Programação em Python - Programas simples

Gustavo de Assis Costa

Março de 2013

Problemas com Números

- Múltiplos de 3 ou 5
- Números primos
- Série de Fibonacci
- Fatores primos
- Números palíndromos

Problemas com listas

- Pares e ímpares
- Mediana
- Palavras de uma string
- Elemento mais frequente

Múltiplos de 3 ou 5

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos

3 5 6 9

Faça um programa para listar todos os números naturais menores que 100 que são múltiplos de 3 ou 5.

Múltiplos de 3 ou 5 - Solução

```
# coding: utf-8
#
# Imprime todos os números naturais menores que 100
# que são múltiplos de 3 ou 5
#
for i in range(1,100):
    if i%3 == 0 or i%5==0:
        print i,
```

```
3 5 6 9 10 12 15 18 20 21 24 25 27 30 33 35 36 39 40
42 45 48 50 51 54 55 57 60 63 65 66 69 70 72 75 78 80
81 84 85 87 90 93 95 96 99
```

Números primos

Um número é primo se é divisível apenas por si próprio e a unidade. Por exemplo, os seguintes números são primos:

2 3 5 7 9 11 13 17 19 23 29

Faça um programa que leia um número e diga se é primo ou não.

Primos - Solução

```
# coding: utf-8
#
# Solicita um número inteiro e diz se é primo
#
n = input("Entre com um número natural: ")
primo = True
for i in range(2,n):
    if n%i == 0:
        primo = False
if primo:
    print n, "é primo"
else:
    print n, "não é primo"
```

Primos - Exemplos

```
Entre com um número natural: 97
97 é primo
```

```
Entre com um número natural: 61
61 é primo
```

```
Entre com um número natural: 57
57 não é primo
```

Série de Fibonacci

Em matemática, a série de Fibonacci é constituída pelos números

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...$$

Por definição, os dois primeiros números da sequência são 0 e 1 e os demais são dados pela soma dos dois anteriores:

$$F_0 = 0, F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}, \text{ para } n > 1$$

Escreva um programa para computar a soma dos termos da sequência cujos valores são números pares não maiores do que 4 milhões

Série de Fibonacci - Solução

```
# coding: utf-8
#
# Computa a soma dos termos pares não maiores que
# 4 milhões da sequência de Fibonacci
#
penultimo, ultimo = 0, 1
soma = 0
while ultimo <= 4000000:
    if ultimo%2 == 0:
        soma = soma + ultimo
    penultimo, ultimo = ultimo, penultimo+ultimo
print soma
```

4613732

Fatores Primos

Qualquer número natural pode ser expresso como o produto de seus fatores primos. Por exemplo

$$13195 = 5 \times 7 \times 13 \times 29.$$

Escreva um programa para computar o maior fator primo de 6008511.

Fatores Primos - Solução

```
def eprimo(n):
    """Retorna True se e somente se n é primo"""
    divisor = 2
    while divisor < n:
        if n % divisor == 0: return False
        divisor = divisor + 1
    return True

numero = 6008511
fator = numero-1
while fator>1:
    if numero % fator == 0 and eprimo(fator):
        break
    fator = fator - 1
print fator
```

4613732

Números palíndromos

Um número palíndromo apresenta a mesma sequência de dígitos quando lido da esquerda para a direita ou vice-versa, por exemplo:

```
12321
1005001
99
```

Escreva uma função chamada `palindromo(`n`)` que retorna `True` ou `False` conforme o inteiro n seja ou não palíndromo quando escrito em base 10

Números Palíndromos - Solução

```
from math import log10

def digito(i,n):
    """Retorna o iésimo dígito da representação decimal de n.
    (A casa das unidades corresponde ao 0'ésimo digito)."""
    return n/(10**i)%10

def ndigitos(n):
    """Retorna quantos dígitos tem o número inteiro n."""
    return int(log10(n))+1

def palindromo(n):
    """Retorna True se o número n é palíndromo (em decimal)."""
    k = ndigitos(n)
    for i in range(k/2):
        if digito(i,n)!=digito(k-i-1,n): return False
    return True
```

Números Palíndromos - Exemplo

```
print palindromo(1234)
print palindromo(1234567654321)
```

```
False
True
```

Pares e ímpares

Escreva uma função chamada `par_impar(`l`)` que toma uma lista de inteiros l e retorna 2 listas, uma com os números pares de l e outra com os ímpares.

Pares e ímpares - solução

```
def par_impar(l):
    """Retorna em 2 listas os valores pares e ímpares
    da lista de inteiros l."""
    par = []
    impar = []
    for x in l:
        if x%2 == 0:
            par = par + [x]
        else:
            impar = impar + [x]
    return par,impar

print par_impar([9,23,1,2,4,22,100,17])
```

```
([2, 4, 22, 100], [9, 23, 1, 17])
```

Mediana

Em teoria de probabilidade e estatística, a *mediana* de uma amostra é definida como o valor que separa a metade mais alta da amostra da metade mais baixa. Dada uma lista de números, a mediana pode ser obtida ordenando os elementos e tomando o elemento na posição central. Por exemplo, a mediana de

```
[9,1,2,8,7]
```

é `7`. Faça uma função `mediana(`l`)` que retorne a mediana de uma lista l.

Mediana - solução

```
def menores(l,x):
    "Retorna quantos números na lista l são menores que x"
    n = 0
    for elem in l:
        if elem<x: n = n+1
    return n

def mediana(l):
    "Retorna a mediana dos números na lista l"
    k = len(l)/2
    for x in l:
        if menores(l,x)==k: return x

print mediana([1,2,3,4,5,6,7])
print mediana([9,1,2,8,7])
```

4

7

Palavras de uma string

No processamento de strings, frequentemente queremos analisar um texto palavra por palavra. Escreva uma função `palavras(`txt`)` que retorna em uma lista as palavras de uma string.

OBS.: O operador `in` pode ser usado para testar se uma string está contida em outra. Por exemplo `"ab"in "abc"` é avaliado como `True`, enquanto que `"abd"in "abc"` retorna `False`.

Palavras de uma string - Solução

```
def palavras(texto):
    "Retorna em uma lista as palavras da string texto"
    lista = []
    i = 0
    for j in range(len(texto)):
        if texto[j] in """,.:;/?!@#$%&*()-_+'" \n\r\t""":
            if j>i: lista += [texto[i:j]]
            i = j+1
    if j>i: lista += [texto[i:j+1]]
    return lista
```

Palavras de uma string - Exemplo

```
print palavras("""Em programação e em linguagens
formais, uma cadeia de caracteres é uma sequência
ordenada de caracteres (símbolos) escolhidos
a partir de um conjunto pré-determinado""")
```

```
['Em', 'programa\xc3\xa7\xc3\xa3o', 'e', 'em',
'linguagens', 'formais', 'uma', 'cadeia', 'de',
'caracteres', '\xc3\xa9', 'uma', 'sequ\xc3\xaancia',
'ordenada', 'de', 'caracteres', 's\xc3\xadmbolos',
'escolhidos', 'a', 'partir', 'de', 'um', 'conjunto',
'pr\xc3\xa9', 'determinado']
```

Elemento mais frequente

Escreva uma função chamada `maisfrequente(`l`)` que retorna o elemento que aparece mais vezes na lista l. Por exemplo,

```
maisfrequente([1,2,3,2,3,3,1])
```

deve retornar 3

Elemento mais frequente - Solução

```
def maisfrequente(l):
    "Retorna o numero mais frequente da lista l"
    visto = [] # lista de pares [elemento,frequencia]
    for elem in l:
        achei = False
        for i in range(len(visto)):
            if elem == visto [i][0]:
                achei = True
                visto[i][1] = visto[i][1]+1
                break
        if not achei: visto = visto + [[elem,1]]
    j = 0 # Posicao do elemento mais frequente
    for i in range(len(visto)):
       if visto[i][1] > visto[j][1]: j = i
    return visto[j][0]
```