Caderno de Exercícios
Processamento da Informação – Edição Python

Jesús P. Mena-Chalco

Centro de Matemática, Computação e Cognição - CMCC
Universidade Federal do ABC
jesus.mena@ufabc.edu.br
http://professor.ufabc.edu.br/~jesus.mena/

Atualizado em 13 de julho de 2013

A reprodução total ou parcial do conteúdo desta publicação é permitida desde que seja citada a fonte e a finalidade não seja comercial. Os créditos deverão ser atribuídos aos respectivos autores.

Licença Creative Commons License Deed

Atribuição-Uso Não-Comercial Compartilhamento pela mesma Licença 2.5 Brasil

Você pode: copiar, distribuir, exibir e executar a obra; criar obras derivadas. Sob as seguintes condições: atribuição - você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante; uso não comercial - você não pode utilizar esta obra com finalidades comerciais; compartilhamento pela mesma licença: se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta. Para cada novo uso ou distribuição, você deve deixar claro para outros os termos da licença desta obra. Qualquer uma destas condições pode ser renunciada, desde que você obtenha permissão do autor. Nada nesta licença restringe os direitos morais do autor. Creative Commons License Deed - http://creativecommons.org/licenses/by-nc-sa/3.0/deed.pt.

Este caderno de exercícios foi editado com `vim` 7.3 e preparado com LaTeX.

Alguns exercícios podem conter erros. Caso os identifique, por favor, envie a sugestão de melhora ou correções ao email jesus.mena@ufabc.edu.br. O prêmio dado por erro enontrado ainda não foi fixado.

Sumário

1 Funções **5**

2 Desvio condicional **6**

3 Laços **8**

4 Laços aninhados **16**

5 Strings **21**

6 Listas **28**

7 Matrizes **33**

8 Recursividade **39**

9 Algoritmos de ordenação **41**

10 Conjuntos (através de listas) **45**

11 Busca de dados **45**

12 Prova Teoria 01: 2013-Q1 **46**

13 Prova Teoria 02: 2013-Q1 **52**

14 Prova Pratica 01: 2013-Q1 **56**

15 Prova Pratica 02: 2013-Q1 **59**

1 Funções

1. Crie uma função `potencia` que receba dois números a e b (base e expoente, respectivamente) e retorne a^b.

```
def potencia(base, expoente):
    resposta = base**expoente
    return resposta
```

2. Crie uma função que permita a conversão de graus Celsius para Fahrenheit.

```
def celsius2fahrenheit(graus):
    return 9/5.0*graus+32
```

3. Crie uma função `numero_par` que permita verificar um dado número, x, passado como parâmetro é número par.

```
def numero_par(x):
    if x%2==0:
        return True
    else:
        return False
```

4. Dadas as seguintes funções:

```
def equacao1(p, q):
    r1 = p+q
    r2 = p-q
    return r1*r2

def equacao2(r, s):
    return r**2 - s**2
```

Determine os valores para as seguintes operações:

- equacao1(3,4) : -7
- equacao1(4,3) : 7
- 2**equacao2(2,0) : 16
- equacao1(0,2) + equacao2(0,4) : -20
- equacao1(9,99)-equacao2(9,99) : 0

5. Dadas as seguintes funções:

```
def numero_par(x):
    if x%2==0:
        return True
    else:
        return False

def funcaoX(a, b):
    if numero_par(a):
        return a-b
    else:
        return a-2*b
```

Determine os valores para as seguintes operações:

- Determine os valores para as seguintes operações:
- funcaoX(0,20) : -20
- funcaoX(20,3) : 17
- funcaoX(3,20) : -37
- numero_par(1)+numero_par(2) : 1
- numero_par(4)*funcaoX(1, funcaoX(2,3)) : 3

2 Desvio condicional

1. Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
if x>=8.5:
    print "Conceito A"
if x>=7.5:
    print "Conceito B"
if x>=5.5:
    print "Conceito C"
if x>=5:
    print "Conceito D"
```

Resposta:

```
Conceito B
Conceito C
Conceito D
```

2. Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
if x>=8.5:
    print "Conceito A"
elif x>=7.5:
    print "Conceito B"
elif x>=5.5:
    print "Conceito C"
elif x>=5:
    print "Conceito D"
```

Resposta:

```
Conceito B
```

3. Indique a mensagem que apresentará a execução das seguintes instruções:

```
aluno = "Joao Carlo"
disciplina = "PI"
if aluno=="Joao Carlos" and disciplina=="PI":
    print "Conceito A"
else:
    print "Aluno nao cadastrado"
```

Resposta:

```
Aluno nao cadastrado
```

4. Indique a mensagem que apresentará a execução das seguintes instruções:

```
aluno = "Joao Carlo"
disciplina = "PI"
if aluno=="Joao Carlos" or disciplina=="PI":
    print "Conceito A"
else:
    print "Aluno nao cadastrado"
```

Resposta:

```
Conceito A
```

5. Indique a mensagem que apresentará a execução das seguintes instruções:

```
x=8
y=5
z=13
if x>=1 and x<=31:
    if y>0 and y<13:
        if x+y!=z:
            print "A data de hoje eh 8/5/13"
        else:
            print "A data de hoje nao eh 8/5/13"
```

Resposta:

```
A data de hoje nao eh 8/5/13
```

6. Crie uma função em que, dados 3 números como parâmetros, permita verificar se a soma de quaisquer par de números gera a soma do terceiro número.

```
def verificar_somatoria(a,c,b):
    if a+b==c:
        return str(a)+'+'+str(b)+'='+str(c)
    if a+c==b:
        return str(a)+'+'+str(c)+'='+str(b)
    if b+c==a:
        return str(b)+'+'+str(c)+'='+str(a)
```

7. Crie uma função `determinar_o_maior_numero` que receba dois números (inteiros ou reais) e retorne o maior valor de ambos os números.

```
def determinar_o_maior_numero(p, q):
    if p<q:
        return q
    else:
        return p
```

8. Crie uma função `determinar_o_maior_numero` que receba três números (inteiros ou reais) e retorne o maior valor do números.

```
def determinar_o_maior_numero(a, b, c):
    if a<b:
        if b<c:
           return c
        else:
           return b
    else:
        if a<c:
           return c
        else:
           return a
```

3 Laços

1. Simule a execução da função abaixo e indique (algebricamente) a sua saída. Considere valores de b diferentes de zero, e $p < q$.

```
def funcaoEnigma1(a,b,p,q):
    a = float(a)
    b = float(b)
    p = int(p)
    q = int(q)
    i = 0
    soma = 0
    while i<=(q-p):
        soma = (a/b)**(i)
        i = i+1
    return soma
```

Resposta:

$$\left(\frac{a}{b}\right)^{q-p}$$

2. Simule a execução da função abaixo e indique (algebricamente) a sua saída. Considere valores de b diferentes de zero, e $p < q$.

```
def funcaoEnigma2(a,b,p,q):
    a = float(a)
    b = float(b)
    p = int(p)
    q = int(q)
    i = 0
    soma = 0
    while i<=(q-p):
        soma = soma + (a/b)**(i)
        i = i+1
    return soma
```

Resposta:

$$\sum_{i=0}^{q-p} \left(\frac{a}{b}\right)^i$$

3. Simule a execução da função abaixo e indique (algebricamente) a sua saída. Considere valores de $n > 0$

```
def funcaoEnigma3(n):
    soma = 0
    for i in range(1,n+1):
        soma = soma+i
    j=1
    while j<=n:
        soma = soma-j
        j = j+1
    return soma
```

Resposta: 0

4. Simule a execução da função abaixo e indique (algebricamente) a sua saída. Considere valores de $n > 0$

```
def funcaoEnigma4(n):
    mult = 1.0
    i = 0
    while i<=n-2:
        mult = mult*((i+1.0)/(i+2.0))
        i = i+1
    return mult*n
```

Resposta: 1

5. Crie uma função que permita imprimir a palavra SPAM, n vezes.

```
# Primeira versao usando o laco for
def imprimir_spam(n):
    for i in range(1,n+1):
        print "SPAM"
```

```
# Segunda versao usando o laco while
def imprimir_spam(n):
    i = 1
    while i <= n:
        print "SPAM"
        i = i+1
```

6. Crie uma função que permita imprimir os primeiros n números naturais.

```
# Primeira versao usando o laco for
def imprimir_sequencia(n):
    for i in range(1,n+1):
        print i
```

```
# Segunda versao usando o laco while
def imprimir_sequencia(n):
    i = 1
    while i <= n:
        print i
        i = i+1
```

7. Crie uma função que permita mostrar uma sequência de números ímpares de 1 até n.

```
def sequencia_impares(n):
    i = 1
    while i <= n:
        print i
        i = i+2
```

8. Crie uma função que permita mostrar a sequência de números inteiros, no intervalo $[x, y]$. Considere $x < y$.

```
def imprimir_intervalo(x,y):
    for i in range(x,y+1):
        print i
```

9. Crie uma função que permita somar a sequência de números inteiros, no intervalo $[x, y]$. Considere $x < y$:

```
def somar_intervalo(x,y):
    soma = 0
    for i in range(x,y+1):
        soma = soma+i
    return soma
```

10. Crie uma função que permita o calculo da seguinte somatória:

$$-1 + 2 - 3 + 4 - 5 + 6 + \cdots + n$$

```
def somaQ(n):
    soma = 0
    for i in range(1,n+1):
        soma = soma + i*( (-1)**i)
    return soma
```

11. Crie uma função `somaP`, em que dado um inteiro $n > 0$, permita somar a seguinte sequência:

$$1^2 + 2^2 + \cdots + n^2$$

```
def somaP(n):
    soma = 0
    for i in range(1,n+1):
        soma = soma + i**2
    return soma
```

12. Modifique a função `somaP` em que, além do número n, seja utilizado um outro número k, de tal forma que o calculo da seguinte somatória seja realizada:

$$1^k + 2^k + \cdots + n^k$$

```
def somaP(n, k):
    soma = 0
    for i in range(1,n+1):
        soma = soma + i**k
    return soma
```

13. Modifique a função `somaP` em que, além do número n, seja utilizado um outro número k, de tal forma que o calculo da seguinte somatória seja realizada:

$$k^1 + k^2 + \cdots + k^n$$

```
def somaP(n, k):
    soma = 0
    for i in range(1,n+1):
        soma = soma + k**i
    return soma
```

14. Crie uma função em que, dado um inteiro não-negativo n, seja possível determinar $n!$.

```
# Primeira versao usando o laco for
def fatorial(n):
    mult = 1
    for p in range(1,n+1):
        mult = mult*p
    return mult
```

```
# Segunda versao usando o laco while
def fatorial(n):
    mult = 1
    p = 1
    while p<=n:
        mult = mult*p
        p = p+1
    return mult
```

15. Dizemos que um número natural é triangular se ele é produto de três número naturais consecutivos. Por exemplo: 120 é triangular, pois 4*5*6 = 120. 2730 é triangular, pois 13*14*15 = 2730. Dado um inteiro não-negativo n, crie uma função para verificar se n é triangular. Deve-se devolver True se o número for triangular, caso contrário False.

```
# Primeira versao usando o laco for
def numeroTriangular(n):
    for i in range(3,n+1):
        if (i-2)*(i-1)*(i)==n:
            return True
    return False
```

```
# Segunda versao usando o laco while
def numeroTriangular(n):
    i=3
    while i<=n:
        if (i-2)*(i-1)*(i)==n:
            return True
        i=i+1
    return False
```

16. Dado um inteiro positivo p, crie uma função para verificar se p é primo.

```
# Solucao 1
def primo(p):
    contador = 0
    i = 1
    while i<=p:
        if p%i==0:
            contador = contador+1
        i = i+1
    if contador==2:
        return True
    else:
        return False
```

```
# Solucao 2
def primo(p):
    contador = 0
    for i in range(1,p+1):
        if p%i==0:
            contador = contador+1
    if contador==2:
        return True
    else:
        return False
```

17. Escreva uma função que receba um inteiro não-negativo n e imprima a soma dos n primeiros números primos. Por exemplo: Para n=3, resultado 10=2+3+5. Para n=7, resultado 58=2+3+5+7+11+13+17. Para n=100, resultado 24133.

```
def soma_primos(n):
    i = 1
    soma = 0
    numero_primos = 0
    while numero_primos<n:
        if primo(i):
            soma = soma + i
            numero_primos = numero_primos+1
        i = i+1
    return soma
```

18. Faça uma função que calcula a soma:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{1}{9999} - \frac{1}{10000}$$

Pelas seguintes maneiras:

- Adição de termos da esquerda para a direita.

```
def equacaoE():
    soma = 0
    i=1
    while i<=10000:
        soma = soma + (1.0/i)*((-1)**(i+1))
        i=i+1
    return soma
```

- Adição de termos da direita para a esquerda.

```
def equacaoE():
    soma = 0
    i=10000
    while i>=1:
        soma = soma + (1.0/i)*((-1)**(i+1))
        i=i-1
    return soma
```

- Adição separada dos termos positivos e dos termos negativos da esquerda para a direita.

```
def equacaoE():
    somaPos = 0
    i = 1
    while i<=10000:
        somaPos = somaPos + 1.0/i
        i = i+2
    somaNeg = 0
    i = 2
    while i<=10000:
        somaNeg = somaNeg + 1.0/i
        i = i+2
    return somaPos-somaNeg
```

19. Crie uma função que permita somar apenas os números impares da sequência de inteiros contida no intervalos $[x, y]$, para $x < y$.

```
def soma_impares(x,y):
    soma = 0
    for i in range(x,y+1):
        if i%2==1:
            soma = soma+i
    return soma
```

20. Dado um inteiro positivo n, crie uma função para calcular a seguinte soma:

$$\frac{1}{n} + \frac{2}{n-1} + \frac{3}{n-2} + \ldots \frac{2}{n-1} + \frac{n}{1}$$

```
def soma(n):
    soma = 0
    for i in range(1,n+1):
        soma = soma + float(i)/(n-(i-1))
    return soma
```

21. Crie uma função `arctan` que recebe o número real $x \in [0, 1]$ e devolve uma aproximação do arco tangente de x (em radianos) através da série:

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

Considere somente os 100 primeiros termos da série.

```
def arctan(x):
    x = float(x)
    soma = 0
    sinal = -1
    for i in range(1,100+1):
        coef  = 2*i-1
        sinal = sinal*-1
        soma  = soma + sinal * (x**coef)/coef
    return soma
```

22. Crie uma função `arctan2` que recebe o número real $x \in [0, 1]$ e devolve uma aproximação do arco tangente de x (em radianos) através da série:

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

Considere todos os termos da serie até que:

$$\left|\frac{x^k}{k}\right| < 0.0001$$

```
def arctan2(x):
    x = float(x)
    soma  = 0
    sinal = -1
    i = 1
    while True:
        coef  = 2*i-1
        sinal = sinal*-1
        termo = (x**coef)/coef
        soma  = soma + sinal * termo
        if abs(termo)<0.0001:
            break
        i=i+1
    return soma
```

23. Escreva uma função `encaixa` que, recebendo dois números inteiros a e b como parâmetros, verifica se b corresponde a os últimos dígitos de a. Por exemplo 45 encaixa em 12345, 2026 encaixa em 2026, 12345 não encaixa em 45.

```
def encaixa(a, b):
    while True:
        if a%10==b%10:
            a = a/10
            b = b/10
            if b==0:
                return "encaixa"
        else:
            return "nao encaixa"
```

24. Problema 3n+1 (Conjectura Collatz): As vezes nem sempre é fácil determinar se o laço irá terminar. Um exemplo:

```
def sequencia3n1(n):
    while n != 1:
        print n
        if n%2 == 0:
            n = n/2
        else:
            n = n*3+1
    print 'Finalizou'
```

Para $n = 27$, quantas vezes o laço é executado? Qual é o maior valor atingido?

Resposta: O número de vezes é 111 e o maior valor obtido é 9232.

Modifique a função anterior para apresentar o número de vezes que o laço é executado, o maior valor atingido, e a somatoria de todos os elementos considerados na sequência.

```
def sequencia3n1(n):
    vezes=0
    maior=0
    soma=0
    while n != 1:
        vezes=vezes+1
        if n>maior:
            maior = n
        soma = soma+n
        print n
        if n%2 == 0:
            n = n/2
        else:
           n = n*3+1
    print 'Finalizou'
    print 'Numero de vezes: ' + str(vezes)
    print 'Maior valor atingido: ' + str(maior)
    print 'Somatoria de todos os elementos: ' + str(soma)
```

25. Crie uma função que permita imprimir os n primeiros números da sequência de Fibonacci. Considere $n \geq 2$ e apenas um laço 'while'.

```
def fibonacci(n):
    k  = 2
    t1 = 0
    t2 = 1
    print t1
    print t2
    while k<=n:
        seguinte = t1+t2
        print seguinte
        t1 = t2
        t2 = seguinte
        k  = k+1
```

26. Dado um número natural n na base decimal, crie uma função que permita transformá-lo para a base b.

```
def converter(n, b):
    resposta = ''
    while n>0:
        r = n%b
        n = n/b
        resposta = str(r)+resposta
    return int(resposta)
```

4 Laços aninhados

1. Crie uma função para imprimir a tabuada de a até b. Considere $a \leq b$.

```
def tabuada(a,b):
   for i in range(a,b+1):
       print "\nTabuada: "+str(i)
       for j in range(1,11):
           print str(i)+"x"+str(j)+"="+str(i*j)
```

2. Dada a seguinte função:

```
def funcao(n):
    soma = 0
    for i in range(1,n+1):
        for j in range(1,n+1):
            for k in range(1,n+1):
                soma = soma + i
    return soma
```

Considere como parâmetro de entrada um número inteiro n positivo.

- Indique algebricamente a somatória que a função realiza.
  Resposta:
$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} i = n^2 \sum_{i=1}^{n} i = n^2 \frac{n(n+1)}{2}$$
- Qual é o resultado para o chamado a função com o parâmetro n=10.
  Resposta: 5500

3. Dados os números reais x e $\epsilon > 0$, calcular a aproximação para e^x usando a seguinte expansão da Série de Taylor:

$$1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \ldots$$

Incluindo todos os termos da série até que:

$$\left|\frac{x^k}{k!}\right| < \epsilon$$

```
def exp2(x, epsilon):
    x = float(x)
    soma  = 1
    i = 1
    while True:
        fact=1
        for i in range(1,i+1):
            fact = fact*i
        termo = (x**i)/fact
        soma  = soma + termo
        if abs(termo)<epsilon:
            break
        i=i+1
    return soma
```

4. Dada a seguinte função:

```
def funcaoEnigma1(p, q):
    soma = 0
    for i in range(p,q+1):
        for j in range(1,i+1):
            soma = soma + i
        if soma==0:
            break
    return soma
```

- Indique algebricamente (de forma concisa) a somatória que a seguinte função realiza. Resposta:

$$= \sum_{i=p}^{q} \sum_{j=1}^{i} i = \sum_{i=p}^{q} i^2$$

- Qual é o resultado para o chamado a função com os parâmetros p=3, e q=4. Resposta: 25

5. Use a seguinte série de Taylar para determinar o valor de π:

$$\frac{\pi^2}{8} = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \frac{1}{9^2} + \ldots$$

Considere a somatória até que o k-ésimo termo seja menor do que um valor epsilon (ϵ), por exemplo, $\epsilon = 1e - 10$)

```
def pi_taylor():
    soma = 0
    i = 1
    while 1.0/(i**2) >= 1e-10:
        soma = soma + 1.0/(i**2)
        i = i+2
    return sqrt(soma*8)
```

6. O matemático Srinivasa Ramanujan encontrou uma série infinita, que pode ser usada para gerar uma aproximação numérica de pi:

$$\frac{1}{\pi} = \sum_{k=0}^{\infty} \frac{(4k)!(1103 + 26390k)}{(k!)^4 396^{4k}}$$

Escreva uma função chamada `estimar_pi` que usa esta fórmula para calcular e retornar uma estimativa de π. Deve usar um laço while para calcular os termos de soma até o último termo ser menor que 1e-15 (que é a notação Python para 1×10^{-15}).

```python
# Solucao 1
def estimar_pi():
    soma  = 0.0
    termo = 1.0
    k = 0
    while termo>=1e-15:
        fact = 1
        for i in range(1,4*k+1):
            fact = fact*i
        fact2 = 1
        for i in range(1,k+1):
            fact2 = fact2*i
        termo = (fact*(1103+26390*k))/((fact2**4)*396**(4*k))
        soma  = soma+termo
        k = k+1
    return (1/((2*sqrt(2)/9801)*soma))
```

```python
# Solucao 2
def factorial(n):
    f = 1
    for i in range(2,n+1):
        f = f*i
    return f

def estimar_pi():
    soma  = 0.0
    termo = 1.0
    k = 0
    while termo>=1e-15:
        termo = (factorial(4*k)*(1103+26390*k))/((factorial(k)**4)*396**(4*k))
        soma  = soma+termo
        k = k+1
    return (1/((2*sqrt(2)/9801)*soma))
```

7. Dado um inteiro positivo n, imprimir as n primeiras linhas do triângulo de Pascal.

```
# Solucao 1
def criar_matriz_zeros(l,c):
normalizar_matriz(A).
    matriz = [0]*l
    for i in range(0,l):
        matriz[i] = [0]*c
    return matriz

def pascal1(n):
    M=criar_matriz_zeros(n,n)
    for i in range (0,n):
        M[i][i]=1
        M[i][0]=1
        for j in range (0,n):
            if i!=j and i>j:
                termo=M[i-1][j]+M[i-1][j-1]
                M[i][j]=termo
    for i in range (0,len(M)):
        print M[i][0:i+1]
```

```
# Solucao 2
def criar_matriz_uns(l,c):
    matriz = [1]*l
    for i in range(0,l):
        matriz[i] = [1]*c
    return matriz

def pascal2(n):
    M = criar_matriz_uns (n, n)
    for i in range (0, len(M)):
        for j in range (0, len(M[0])):
            if i<j:
                M[i][j] = 0
    k = 0
    while k < n:
        i = k
        for j in range (1,i):
            M[i][j] = M[i-1][j-1] + M[i-1][j]
        print M[i][0:i+1]
        k = k+1
```

```
# Solucao 3
def criar_lista_zeros(n):
    lista = [0]*(n**2)
    return lista

def pascal3(n):
    i = 0
    j = 0
    A = criar_lista_zeros(n)
    for i in range(0,n):
        for j in range(0,i+1):
            if j==0 or i==j:
                A[n*i+j]=1
            else:
                A[n*i+j]=A[n*(i-1)+(j-1)]+A[n*(i-1)+j]
    for i in range(0,n):
        str1 = ''
        for j in range(0,i+1):
            str1 = str1 + ' ' + str(A[n*i+j])
        print str1
```

8. Dados dois inteiros positivos m e n, determinar, entre todos os pares de números inteiros (x,y) tais que $0 \leq x \leq m$, e $0 \leq y \leq n$, um par para o qual o valor da expressão: $xy - x^2 + y$ seja máximo, e calcular também esse máximo.

```
def maximo(m,n):
    max = 0
    indx = 0
    indy = 0
    for x in range(0,m+1):
        for y in range(0,n+1):
            if max <= x*y-x**2+y:
                max = x*y-x**2+y
                indx = x
                indy = y
    print 'Maximo: '+str(max)
    print 'Par   : '+str(indx)+','+str(indy)
```

5 Strings

1. Indique a mensagem que apresentará a execução da seguinte função. Considere como parâmetro de entrada a string 'abracadabra'

```
def funcao1(frase):
    str1 = ""
    str2 = ""
    k = len(frase)-1
    while k>=0:
        str1 = str1 + frase[k]
        str2 = frase[k] + str2
        k = k-1
    print str1
    print str2
```

Resposta:

```
arbadacarba
abracadabra
```

2. Indique a mensagem que apresentará a execução da seguinte função. Considere como parâmetro de entrada a string 'UFABC' e '123'.

```
def funcao2(palavra):
    str = ''
    k = 0
    while k<len(palavra):
        str = str + palavra[k]
        k = k+1
    while k>0:
        k = k-1
        str = str + palavra[k]
    print str
```

Resposta:
UFABCCBAFU
123321

3. Indique a mensagem que apresentará a execução da seguinte função. Considere como parâmetro de entrada a string 'um dois tres'.

```
def funcao3(frase):
    contador = 0
    k = 0
    while k<len(frase)/2:
        if frase[k]==" ":
            contador = contador+1
    print contador
```

Resposta: A função não apresenta nenhuma resposta. Loop infinito, pois a variável k não é modificada.

4. Indique a mensagem que apresentará a execução da seguinte função. Considere como parâmetro de entrada a string '54321'.

```
def funcao4(frase):
    nova_frase = ''
    k = 0
    while k<len(frase):
        nova_frase = frase[k] + nova_frase + frase[k]
        k = k+1
    print nova_frase
```

Resposta: '1234554321'

5. Crie uma função que permita identificar o índice de um caractere em uma string. Se o caractere não for encontrado, a função deve retornar -1.

```
def find(cadeia, caractere):
    indice = 0
    while indice < len(cadeia):
        if cadeia[indice] == caractere:
            return indice
        indice = indice + 1
    return -1
```

6. Crie uma função que permita contar o número de vezes que aparece uma letra em uma string.

```
def contar_vezes(cadeia, letra)
  contador = 0
  for l in cadeia:
      if l == letra
          contador = contador + 1
  return contador
```

7. Crie uma função que permita inverter uma palavra dada como parâmetro.

```
def inverter_palavra(palavra):
    temporal = ""
    i = len(palavra)-1
    while i>=0:
        temporal = temporal + palavra[i]
        i=i-1
    return temporal
```

8. Crie uma função que receba duas palavras e retorne True se uma das palavras é o reverso da outra. Exemplo: 'pots' é reverso de 'stop'. 'livres' é reverso de 'servil'

```python
# Solucao 1
def reverso(palavra1, palavra2):
    if len(palavra1)!=len(palavra2):
        return False
    i = 0
    j = len(palavra2)-1
    while j>=0:
        if palavra1[i] != palavra2[j]:
            return False
        i = i+1
        j = j-1
    return True
```

```python
# Solucao 2
def reverso(palavra1, palavra2):
    if palavra1 == inverter_palavra(palavra2):
        return True
    else:
        return False
```

```python
# Solucao 3
def reverso(palavra1, palavra2):
    if len(palavra1) != len(palavra2):
        return False
    n = len(palavra1)
    i = 0
    while i<n:
        if palavra1[i] != palavra2[n-1-i]:
            return False
        i = i+1
    return True
```

9. Crie uma função que receba duas palavras e retorne True caso a primeira palavra seja um prefixo da segunda. Exemplo: 'uf' é prefixo de 'ufabc'. 'ufabc' não é prefixo de 'uf'.

```python
def prefixo(palavra1, palavra2):
    if len(palavra1)>len(palavra2):
        return False
    i = 0
    while i<len(palavra1):
        if palavra1[i]!=palavra2[i]:
            return False
        i = i+1
    return True
```

10. Crie uma função que receba duas palavras e retorne True caso a primeira palavra seja um sufixo da segunda. Exemplo: 'abc' é sufixo de 'ufabc'. 'ufabc' não é sufixo de 'abc'.

```
def sufixo(palavra1, palavra2):
    n1 = len(palavra1)
    n2 = len(palavra2)
    if n1 > n2:
        return False
    j = 0
    while j<n1:
        if palavra1[n1-1-j] != palavra2[n2-1-j]:
            return False
        j = j+1
    return True
```

11. Fazer uma função que receba como parâmetro uma string e verifique se ela é palíndroma, isto é, se ela é igual lida da esquerda para a direita e vice-versa. Exemplos: 'RADAR' é palíndroma. 'B123321B' é palíndroma. 'python' não é palíndroma.

```
# Solucao 1
def palindroma(str):
    if str == inverter_palavra(str):
        return True
    else:
        return False
```

```
# Solucao 2
def palindroma(str):
    n = len(str)
    k = 0
    while k<n/2:
        if str[k] != str[n-k-1]:
            return False
        k = k+1
    return True
```

12. Crie uma função que receba, como parâmetro, uma string e imprima somente a última palavra da mesma. Se a string for 'Universidade Federal do ABC', deverá ser impresso na tela a substring 'ABC'.

```
def ultima_palavra(frase):
    palavra = ""
    k = len(frase)-1
    while k>=0:
        if frase[k]!=" ":
            palavra = frase[k]+palavra
        else:
            break
        k = k-1
    print palavra
```

13. Escreva uma função que aceita uma string como parâmetro e retorna um número inteiro. A função deve imprimir o resultado da substituição de todos os espaços do seu parâmetro pelo caractere '-', e retorna o número de substituições feitas.

```
def substituir_brancos(frase):
    nova_frase = ''
    brancos = 0
    k = 0
    while k<len(frase):
        if frase[k]==' ':
            nova_frase = nova_frase + '-'
            brancos = brancos + 1
        else:
            nova_frase = nova_frase + frase[k]
        k = k+1
    print nova_frase
    return brancos
```

14. Crie uma função para ler uma frase (string) e contar o número de palavras dessa frase. Considere que as palavras estão separadas por espaços brancos ou vírgulas. Exemplos: 'Processamento' contém 1 palavra. 'Processamento da informação' contém 3 palavras. 'computador, caderno e caneta' contém 4 palavras. ' linux ' contém 1 palavra. ' ' não contém palavras. ' , , , ' não contém palavras.

```
# Solucao 1
def conta_palavras(frase):
    contador = 0
    k = 0
    while k<len(frase):
        if frase[k]!=" " and frase[k]!=",":
            contador = contador+1
            while frase[k]!=" " and frase[k]!="," and k<len(frase)-1:
                k = k+1
        k = k+1
    return contador
```

15. Dada uma frase (sem espaços em branco), crie uma função que permita determinar o número de segmentos consecutivos que compõem a frase. Exemplo: 'AAAAAbbbbbccccccCCCDDDDDddd' contém 6 segmentos. 'AAAAA' contém 1 segmento.

```
# Solucao 1
def segmentos_consecutivos(frase):
    if len(frase)>0:
        elemento = frase[0]
        contador = 1
        k=1
        while k<len(frase):
            if elemento!=frase[k]:
                contador = contador+1
                elemento = frase[k]
            k = k+1
    else:
        contador = 0
    return contador
```

```
# Solucao 2
def segmentos_consecutivos(frase):
    string=''
    k = 0
    for i in range(0,len(frase)):
        if string!=frase[i]:
            k+=1
            string=frase[i]
    return(k)
```

16. Dada uma frase (sem espaços em branco), crie uma função que retorne o maior segmento consecutivo que compõe a frase. Exemplo: O maior segmento consecutivo da frase 'AAAAAbbbbbccccccCCCDDDDDddd' é 'ccccccc'.

```
def maior_segmento_consecutivo(frase):
    segmento_maior = ''
    if len(frase)>0:
        elemento = frase[0]
        segmento = frase[0]
        k=1
        while k<len(frase):
            if elemento!=frase[k]:
                elemento = frase[k]
                segmento = frase[k]
            else:
                segmento = segmento+frase[k]
            if len(segmento)>len(segmento_maior):
                segmento_maior = segmento
            k = k+1
    return segmento_maior
```

17. Crie uma função que permita contar o número de ocorrências de uma palavra em uma frase. Exemplo: a frase 'ana' está presente 4 vezes na frase 'banana, mariana, e diana'.

```
def ocorrencias(frase, palavra):
    contador = 0
    k = 0
    while k<len(frase)-len(palavra)+1:
        comprimento = 0
        j=0
        while j<len(palavra):
            if frase[k+j]==palavra[j]:
                comprimento = comprimento+1
            j = j+1
        if comprimento==len(palavra):
            contador = contador+1
        k = k+1
    return contador
```

18. Crie uma função que receba uma string e duas substrings. Trocar todas as ocorrências

da primeira substring pela segunda na string. Exemplo: Seja a string 'maracatu' e as substrings 'mar' e 'sol', então a string ficará 'solacatu'.

```
def troca_ocorrencias(frase, string1, string2):
    nova_frase = ''
    i = 0
    while i < len(frase):
        contador=0
        for k in range(0,len(string1)):
            if i+k<len(frase) and frase[i+k]==string1[k]:
                contador = contador+1
        if contador==len(string1):
            nova_frase = nova_frase + string2
            i = i+contador
        else:
            nova_frase = nova_frase + frase[i]
            i = i+1
    return nova_frase
```

6 Listas

1. Indique a mensagem que apresentará a execução da seguintes função. Considere como parâmetro de entrada a lista [1,2,4,16,32,64,-128].

```
def funcao1(lista):
    temp1 = lista[0]
    temp2 = lista[len(lista)-1]
    for elemento in lista:
        if temp1>elemento:
            temp1 = elemento
        if temp2<elemento:
            temp2 = elemento
    print str(temp1) + ' ' + str(temp2)
```

Resposta: '-128 64'
Função que imprime o menor e o maior elemento de uma lista.

2. Indique a mensagem que apresentará a execução da seguintes função. Considere como parâmetro de entrada a lista [1,2,4,16,32,64,-128]

```
def funcao2(lista):
    temp1 = lista[0]
    temp2 = lista[0]
    for elemento in lista:
        if temp1>elemento:
            temp2 = temp1
            temp1 = elemento
    print str(temp1) + ' ' + str(temp2)
```

Resposta: '-128 1'
Função que imprime o primeiro e o segundo menor elemento de uma lista.

3. Indique o resultado apresentará a execução da seguintes função. Considere: L1=[1,3,4], L2=[-1,0,2,5,7,9,10].

```
def funcao3(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Resposta: [-1, 0, 1, 2, 3, 4, 5, 7, 9, 10]
A função intercala duas listas ordenadas em uma única lista ordenada na forma crescente.

4. Dadas uma lista numérica A, crie uma função que permita imprimir todos seus elementos.

```
def imprimir_lista(A):
    for i in range(0,len(A)):
        print A[i]
```

5. Dadas uma lista numérica, A e um escalar x, crie uma função que permita determinar o produto $Y = x * A$.

```
def multiplica(A,x):
    B = [ ]
    for i in range(0,len(A)):
        B.append(A[i]*x)
    return B
```

6. Crie uma função que permita somar todos os elementos de uma lista.

```
def somar_elementos(lista):
    soma = 0
    for elemento in lista:
        soma = soma + elemento
    return soma
```

7. Dadas duas listas numéricas, A e B, crie uma função que permita determinar o produto interno dessas listas.

```
def produto_interno(A,B):
    soma = 0
    if len(A)==len(B):
        for i in range(0,len(A)):
            soma = soma + A[i]*B[i]
        return soma
    print 'Listas de comprimento diferente'
```

8. Crie uma função que permita contar o número de elementos em comum entre 2 listas dadas como parâmetro. Considere listas com elementos únicos. Exemplo: L1=[1,2,3,4,5] e L2=[2,4] tem 2 elementos em comum.

```
def elementos_em_comum(L1, L2):
    contador = 0
    for elemento1 in L1:
        for elemento2 in L2:
            if elemento1==elemento2:
                contador += 1
    return contador
```

9. Crie uma função que permita intercalar os elementos de duas listas de igual comprimento. Exemplo: ['a',10,'b',20,'c',30,'d',40] é o resultado de intercalar as listas: L1=['a','b','c','d'] e L2= [10,20,30,40]

```
def intercala_listas(L1, L2):
    L3 = [ ]
    for i in range(0, len(L1)):
        L3.append(L1[i])
        L3.append(L2[i])
    return L3
```

10. entando descobrir se um dado era viciado, um dono de um casino honesto o lançou n vezes. Dada uma lista de n elementos, contendo os resultados dos lançamentos, determinar o número de ocorrências de cada face.

```
def listar_ocorrencias(Lista):
    Ocorrencias = [0]*6
    for i in Lista:
        Ocorrencias[i-1] += 1
    for i in range(0, len(Ocorrencias)):
        print 'Face '+str(i+1)+': '+str(Ocorrencias[i])
```

11. Crie uma função que permita verificar se 2 listas, dadas como parâmetro, são iguais.

```
def comparar_listas(L1,L2):
    if len(L1) == len(L2):
        for i in range(0,len(L1)):
            if L1[i] != L2[i]:
                return False
        return True
    return False
```

```
def comparar_listas(L1, L2):
    if len(L1) != len(L2):
        return false
    if len(L1)==1:
      if L1[0]==L2[0]:
          return true
      else:
          return false
    if L1[0]==L2[0]:
        La=L1[1:len(L1)]
        Lb=L2[1:len(L2)]
        return comparaLista(La,Lb)
    else:
        return false
```

12. Crie uma função que permita verificar se 2 matrizes, dadas como parâmetro, são iguais.

```
# Versao iterativa
def comparar_matrizes(M1,M2):
    if len(M1) == len(M2) and len(M1[0]) == len(M2[0]):
        for i in range(0,len(M1)):
            for j in range(0,len(M1[0])):
                if M1[i][j] != M2[i][j]:
                    return False
        return True
    return False
```

```
# Versao recursiva 1
def comparar_matrizes(M1, M2, a, b): # a e b tem que ser iguais a zero
    if len(M1)!=len(M2):
        return false
    if b==len(M1[a]):
        b=0
        a=a+1
        if a==len(M1):
            return true
    if M1[a][b]==M2[a][b]:
        return verificarMatrizes(M1,M2,a,b+1)
    return false
```

```
def comparar_matriz(M1, M2):
    if len(M1) != len(M2):
        return false
    if len(M1)==1:
       if comparaLista(M1[0],M2[0])==1:
           return true
    if comparaLista(M1[0],M2[0])==1:
       Ma=M1[1:len(M1)]
       Mb=M2[1:len(M2)]
       return comparaMatriz(Ma,Mb)
    else:
       return false
```

13. Crie uma função que permita verificar se 2 matrizes, dadas como parâmetro, são similares. Dois valores x, e y são similares se abs(x-y)¡=distancia.

```
def comparar_matrizes_aprox(M1,M2,d):
    if len(M1) == len(M2) and len(M1[0]) == len(M2[0]):
        for i in range(0,len(M1)):
            for j in range(0,len(M1[0])):
                a = M1[i][j] - M2[i][j]
                if d < abs(a):
                    return False
        return True
    return False
```

14. Crie uma função para determine o tamanho t da maior sequência de números iguais em uma lista A. Exemplo: Supor que sejam armazenados os seguintes valores para a lista A: [1,1,6,6,7,7,7,7,1,1,1], então t=4.

```
def maior_comprimento(A):
    elemento_da_sequencia = A[0]
    comprimento_atual = 1
    t = 1
    for i in range(1,len(A)):
        if A[i]==elemento_da_sequencia:
            comprimento_atual += 1
            if t<comprimento_atual:
                t = comprimento_atual
        else:
            elemento_da_sequencia = A[i]
            comprimento_atual = 1
    return t
```

15. Dadas 2 listas com n números entre 0 e 9, interpretadas como dois números inteiros de n dígitos, calcular uma lista que representa a somatória dos dois números. Exemplo:

Primeira lista:		1	2	3	4
Segunda lista:		9	9	0	0
Somatória	1	1	1	3	4

```
def somatoria(Lista1, Lista2):
    if len(Lista1)!=len(Lista2):
        return 'Listas de tamanhos diferentes!'
    Lista3 = []
    for i in range (0,len(Lista1)):
        Lista3.append(0)
    vai_um = 0
    i = len(Lista1)-1
    while i >= 0:
        soma = Lista1[i] + Lista2[i] + vai_um
        if soma >= 10:
            soma = soma - 10
            vai_um = 1
        else:
            vai_um = 0
        Lista3[i] = soma
        i = i-1
    if vai_um == 1:
        Lista3 = [1]+Lista3
    return Lista3
```

7 Matrizes

1. Execute o seguinte bloco de instruções:

```
    A = [[1,0,2], [0,2,1], [2,0,0]]
    C = [[0,0,0], [0,0,0], [0,0,0]]
    for i in range(0,3):
        for j in range(0,3):
            C[i][j] = A[A[i][j]]  [A[j][i]]
```

   Qual o valor da variável C resultante?
   Resposta:

```
[[2, 1, 0], [1, 0, 0], [0, 0, 1]]
```

   Substitua, na linha 5, a variável C por A. Qual o valor de A resultante?
   Resposta:

```
[[2, 2, 0], [0, 0, 0], [2, 2, 2]]
```

2. Dada uma matriz A, crie uma função que permita verificar se a matriz é quadrada.

```
def matriz_quadrada(A):
    if len(A)==len(A[0]):
        return True
    else:
        return False
```

3. Dada uma matriz quadrada A, crie uma função que permita contar o número de zeros contidos na matriz.

```
def conta_zeros(A):
    contador = 0
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            if A[i][j]==0:
                contador = contador+1
    return contador
```

4. Dada uma matriz A, crie uma função que determine a somatória de todos os números presentes na diagonal principal da matriz.

```
# Solucao 1
def soma_diagonal(A):
    soma = 0
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            if i==j:
                soma = soma + A[i][j]
    return soma
```

```
# Solucao 2
def soma_diagonal(A):
    soma=0
    i=0
    while i<len(A) and i<len(A[0]):
        soma = soma+A[i][i]
        i = i+1
    return soma
```

5. Dada uma matriz quadrada A, crie uma função que permita verificar se a matriz é identidade.

```
def matriz_identidade(A):
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            if i==j and A[i][j]!=1:
                return False
            if i!=j and A[i][j]!=0:
                return False
    return True
```

6. Dada uma matriz A, crie uma função que permita verificar se a matriz é simétrica.

```
def matriz_simetrica(A):
    if len(A)!=len(A[0]):
        return False
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            if A[i][j]!=A[j][i]:
                return False
    return True
```

7. Criação de matrizes

```
def criar_matriz_zeros(l,c):
    matriz = [0]*l
    for i in range(0,l):
        matriz[i] = [0]*c
    return matriz
```

```
def criar_matriz_uns(l,c):
    matriz = [1]*l
    for i in range(0,l):
        matriz[i] = [1]*c
    return matriz
```

```
def criar_matriz_identidade(n):
    matriz = [0]*n
    for i in range(0,n):
        matriz[i] = [0]*n
    for i in range(0,n):
        matriz[i][i] = 1
    return matriz
```

```
def visualizar_matriz(matriz):
    for i in range(0,len(matriz)):
        print matriz[i]
```

8. Crie uma função que permita calcular a Transposta de uma matriz dada como entrada.

```
def transposta(A):
    B = criar_matriz_zeros(len(A[0]), len(A))
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            B[j][i] = A[i][j]
    return B
```

9. Crie uma função que permita somar duas matrizes dadas como parâmetro.

```
def somar_matrizes(A,B):
    if len(A)!=len(B) or len(A[0])!=len(B[0]):
        print 'Matrizes com dimensoes diferentes'
    else:
        C = criar_matriz_zeros(len(A),len(A[0]))
        for i in range(0,len(A)):
            for j in range(0,len(A[0])):
                C[i][j] = A[i][j]+B[i][j]
        return C
```

10. Crie uma função que permita determinar o menor elemento de uma matriz dada como parâmetro.

```
def menor_elemento(A):
    menor = A[0][0]
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            if menor>A[i][j]:
                menor = A[i][j]
    return menor
```

11. Crie uma função que permita determinar apenas o segundo menor elemento de uma matriz dada como parâmetro.

```
def converter_matriz_em_lista(M):
    L = [ ]
    for i in range(0,len(M)):
        for j in range(0,len(M[0])):
            L.append(M[i][j])
    return L

def ordenar_lista(L):
    for i in range(0,len(L)-1):
        menor = i
        for j in range(i+1, len(L)):
            if L[menor]>L[j]:
                menor = j
        if menor!=i:
            temp = L[i]
            L[i] = L[menor]
            L[menor] = temp
    return L

def segundo_menor_elemento(A):
    L1 = converter_matriz_em_lista(A)
    L2 = ordenar_lista(L1)
    return L2[1]
```

12. Indique o que realiza a seguinte função

```
def funcaoM2L(M):
    L = [0]*len(M)*len(M[0])
    for i in range(0,len(M)):
        for j in range(0,len(M[0])):
            L[i*len(M[0])+j] = M[i][j]
    return L
```

Resposta: Função que converte uma matriz em uma lista

13. Crie uma função que permita verificar se a matriz, dada como parâmetro, é triangular superior.

```
def matriz_triangular_superior(A):
    if len(A[0])!=len(A):
        return False
    contador_zeros_inf = 0
    contador_zeros_sup = 0
    for i in range(0,len(A)):
        for j in range(0,len(A)):
            if i>j and A[i][j]==0:
                contador_zeros_inf +=1
            if i<j and A[i][j]==0:
                contador_zeros_sup +=1
    x = len(A)*(len(A)-1)/2
    if contador_zeros_inf==x and contador_zeros_sup!=x:
        return True
    else:
        return False
```

14. Crie uma função que permita multiplicar duas matrizes dadas como parâmetro.

```
def multiplicar_matrizes(A,B):
    nlA = len(A)
    ncA = len(A[0])
    nlB = len(B)
    ncB = len(B[0])
    if ncA!=nlB:
        return 'Matrizes com dimensoes incongruentes'
    C = criar_matriz_zeros(nlA,ncB)
    for i in range(0,nlA):
        for j in range(0,ncB):
            val = 0
            for k in range(0,ncA):
               val = val + A[i][k]*B[k][j]
            C[i][j]=val
    return C
```

15. Dizemos que uma matriz quadrada inteira é um quadrado mágico se a soma dos elementos de cada coluna e a soma dos elementos das diagonais principal e secundária são todas iguais. Dada uma matriz quadrada A, verificar se A é um quadrado mágico.

```
def quadrado_magico(Matriz):
    n = len(Matriz)
    soma = 0
    for j in range(0,n):
        soma += Matriz[0][j]
    # Para as linhas
    for i in range(0,n):
        soma_linhas = 0
        for j in range(0,n):
            soma_linhas += Matriz[i][j]
        if soma_linhas!=soma:
            return False
    # Para as colunas
    for j in range(0,n):
        soma_colunas = 0
        for i in range(0,n):
            soma_colunas += Matriz[i][j]
        if soma_colunas!=soma:
            return False
    # Para a diagonal principal e secundaria
    soma_diagonal_p = 0
    soma_diagonal_s = 0
    for k in range(0,n):
        soma_diagonal_p += Matriz[k][k]
        soma_diagonal_s += Matriz[n-k-1][k]
    if soma_diagonal_p!=soma or soma_diagonal_s!=soma:
        return False
    # Caso contrario
    return True
```

16. Crie uma função que permita realizar a somatória das linhas de uma matriz.

```
def soma_linhas(Matriz):
    S = [0]*len(Matriz)
    for i in range(0,len(Matriz)):
        soma_linhas = 0
        for j in range(0,len(Matriz[0])):
            soma_linhas += Matriz[i][j]
        S[i] = soma_linhas
    return S
```

17. Crie uma função que permita realizar a somatória das colunas de uma matriz.

```
def soma_colunas(Matriz):
    S = [0]*len(Matriz[0])
    for j in range(0,len(Matriz[0])):
        soma_colunas = 0
        for i in range(0,len(Matriz)):
            soma_colunas += Matriz[i][j]
        S[j] = soma_colunas
    return S
```

18. Crie uma função que imprima o número de linhas e colunas nulas de uma matriz dada como entrada.

```
def nulos(Matriz):
    linhas_nulas = 0
    colunas_nulas = 0
    # Para as linhas
    for i in range(0,len(Matriz)):
        soma = 0
        for j in range(0,len(Matriz[0])):
            soma += Matriz[i][j]
        if soma==0:
            linhas_nulas += 1
    # Para as colunas
    for j in range(0,len(Matriz[0])):
        soma = 0
        for i in range(0,len(Matriz)):
            soma += Matriz[i][j]
        if soma==0:
            colunas_nulas += 1
    # Imprimir resultados
    print 'Linhas nulas: '+str(linhas_nulas)
    print 'Colunas nulas:'+str(colunas_nulas)
```

19. Crie uma função que permita identificar as posições (linha,coluna) do elemento que estiver duas vezes na matriz dada como entrada.

```
def identificar_par_de_elementos(Matriz):
    for i in range(0,len(Matriz)):
        for j in range(0,len(Matriz[0])):
            elemento = Matriz[i][j]
            for k in range(0,len(Matriz)):
                for l in range(0,len(Matriz[0])):
                    if elemento==Matriz[k][l] and (i!=k or j!=l):
                        print 'Elem: '+str(elemento)
                        print 'Pos.: ('+str(i)+','+str(j)+') ('+str(k)+','+str(l)+')
                        return
    return 'Matriz sem par de elementos'
```

8 Recursividade

1. Exemplo básico.

```
# versao iterativa
def contagem_regressiva(n):
    while n>0:
        print n
        n = n-1
    print "Fogo!"
```

```
# versao recursiva
def contagem_regressiva(n):
    if n==0:
        print "Fogo!"
    else:
        print n
        contagem_regressiva(n-1)
```

2. Crie uma função que permita imprimir o n-éssimo número da sequência de Fibonacci.

```
# versao iterativa
def fibonacci(n):
    k  = 1
    t1 = 0
    t2 = 1
    while k<n:
        seguinte = t1+t2
        t1 = t2
        t2 = seguinte
        k  = k+1
    print t2
```

```
# versao recursiva
def fib(n):
    if n==0:
        return 0
    elif n==1:
        return 1
    else:
        return fib(n-1)+fib(n-2)
```

3. Crie uma função para determinar o fatorial de um número dada como entrada.

```
# versao iterativa
def fatorial(n):
    f = 1
    while n>1:
        f = f*n
        n = n-1
    return f
```

```
# versao recursiva
def fact(n):
    if n==0:
        return 1
    else:
        return n*fatorial(n-1)
```

4. Crie uma função recursiva que permita somar todos os números naturais no intervalo 1 até n (dado como entrada).

```
def somatoria(n):
    if n==1:
        return 1
    else:
        return n+somatoria(n-1)
```

5. Indique o que faz a seguinte função:

```
def recursiva():
    print 'recursiva'
    Recursiva()
```

Resposta: Recursividade infinita. A função para quando a profundidade máxima da recursividade é alcançada (StackOverflowError).

6. Crie uma função recursiva para calcula n^x.

```
def potencia(n,x):
    if x==1:
        return n
    else:
        return n*potencia(n,x-1)
```

7. Indique o resultado apresentará a execução da seguinte função. Considere como parâmetros de entrada: frase='ufabc'. O que faz a função?

```
def funcaoR(frase):
    if len(frase)==1:
        return frase
    else:
        return funcaoR(frase[1:])+frase[0]
```

Resposta: Retorna a frase (uma string) de trás para frente.

8. Indique o que realiza a seguinte função? Teste com número1=13, número2=2.

```
def funcaoC(numero1, numero2):
    if numero1<numero2:
        print numero1
    else:
        funcaoC(numero1/numero2, numero2)
        print numero1%numero2
```

Resposta: Converte o numero1, da base 10, para a base numero2.

```
1
1
0
1
```

9 Algoritmos de ordenação

1. Selection sort

```python
# Versao 1
def selection_sort(L):
    for i in range(0,len(L)-1):
        menor = i
        for j in range(i+1, len(L)):
            if L[menor]>L[j]:
                menor = j
        if menor!=i:
            temp = L[i]
            L[i] = L[menor]
            L[menor] = temp
    return L
```

```python
# Versao 2
def selection_sort(L):
    for i in range(0,len(L)-1):
        menor = i
        for j in range(i+1, len(L)):
            if L[menor]>L[j]:
                menor = j
        L[i], L[menor] = L[menor],L[i]
    return L
```

```python
# Versao 3 (recursiva)
def selection_sort_rec(L, indice):
    print L
    if indice >=len(L)-1:
        return L
    menor = indice
    for j in range(indice+1, len(L)):
        if L[menor]>L[j]:
            menor = j
    L[indice], L[menor] = L[menor],L[indice]
    return selection_sort_rec(L, indice+1)
```

2. Bubble sort.

```python
def bubble_sort(L):
    j = len(L)-1
    while j>0:
        for i in range(0,j):
            if L[i]>L[i+1]:
                L[i],L[i+1] = L[i+1],L[i]
        j = j-1
    return L
```

3. Insertion sort.

```
def insertion_sort(L):
    for i in range(1,len(L)):
        j = i
        while j>=1 and L[j-1]>L[j]:
            L[j-1],L[j] = L[j],L[j-1]
            j = j-1
    return L
```

4. Cocktail sort.

```
# Versao iterativa
def cocktail_sort(L):
    k = len(L)-1
    p = 0
    while k > p:
        for i in range(p,k): #programa esta andando para direita
            if L[i]>L[i+1]:
                L[i],L[i+1]=L[i+1],L[i]
        j = k - 1
        while j>p: #programa esta andando para a esquerda
            if L[j]<L[j-1]:
                L[j],L[j-1]=L[j-1],L[j]
            j = j-1
        k = k-1
        p = p+1
    return L
```

```
# Versao recursiva
def cocktail_sort(lista):
    swaped = False
    for i in range(len(lista) - 1):
        if lista[i] > lista[i+1]:
            lista[i], lista[i+1] = lista[i+1], lista[i]
            swaped = True
    if swaped==False:
        return lista
    swaped = False
    for i in range(len(lista) - 2, 0, -1):
        if lista[i] < lista[i - 1]:
            lista[i], lista[i-1] = lista[i-1], lista[i]
            swaped = True
    if swaped == False:
        return lista
    return [lista[0]] + coktail(lista[1:-1]) + [lista[-1]]
```

5. Merge sort

```python
def merge_sort(L):
    if len(L) <= 1:
        return L
    meio = len(L)/2
    L1 = merge_sort(L[:meio])
    L2 = merge_sort(L[meio:])
    return intercala(L1, L2)

def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

6. Seja L uma lista de n elementos. O algoritmo de ordenação chamado Quick-sort é o seguinte:
   (a) Reorganize L de tal maneira que $L[0]$ ocupe uma nova posição r, com os elementos $L[0], \ldots, L[r-1]$ todos menores ou iguais a $L[0]$ e os elementos $L[r+1], \ldots, L[n-1]$ todos maiores a $L[0]$.
   (b) Aplique Quick-sort recursivamente nas porções $L[0], \ldots, L[r-1]$ e $L[r+1], \ldots, L[n-1]$.
   A lista resultante estará ordenada.
   Escreva uma função recursiva que permita ordenar uma lista L de n elementos.

```
# Versao recursiva
def quick_sort(L):
    if len(L)<=1:
      return L
    L1 = [ ]
    L2 = [ ]
    for i in range(1, len(L)):
        if L[i]<=L[0]:
            L1.append(L[i])
        else:
            L2.append(L[i])
    return quick_sort(L1) + [L[0]] + quick_sort(L2)
```

10 Conjuntos (através de listas)

1. Crie uma função que permita verificar se duas listas (conjuntos) são equivalentes. Apenas use listas

```
def conjuntos_iguais(A, B):
    for a in A:
        if not a in B:
            return False
    for b in B:
        if not b in A:
            return False
    return True
```

2. Crie uma função que permita unir dois conjuntos A e B.

```
def uniao_conjuntos(A, B):
    C = [ ]
    for a in A:
        if not a in C:
            C.append(a)
    for b in B:
        if not b in C:
            C.append(b)
    return C
```

3. Crie uma função que calcule a interseção de dois conjuntos A e B dados como entrada.

```
def intersecao_conjuntos(A, B):
    C = []
    for a in A:
        if a in B and not a in C:
            C.append(a)
    return C
```

11 Busca de dados

1. Busca sequencial iterativo.

```python
# Versao 1
def busca_sequencial(x, L):
    for i in range(0,len(L)):
        if x==L[i]:
            return i
    return -1
```

```python
# Versao 2
def busca_sequencial(x, L):
    for i in range(0,len(L)):
        if x==L[i]:
            return True
    return False
```

2. Busca sequencial recursivo.

```python
def busca_rec(x, L):
    if len(L)==0:
        return False
    else:
        if x==L[0]:
            return True
        else:
            return busca_rec(x,L[1:])
```

3. Busca binária.

```python
def busca_binaria(x, L):
    if len(L)==0:
        return False
    meio = len(L)/2
    if L[meio]==x:
        return True
    else:
        if x<L[meio]:
            return busca_binaria(x, L[:meio])
        else:
            return busca_binaria(x, L[meio+1:])
```

12 Prova Teoria 01: 2013-Q1

1. Indique a mensagem que apresentará a execução da seguinte função.

```python
def operacoes(x1, y1, x2, y2, ra):
    p = 0
    while p < (x2-x1)**2 + (y2-y1)**2:
        if x1 < x2:
            print str(ra) + '-' + str(x1)
        else:
            print str(ra) + '-' + str(y1)
        p = p+2
```

- Considerando como parâmetros: x1=2, y1=3, x2=4, y2=5, ra=112233.
  Resposta:

  ```
  112233-2
  112233-2
  112233-2
  112233-2
  ```

- Considerando como parâmetros: x1=4, y1=5, x2=3, y2=2, ra=112233.
  Resposta:

  ```
  112233-5
  112233-5
  112233-5
  112233-5
  112233-5
  ```

2. Indique a mensagem que apresentará a execução da seguinte função.

```
def operacoes_xyz(x, y, z, ra):
    p = 0
    while p < (x+y+z)**2:
        p = p+1
    if p%2==0:
        print str(ra) + '-' + str(x)
    else:
        print str(ra) + '-' + str(y)
```

- Considerando como parâmetros: x=1, y=2, z=3, ra=112233.
  Resposta:

  ```
  112233-1
  ```

- Considerando como parâmetros: x=3, y=2, z=1, ra=112233.
  Resposta:

  ```
  112233-3
  ```

3. Considere a função `enigma`. Indique apenas a resposta que apresentará a execução da função quando a entrada seja igual a seu nome completo. Por exemplo, se seu nome completo for 'José Silva da Costa', a execução da função enigma deve considerar como único parâmetro de entrada 'José Silva da Costa'.

```
def enigma(nome):
    resposta = '' # nao tem espaco em branco
    k = len(nome)-1
    while k>0:
        aux = ''  # nao tem espaco em branco
        p = k
        while p>=0 and nome[p]!=' ':  # espaco em branco
            aux = nome[p] + aux
            p = p-1
        if len(resposta)<len(aux):
            resposta = aux
            k = p
        k = k-1
    return resposta
```

- Considerando como parâmetro: 'José Silva da Costa'
  Resposta: `'Costa'`
- Considerando como parâmetro: 'Carlos Rodrigues de Salles'
  Resposta: `'Rodrigues'`

(*) Note que a função `enigma`, dada uma frase (com o sem espaços), determina a última palavra de maior comprimento.

4. Considere a função `n_enigma`. Indique apenas a resposta que apresentará a execução da função quando a entrada seja igual a seu nome completo. Por exemplo, se seu nome completo for 'José Silva da Costa', a execução da função enigma deve considerar como único parâmetro de entrada 'José Silva da Costa'.

```
def n_enigma(nome):
    resposta = nome
    k = len(nome)-1
    while k>=0:
        aux = '' # nao tem espaco em branco
        p = k
        while p>=0 and nome[p]!=' ':  # espaco em branco
            aux = nome[p] + aux
            p = p-1
        if len(resposta)>len(aux) and len(aux)>=1:
            resposta = aux
        k = p-1
    return resposta
```

- Considerando como parâmetro: 'José Silva da Costa'
  Resposta: `'da'`
- Considerando como parâmetro: 'Roberto Carusso Barreira'
  Resposta: `'Carusso'`

(*) Note que a função `n_enigma`, dada uma frase (com o sem espaços), determina a última palavra de menor comprimento.

5. Escreva uma função, com nome `eliminar_caracteres`, que aceita uma string, e dois caracteres como parâmetros e retorna a mesma string onde ambos os caracteres sejam eliminados. Por exemplo, o chamado a função `eliminar_caracteres('momentum', 'm','n')` deverá retornar 'oetu'.

```
def eliminar_caracteres(frase, a, b):
    nova_frase = ''
    i = 0
    while i<len(frase):
        if frase[i]!=a and frase[i]!=b:
            nova_frase = nova_frase + frase[i]
        i = i+1
    return nova_frase
```

6. Escreva uma função, com nome `substituir_caractere`, que aceita uma string, e dois caracteres como parâmetros e retorna a mesma string onde o primeiro caractere seja substituído pelo segundo caractere. Por exemplo, o chamado a função `substituir_caractere('palavra','a','o')` deverá retornar 'polovro'.

```
def substituir_caractere(frase, a, b):
    nova_frase = ''
    i = 0
    while i<len(frase):
        if frase[i]==a:
            nova_frase = nova_frase + b
        else:
            nova_frase = nova_frase + frase[i]
        i = i+1
    return nova_frase
```

7. Escreva uma função, com nome `eliminar_vogais`, que aceita uma string como parâmetros e retorna a mesma string sem vogais. Por exemplo, o chamado a função `eliminar_vogais('obrigado')` deverá retornar 'brgd'.

```
# Solucao 1
def eliminar_vogais(frase):
    nova_frase = ''
    i = 0
    while i<len(frase):
        if frase[i]!='a' and frase[i]!='e' and frase[i]!='i' and \
           frase[i]!='o' and frase[i]!='u':
            nova_frase = nova_frase + frase[i]
        i = i+1
    return nova_frase
```

```
# Solucao 2
def eliminar_vogais(frase):
    nova_frase = ''
    i = 0
    while i<len(frase):
        if not frase[i] in 'aeiouAEIOU':
            nova_frase = nova_frase + frase[i]
        i = i+1
    return nova_frase
```

8. Escreva uma função, com nome `duplicar_vogais`, que aceita uma string como parâmetros e retorna a mesma string com as vogais duplicadas. Por exemplo, o chamado a função `duplicar_vogais('obrigado')` deverá retornar 'oobriigaadoo'.

```
# Solucao 1
def duplicar_vogais(frase):
    nova_frase = ''
    i = 0
    while i<len(frase):
        if frase[i]=='a' or frase[i]=='e' or frase[i]=='i' or \
           frase[i]=='o' or frase[i]=='u':
            nova_frase = nova_frase + frase[i] + frase[i]
        else:
            nova_frase = nova_frase + frase[i]
        i = i+1
    return nova_frase
```

```
# Solucao 2
def duplicar_vogais(frase):
    nova_frase = ''
    i = 0
    while i<len(frase):
        nova_frase = nova_frase + frase[i]
        if frase[i] in 'aeiouAEIOU':
            nova_frase = nova_frase + frase[i]
        i = i+1
    return nova_frase
```

9. Crie uma função para calcular e imprimir na tela a seguinte somatória

$$\left(2\sum_{i=1}^{n} i^2\right) + \left(3\sum_{j=1}^{n} j^3\right) + \left(4\sum_{j=1}^{n} j^4\right)$$

Em que n é um número inteiro > 0 dado como parâmetro.

```
def somatoria1(n):
    soma = 0
    for j in range(1,n+1):
        soma = soma + 2*(j)**2 + 3*(j)**3 + 4*(j)**4
    print soma
```

10. Crie uma função para calcular e imprimir na tela a seguinte somatória

$$(2n)! + \left(3\sum_{j=1}^{n} j^3\right) + \left(4\sum_{j=1}^{n} j^4\right)$$

Em que n é um número inteiro > 0 dado como parâmetro.

```
def somatoria2(n):
    fact = 1
    for i in range(1,2*n+1):
        fact = fact*i
    soma = fact
    for j in range(1,n+1):
        soma = soma + 3*(j)**3 + 4*(j)**4
    print soma
```

11. Crie uma função para calcular e imprimir na tela a seguinte somatória

$$(2\sum_{p=1}^{n} p)! + \left(3\sum_{p=1}^{n} p^3\right)$$

Em que n é um número inteiro > 0 dado como parâmetro.

```
def somatoria3(n):
    soma1 = 0
    soma2 = 0
    for p in range(1,n+1):
        soma1 = soma1 + p
        soma2 = soma2 + p**3
    fact = 1
    for i in range(1,2*soma1+1):
        fact = fact*i
    print fact + 3*soma2
```

12. Crie uma função para calcular e imprimir na tela a seguinte somatória

$$\left(2\sum_{p=1}^{2n} p^2\right) - (3\sum_{p=1}^{n} p)!$$

Em que n é um número inteiro > 0 dado como parâmetro.

```
def somatoria4(n):
    soma1 = 0
    for p in range(1,2*n+1):
        soma1 = soma1 + p**2
    soma2 = 0
    for p in range(1,n+1):
        soma2 = soma2 + p
    fact = 1
    for i in range(1,3*soma2+1):
        fact = fact*i
    print 2*soma1 - fact
```

13 Prova Teoria 02: 2013-Q1

1. O que será impresso pela função a seguir?

```
def enigma():
    M = [[0,0,0,0],[0,0,0,0],[0,0,0,0]]
    for t in range(0,len(M[0])):
        i = 0
        while i<3:
            M[i][t] = ((t*len(M))+i+1)*2
            i = i+1
    for t in range(0,len(M)):
        print M[t]
```

```
Resposta:
[2, 8, 14, 20]
[4, 10, 16, 22]
[6, 12, 18, 24]
```

2. O que será impresso pela função a seguir?

```
def enigma():
    M = [[0,0,0,0],[0,0,0,0],[0,0,0,0]]
    for t in range(0,len(M[0])):
        i = 0
        while i<3:
            M[i][t] = ((t*len(M))+i+1)*2-1
            i = i+1
    for t in range(0,len(M)):
        print M[t]
```

```
Resposta:
[1, 7, 13, 19]
[3, 9, 15, 21]
[5, 11, 17, 23]
```

3. Considere A uma matriz. A normalização de uma matriz é realizada, dividindo cada elemento da matriz A pelo maior elemento da linha correspondente. Crie uma função que permita normalizar uma matriz de números inteiros dada como parâmetro.

```python
def normalizar_matriz(A):
    B = criar_matriz_zeros(len(A),len(A[0]))
    for i in range(0,len(A)):
        max = A[i][0]
        for j in range(0,len(A[0])):
            if A[i][j]>max:
                max = A[i][j]
        for j in range(0,len(A[0])):
            B[i][j] = float(A[i][j])/max
    return B

def criar_matriz_zeros(l,c):
    matriz = [0]*l
    for i in range(0,l):
        matriz[i] = [0]*c
    return matriz
```

4. Escreva uma função que recebe uma matriz de caracteres 8x8 representando um tabuleiro de xadrez e calcula o valor total das peças do jogo. Espaços vazios do tabuleiro são codificados como casas com ' ' (branco) e têm valor 0 (zero). O valor das demais peças e sua representação na matriz são dados de acordo com a tabela:

Peca	Representacao	Valor
Peao	'P'	1
Cavalo	'C'	3
Bispo	'B'	3
Torre	'T'	5
Rainha	'D'	10
Rei	'R'	50

```python
def calcula_valor_jogo(M):
    pecas = [' ','P','C','B','T','D','R']
    valores = [0,1,3,3,5,10,50]
    valor = 0
    for i in range(0,8):
        for j in range(0,8):
            for p in range(0,len(pecas)):
                if pecas[p]==M[i][j]:
                    valor += valores[p]
    return valor
```

5. Crie uma função para calcular e imprimir os n primeiros números de Tribonacci. A serie de Tribonacci consiste em: 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, 274, 504,... Para calculá-la o primeiro elemento vale 1, o segundo elemento vale 1, o terceiro elemento vale 2, e daí por diante. Assim, o i-ésimo elemento vale o (i-1)-ésimo elemento somado ao (i-2)-ésimo elemento somado ao (i-3)-ésimo elemento. Exemplo, 13=7+4+2. Observe que n deve ser positivo.

```
# Versao iterativa
def tribonacci(n):
    t = [1,1,2]
    if n<4:
        return t[:n]
    for i in range(3,n):
        t.append(t[-1]+t[-2]+t[-3])
    return t
```

```
# Versao recursiva
def tribonacci(n):
    if n>0:
        for i in range(1,n+1):
            print trib(i)

def trib(n):
    if n==1:
        return 1
    if n==2:
        return 1
    if n==3:
        return 2
    if n>3:
        return trib(n-1)+trib(n-2)+trib(n-3)
```

6. Dada uma sequência de n números reais, determinar os números que compõem a sequência e o número de vezes que cada um deles ocorre na mesma.

   Por exemplo para a lista L=[-1.7, 3.0, 0.0, 1.5, 0.0, -1.7, 2.3, -1.7], a saída deverá ser:

```
-1.7 ocorre 3 vezes
3.0 ocorre 1 vez
0.0 ocorre 2 vezes
1.5 ocorre 1 vez
2.3 ocorre 1 vez
```

```
def frequencia(L):
    listaElementos = []
    listaFrequencias = []
    for e in L:
        if not e in listaElementos:
            listaElementos.append(e)
            listaFrequencias.append(1)
        else:
            for i in range(0,len(listaElementos)):
                if e==listaElementos[i]:
                    listaFrequencias[i]+=1
                    break
    for j in range(0,len(listaElementos)):
        print str(listaElementos[j])+' ocorre '+str(listaFrequencias[j])+' vez(es)'
```

7. O máximo divisor comum (mdc) entre dois números inteiros é o maior número inteiro que é fator de tais números. Por exemplo, os divisores comuns de 12 e 18 são 1,2,3 e

6, logo mdc(12,18)=6. Crie uma função que permita calcular e imprimir o máximo divisor comum de dois números inteiros. mdc(a,b)

```
# Primeira versao
def mdc(a,b):
    i = 1
    while i<=a and i<=b:
        if a%i==0 and b%i==0:
            div = i
        i = i+1
    return div
```

```
# Segunda versao
def mdc(a,b):
    if b>a:
        a,b = b,a
    while a-b!=0:
        a = a-b
        if b>a:
            a,b = b,a
    return a
```

8. Dado um número n, seja inv(n) o número que se obtém invertendo-se a ordem dos dígitos de n. Por exemplo inv(322)= 223. Um número é palíndromo se inv(n)=n. Por exemplo, 34543, 1, 99. Escreva uma função que receba como parâmetro um número inteiro n e verifique se n é palíndromo, escrevendo a resposta adequada. Nota: Não podem ser utilizadas funções de conversão de número a string, e vice-versa.

```
def palindromo(n):
    num = n
    rev = 0;
    while num>0:
        dig = num%10
        rev = rev*10 + dig
        num = num/10
    if n==rev:
        return true
    else:
        return false
```

9. Dados o número real, x, e um erro, epsilon, a seguinte recorrência descreve uma função $F(x)$ que permite aproximar e^x a um valor y tal que $|y - e^x| <= epsilon$.

$$F(x, epsilon) = \begin{cases} \frac{1}{F(-x, epsilon)} & , \text{se } x < 0 \\ (F(\frac{x}{2}, epsilon))^2 & , \text{se } x > epsilon \\ 1 + x & , \text{se } 0 < x \leq epsilon \end{cases}$$

```
def F(x,epsilon):
    if x<0:
        return 1/F(-x, epsilon)
    elif x>epsilon:
        valor = F(x/2.0,epsilon)
        return valor**2
    else:
        return 1+x
```

10. Observe que:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n}{k}\frac{(n-1)!}{(k-1)!(n-k)!} = \frac{n}{k}\binom{n-1}{k-1}$$

Escreva uma função recursiva que calcule binomial(n,k) usando a propriedade acima.

```
def coef_binomial(n,k):
    if k==1:
        return n
    else:
        return float(n)/k*coef_binomial(n-1,k-1)
```

14 Prova Pratica 01: 2013-Q1

1. Escreva uma função que permita inverter uma string dada como entrada. Por exemplo, se a string dada como entrada for 'inverter', a resposta deverá ser 'retrevni'.

```
def inverter(frase):
    frase2 = ''
    for p in frase:
        frase2 = p + frase2
    return frase2
```

2. Dado n e dois números inteiros positivos i e j, imprimir em ordem crescente os n primeiros números naturais que são múltiplos de i ou de j ou de ambos. Por exemplo, para n=6, i=2 e j=3 a saída deverá ser: 0 2 3 4 6 8.

```
def multiplos(n, i, j):
    contador = 1
    k = 0
    while contador<=n:
        if k%i==0 or k%j==0:
            print k
            contador = contador+1
        k = k+1
```

3. Dizemos que um número i é congruente módulo m a j si $i\%m = j\%m$. Por exemplo, 35 é congruente módulo 4 a 39, pois $35\%4 = 3 = 39\%4$. Escreva uma função, com definição 'listar_congruentes(n,j,m)', que permita imprimir os n primeiros naturais congruentes a j módulo m.

```
def congruentes(i,j,m):
    if i%m==j%m:
        return True
    else:
        return False

def listar_congruentes(n,j,m):
    numero_de_congruentes=0
    i = 1
    while numero_de_congruentes<n:
        if congruentes(i,j,m):
            print i
            numero_de_congruentes = numero_de_congruentes+1
        i = i+1
```

4. Dizemos que um inteiro positivo n é perfeito se for igual à soma de seus divisores positivos diferentes de n. Por exemplo, 6 é perfeito, pois 1+2+3=6. Escreva uma função, com definição 'listar_perfeitos(k)', que permita imprimir os k primeiros números perfeitos.

```
# Solucao 1
def listar_perfeitos(k):
    numero_de_perfeitos = 0
    i = 2
    while numero_de_perfeitos<k:
        soma=0
        for j in range(1,i):
            if i%j==0:
                soma = soma+j
        if soma==i:
           print i
           numero_de_perfeitos = numero_de_perfeitos+1
        i = i+1
```

```
# Solucao 2
def listar_perfeitos(k):
    numero_de_perfeitos=0
    i = 1
    while numero_de_perfeitos<k:
        if perfeito(i):
            print i
            numero_de_perfeitos = numero_de_perfeitos+1
        i = i+1

def perfeito(n):
    soma = 0
    for i in range(1,n):
        if n%i==0:
            soma = soma+i
    if soma==n:
        return True
    else:
        return False
```

5. Sabe-se que um número da forma n^3 é igual a soma de n ímpares consecutivos. Exemplo: $1^3 = 1$, $2^3 = 3 + 5$, $3^3 = 7 + 9 + 11$, $4^3 = 13 + 15 + 17 + 19$. Escreva uma função, com definição 'impares_consecutivos(n)', que permita imprimir os n ímpares consecutivos cuja soma seja igual a n^3.

```
# Solucao 1
def impares_consecutivos(n):
    impares = []
    for k in range(1,n**3+1):
        if k%2==1:
            impares.append(k)
    for i in range(0, len(impares)-n+1):
        soma = 0
        for j in range(0,n):
            soma = soma+impares[i+j]
        if soma==n**3:
           indice=i
           break
    for p in range(indice, indice+n):
        print impares[p]
```

```
# Solucao 2
def impares_consecutivos(n):
    soma = 0
    impar = n**2-(n-1)
    while soma!=n**3:
        print impar
        soma  = soma+impar
        impar = impar+2
```

```
# Solucao 3
def impares_consecutivos(n):
    x = 1
    j = 1
    k = 0
    var = 0
    cont = 0
    p = 0
    lista = ''
    while cont < n:
        while x**3 != x*j+k:
             j = j + 2
        x = x + 1
        p = p + 2
        k = k + p
        cont = cont + 1
        var = j
        for i in range(0,x-1):
            lista = lista + ' + ' + str(var)
            var = var + 2
        print str(x-1)+'^3 = '+ lista
        lista = ''
```

```
# Solucao 4
def impares_consecutivos(n):
    i = 0
    soma = 0
    j = 2*((((n-1)*(n))/2)+1)-1
    while i<n:
        print j
        soma = soma+j
        j = j+2
        i = i+1
    print soma
```

15 Prova Pratica 02: 2013-Q1

1. Crie uma função que receba como parâmetro um número inteiro que representa a idade de um nadador e permita classifica-lo em uma das seguintes categorias: adulto (idade≥18), juvenil (14≤idade<18), infantil (9≤idade<14) e mirim (idade<9).

```
def classificar_categoria(idade):
    if idade>=18:
        print "Adulto"
    if 14<=idade and idade<18:
        print "Juvenil"
    if 9<=idade and idade<14:
        print "Infantil"
    if idade<9:
        print "Mirim"
```

2. Crie uma função que permita verificar se uma matriz é uma matriz de permutação. Uma matriz de permutação é uma matriz quadrada cujos elementos são 0's ou 1's, tal que em cada linha e em cada coluna exista um, e apenas um, elemento igual a 1. Exemplo:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
def verificar_permutacao(Matriz):
    for k in range(0,len(Matriz)):
        # consultamos a linha k
        uns_na_linha = 0
        for t in range(0,len(Matriz)):
            if Matriz[k][t]==1:
                uns_na_linha += 1
            elif Matriz[k][t]!=0:
                return false
        # consultamos a coluna k
        uns_na_coluna = 0
        for t in range(0,len(Matriz[0])):
            if Matriz[t][k]==1:
                uns_na_coluna += 1
            elif Matriz[k][t]!=0:
                return false
        if uns_na_linha>1 or uns_na_coluna>1:
            return false
    return true
```

3. Escreva uma função que calcule a aproximação para a integral:

$$\int_0^x e^{-u^2} du = x - \frac{x^3}{3 \times 1!} + \frac{x^5}{5 \times 2!} - \frac{x^7}{7 \times 3!} + \ldots$$

Interrompa o cálculo quando o k-ésimo termo ficar menor (em módulo) que uma certa constante (epsilon) dada como parâmetro.

```
def integralE(x,epsilon):
    x = float(x)
    termo = x
    soma = 0
    fact = 1
    i = 1
    while abs(termo)>epsilon:
        soma = soma+termo
        ind = 2*i+1
        fact = fact*i
        termo = (x**ind)/(ind*fact)*((-1)**i)
        i = i+1
    return soma
```

4. Escreva uma função que permita encontrar os primos contíguos (anterior e posterior)

de um número inteiro $n \geq 2$ dado como entrada. Por exemplo, o número 15 tem como números contíguos 13 e 17. O número 100 tem como números contíguos 97 e 101.

```
def encontrar_primos(n):
    i = n+1
    while not primo(i):
        i += 1
    primo_superior = i
    i = n-1
    while not primo(i):
        i -= 1
    primo_inferior = i
    print 'Primo superior: '+str(primo_superior)
    print 'Primo inferior: '+str(primo_inferior)

def primo(p):
    contador = 0
    for i in range(1,p+1):
        if p%i==0:
            contador = contador+1
    if contador==2:
        return True
    else:
        return False
```

5. Crie uma função que receba uma matriz 10X10, calcule e escreva a média dos elementos localizados na área marcada com '*'

*									
*	*								
*	*	*							
*	*	*	*						
*	*	*	*	*					
*	*	*	*	*	*				
*	*	*	*	*	*	*			
*	*	*	*	*	*	*	*		
*	*	*	*	*	*	*	*	*	
*	*	*	*	*	*	*	*	*	*

```
def media(Matriz):
    elementos = 0
    soma = 0.0
    for i in range(0,10):
        for j in range(0,10):
            if i>=j:
                elementos += 1
                soma += Matriz[i][j]
    print soma/elementos
```

6. Crie uma função que receba uma matriz 10X10, calcule e escreva a média dos elementos localizados na área marcada com '*'

	*	*	*	*	*	*	*	*	*
		*	*	*	*	*	*	*	*
			*	*	*	*	*	*	*
				*	*	*	*	*	*
					*	*	*	*	*
						*	*	*	*
							*	*	*
								*	*
									*

```
def media2(Matriz):
    elementos = 0
    soma = 0.0
    for i in range(0,10):
        for j in range(0,10):
            if i<j:
                elementos += 1
                soma += Matriz[i][j]
    print soma/elementos
```

7. Crie uma função que receba uma string (frase) e um caractere qualquer e permita construir uma lista contendo as posições (índices) de onde ocorre o caractere na string. Exemplo: Seja a string 'abracadabra!!!' e o caractere 'a', então a lista de índices deverá conter os seguintes valores: [0 3 5 7 10].

```
def posicoes(frase, caractere):
    lista = []
    for i in range(0,len(frase)):
        if frase[i]==caractere:
            lista.append(i)
    return lista
```