algoritmo de Luhn verifica um número em relação ao seu dígito de verificação. O número deve passar no seguinte teste:
  - A partir do dígito mais à direita, que é o dígito de verificação, e movendo para a esquerda, dobrar o valor de cada segundo dígito. Se o resultado dessa operação de duplicação for maior que 9 (por exemplo, 8 × 2 = 16), adicionar os dígitos do número (por exemplo, 16: 1 + 6 = 7, 18: 1 + 8 = 9) ou, alternativamente, , o mesmo resultado pode ser encontrado ao subtrair 9 (por exemplo, 16: 16 - 9 = 7, 18: 18 - 9 = 9).
  - Calcular a soma de todos os dígitos, incluindo o dígito de verificação
  - Se o módulo 10 total é igual a 0 então o número é **válido** de acordo com a fórmula de Luhn; senão é **não válido**.

Account number	7	9	9	2	7	3	9	8	7	1	**x**
Double every other	7	**18**	9	**4**	7	**6**	9	**16**	7	**2**	**x**
Sum digits	7	**9**	9	4	7	6	9	**7**	7	2	**x**

Listas

Exemplo: Verificar IMEI válido - Proposta solução 1:

- Criar uma lista com os digitos do IMEI, alterar os elementos e fazer a soma

In [105]:
```python
def is_valid_imei(imei):
    """
    checks if an imei is valid.
    """

    if (not isinstance(imei, str)) or len(imei) != 15:
        raise ValueError("Doesn't look like a serial number!")

    imei = list(imei)
    for i in range(len(imei)):
        imei[i] = int(imei[i])

    for i in range(len(imei) - 2, -1, -2):
        imei[i] = imei[i] * 2

    # print(imei)

    for i in range(len(imei)):
        if imei[i] >= 10:
            imei[i] -= 9

    # print(imei)

    acc = 0
    for i in range(len(imei)):
        acc = acc + imei[i]

    return acc % 10 == 0

is_valid_imei("353270079684223")
```

Out[105]: True

Listas

Exemplo: Verificar IMEI válido - Proposta solução 2:

- Criar uma lista e obter a soma sem alterar os elementos

In [106]:
```
#Yet another solution.

def is_valid_imei2(imei):
    """
    checks if an imei is valid.

    """

    if (not isinstance(imei, str)) or len(imei) != 15:
        raise ValueError("Doesn't look like a serial number!")

    imei = list(imei)
    acc = 0
    factor = 1
    for i in range(len(imei) - 1, -1, -1):
        double = int(imei[i]) * factor
        acc += ((double - 9) if double > 9 else double)
        factor = 1 if factor == 2 else 2


    return acc % 10 == 0

is_valid_imei2("353270079684223")
```

Out[106]: True

Listas

Exemplo - Crivo de Eratóstenes

Problema: Dado um número `n`, imprima todos os primos menores ou iguais a `n`. Por exemplo, se `n = 20`, a saída deve ser `2, 3, 5, 7, 11, 13, 17, 19`.

A crivo de Eratóstenes é uma das formas mais eficientes de encontrar todos os primos menores que `n` quando `n` for menor que 10 milhões. Algoritmo:

- Crie uma lista de inteiros consecutivos de 2 a n: `lista = [2, 3, 4, ..., n]`.
- Seleciona o primeiro elemento da lista, `p = 2`.
- Enquanto `p` não for maior que $\sqrt{n}$:
  - (a) removem-se da lista todos os múltiplos de p;
  - (b) passa-se ao número seguinte na lista.
- No final do algoritmo, a lista apenas contém números primos.

Exemplo - Crivo de Eratóstenes, Proposta 1

In [97]:
```
from math import sqrt

def crivo1(n):

    lista = [2] + list(range(3, n+1, 2))

    i = 0
    while lista[i] <= sqrt(n):
        p = lista[i]
        j = i + 1

        while j < len(lista):
            if lista[j] % p == 0:
                del lista[j]
            else:
                j += 1

        i = i + 1

    return lista

%timeit -n 100 crivo1(50)
# Question1: Sera que podemos começar com uma lista menor? Reparem que o único par primo é o 2
# Question2: Porque não podemos utilizar for?
# Question3: Sera que podemos abstrair em uma função (mais eficiente) a eliminção dos múltiplos
```

```
13.5 µs ± 255 ns per loop (mean ± std. dev. of 7 runs, 100 loops each)
```


Exemplo - Crivo de Eratóstenes, Proposta 2

In [99]:
```
def crivo2(n):
    # redizimos a lista inicial à metade tirando os pares
    lista = [2] + list(range(3, n+1, 2))

    i = 0
    limite = sqrt(n)
    while lista[i] <= limite:
        # abstraimos a eliminação de multiplos numa função que cons
iga utilizar for
        elimina_multiplos(lista, i)
        i = i + 1

    return lista


def elimina_multiplos(lista, index):
    p = lista[index]
    fim = len(lista) -1
    for i in range(fim, index, -1):
        if lista[i] % p == 0:
            del lista[i]

crivo2(50)
# Question 1: Sera que podemos manter o tamanho da lista fixo e uti
lizar fors?
```

Out[99]:
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```


Exemplo - Crivo de Eratóstenes, Proposta 3

- Criar uma lista de tamanho n+1 com valores lógicos (inicializados a True) indicando se o inteiro correspondente ao índice é, ou não, um númeror primo

In [100]:
```
def crivo3(n):

    # lista de bools correspondente a inteiros de 0..n indicando se
é ou não primo
    lista = [True]*(n+1)
    lista[0], lista[1] = False, False

    # equivalente ao primeiro while anterior
    for i in range(2, int(sqrt(n)) + 1):
        # colocamos a False as possições multiplas de i
        for j in range(i+i, n+1, i):
            lista[j] = False

    return [i for i in range(n+1) if lista[i]]


crivo3(50)
```

Out[100]:
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```


Exemplo - Crivo de Eratóstenes, Eficiência

In [104]:
```
n = 10000
%time crivo1(n)
print()

%time crivo2(n)
print()

%time crivo3(n)
print()
```

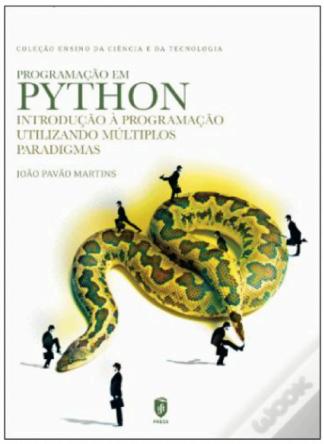
```
CPU times: user 10.3 ms, sys: 646 µs, total: 11 ms
Wall time: 10.5 ms

CPU times: user 6.76 ms, sys: 567 µs, total: 7.32 ms
Wall time: 6.99 ms

CPU times: user 2.86 ms, sys: 62 µs, total: 2.92 ms
Wall time: 2.98 ms
```

Listas - Tarefas próxima aula

- Estudar matéria apresentada até hoje:
  - Fazer todos os programas dos slides



In [ ]: