FÍSICA COMPUTACIONAL 1 (DFTE-FIS0610)

COMANDOS BÁSICOS E EXERCÍCIOS PARA O CAPÍTULO 2 - PARTE 2

Trabalhando com o 'for' em python:

range

Cria um iterador contendo uma lista de valores.
Ex: range(5) retorna a sequência [0,1,2,3,4]
Ex2: range(2,8) retorna a sequência [2,3,4,5,6,7]
Ex3: range(2,20,3) retorna a sequência [2,5,8,11,14,17]

for

O "for" tem sintaxe diferente no python e no C. Vamos dar um exemplo:

```
a = [2,4,6]
for i in a:
    print(i)
```

Este código imprime os numeros 2, 4 e 6 em sequência. O "for" em python recebe uma lista (ou vetor) e a variável "i" do loop passa por todos os elementos da lista. Tipicamente se usa o range() para criar a lista que é usada pelo "for". Por exemplo, o seguinte "for" em C "for (i=3; i<12; i=i+2)" é escrito desta maneira em python:

```
for i in range(3,12,2):
    print (i)
```

Exercício 2.1: Trabalhando com o "for"

Use a função range em conjunto com o "for" para criar um programa que imprime:

a) Os números inteiros menores que 10

b) Os números pares menores que 20

c) Os múltiplos de 5 menores que 50, em ordem decrescente

Exercício 2.2: Soma dos inversos

Escreva um programa que calcula a somatória

$$s = \sum_{k=1}^{100} \frac{1}{k}.$$

O resultado desta soma é 5.187.

Vetores e matrizes

from numpy import *
Importa as funções da biblioteca numpy. **As funções discutidas abaixo devem ser importadas do numpy.**

arange
Funciona da mesma maneira que range(), porém cria vetores ao invés de iteradores. Também aceita argumentos do tipo float.
Ex: a = arange(0.0,1.0,0.1)

zeros
Cria vetores ou matrizes em que todos elementos são iguais a zero.
Ex: a = zeros(10,float). Cria um vetor com 10 elementos do tipo float. Todos elementos valem zero.
Ex2: A = zeros([3,4],int). Cria uma matriz com 3 linhas e 4 colunas do tipo int.

loadtxt
Importa dados de um arquivo de texto externo.
Ex: A = loadtxt("dados.txt",float)

array
Transforma uma lista em vetor. Útil para gerar matrizes.
Ex: a= array (lista)
Ex2: A= array ([[1,0],[0,1]],int)

Aparentemente não precisa informar o tipo do dado, basta apenas explicitá-lo.

e.g.: A = array([[1.0, 0.0],[0, 1]])

Deixando apenas um explícito, ele já entende que todos os outros dados são daquele tipo.

Exercício 2.3: Criando vetores e matrizes

Escreva um programa que cria:

a) Um vetor contendo 100 números entre 0 e 1, igualmente espaçados (dica: use a função *arange* do numpy).

b) A matriz,

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

Tente realizar este procedimento três vezes, usando as funções *zeros*, *array* e *loadtxt*.

Exercício 2.4: Suponha vetores `a` e `b` definidos da seguinte maneira:

```
from numpy import array
a = array([1,2,3,4],int)
b = array([2,4,6,8],int)
```

O que o computador vai imprimir ao executar as seguintes linhas? (Tente descobrir a resposta antes de rodar o código no computador.)

a) `print(b/a+1)`
b) `print(b/(a+1))`
c) `print(1/a)`

Operações com matrizes

As funções discutidas abaixo devem ser importadas do numpy.

log,sqrt,exp... As funções matemáticas discutidas para a biblioteca math podem ser encontradas no numpy. Uma vantagem das funções incluídas no numpy é que elas aceitam vetores como argumento.

max, min, sum Funcionam da mesma maneira das funções de mesmo nome discutidas na aula anterior, porém funcionam com matrizes.

size
Fornece o número de elementos de uma matriz. Se aplicado a uma matriz 3x3, por exemplo, informa que a matriz tem 9 elementos.

shape
Fornece a forma de uma matriz. Seja A uma matriz 3x3. shape (A) retorna (3,3).

dot
Realiza o produto escalar de dois vetores. Também é usado para multiplicar matrizes.
Ex: C = dot(A,B)

Exercício 2.5: Operações com matrizes - parte 1

Crie a matriz B abaixo:

$$B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}.$$

Em seguida, realize as seguintes operações:

a) Multiplique B por 2

b) Some a matriz B e a matriz A definida no exercício 2.3.

c) Multiplique as matrizes A e B. O resultado obtido foi o esperado? Tente usar a função *dot* disponível no pacote numpy.

Exercício 2.6: Operação com matrizes - parte 2

Utilize a matriz B definida no exercício anterior e funções disponíveis no pacote numpy para determinar (dica: se a função apropriada for usada, cada letra pode ser resolvida em uma linha):

a) O maior e o menor elemento da matriz B

b) O valor médio dos elementos da matriz B

c) Para criar uma matriz C, em que cada elemento é a raiz quadrada de um elemento da matriz B.

Copiando vetores

copy
Gera uma cópia de um vetor.
Ex: b = copy(a)

Exercício 2.7: Copiando vetores

a) Realize a seguinte sequência de passos: (i) Crie um vetor a = [1,1]; (ii) crie um segundo vetor b=a; (iii) Mude o primeiro elemento de a: a[0]=2; imprima os vetores a e b. Você esperava este resultado?

b) Utilize agora a função copy do numpy para criar o vetor b: b = copy(a). Repita os passos do item anterior.

Fatiamento

Seleciona parte dos elementos de um vetor ou lista. Seja um vetor 'r'. A sintaxe para realizar o fatiamento é $r[n1 : n2]$. n1 indica o início do intervalo em que valores serão selecionados. O último elemento selecionado é o de índice $n2 - 1$. Teste os comandos abaixo, a parte comentada é o resultado esperado (lembre que o primeiro elemento do vetor 'r' é r[0]):

```
r = [3,5,7,9,11,13,15]
s = r[2:5] # s = [7,9,11]
s = r[:4] # s = [3,5,7,9]
s = r[4:] # s = [11,13,15]
s = r[:] # s = [3,5,7,9,11,13,15]
```

Para uma matriz A, a sintaxe do fatiamento é $A[l1 : l2, c1 : c2]$. Os argumentos antes da vírgula especificam as linhas desejadas e os depois da linha as colunas desejadas. Podemos também dar um único argumento antes ou depois da vírgula. Neste caso, só uma linha ou

coluna será considerada. Considere, por exemplo, o comando $A[0:2,0]$. Como só demos um argumento depois da vírgula, apenas elementos da coluna indicada (0) serão selecionados. Os argumentos antes da vírgula indicam que queremos as duas primeiras linhas. Logo, os elementos $A[0,0]$ e $A[1,0]$ serão selecionados.

Exercício 2.8:

Crie a matriz D,

$$D = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}.$$

Em seguida, use fatiamento para criar:

a) Um vetor cujos elementos são os da primeira linha de D

b) Um vetor cujos elementos são os da segunda coluna de D

c) Um vetor contendo os dois últimos elementos da terceira linha de D

Funções definidas pelo usuário

def

Permite ao usuário definir uma função. Por exemplo, a seguinte função recebe um ângulo em graus e retorna ele em radianos:

```
from math import pi
def convert(theta):
    rad = (theta/180)*pi
    return rad

ang = 90
angr = convert(ang)
print (angr)
```

Também é possível usar a função map (vista na aula anterior) em conjunto com uma função definida pelo usuário. Por exemplo, seja *r* um vetor contendo vários ângulos em graus. "rrad = list(map(convert,r))" gera uma lista *rrad* em que todos os ângulos foram convertidos para radiano.

Exercício 2.9: Fatorial

Defina uma função que calcula o fatorial de um inteiro n. Em seguida, escreva um programa que pede ao usuário um número n, calcula $n!$ e em seguida imprime o resultado.

Exercício 2.10: Mapeando funções definidas pelo usuário

Crie o vetor *a*= [2,3,4,5]. Em seguida, crie um vetor *b* em que cada elemento é o fatorial de um um elemento de *a*. (Dica: Use a função "map" em conjunto com a função "fatorial" definida no exercício anterior para realizar esta tarefa em uma linha.)

Problemas opcionais:

Exercício 2.11: Média aritmética e geométrica

Escreva um programa que calcula a média aritmética e a geométrica dos valores que estão no arquivo "values.txt". A seguinte fórmula pode ser usada para calcular a média geométrica (ver página 63 do livro do Mark Newman):

$$\bar{x} = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\ln x_i\right). \tag{1}$$

Dica: Utilize funções disponíveis no numpy para facilitar o cálculo.

Exercício 2.12: Números primos

(a) Crie uma função que determina os fatores primos de um número (por exemplo: os fatores primos de 20 são 2, 2 e 5). (b) Utilize esta rotina para encontrar todos os números primos menores que 10000 (lembre que números primos só possuem um fator primo).

Exercício 2.13: Recorrência

Uma característica útil de funções definidas pelo usuário é a *recursão*, a habilidade de uma função de chamar a si mesma. Por exemplo, considere a seguinte definição do fatorial $n!$ de um inteiro positivo n:

$$n! = \begin{cases} 1 & \text{if } n = 1, \\ n \times (n-1)! & \text{if } n > 1. \end{cases}$$

Esta constitui uma definição completa do fatorial, que nos permite calcular o valor de $n!$ para qualquer positivo inteiro. Nós podemos empregar esta definição diretamente para criar uma função que calcula fatoriais no Python, desta maneira:

```
def factorial(n):
    if n==1:
        return 1
    else:
        return n*factorial(n-1)
```

Note como, se n não for igual a 1, a função chama a si própria para calcular o fatorial de $n-1$. Isto é recursão. Se nós agora escrevermos "`print(factorial(5))`", o computador irá corretamente imprimir a resposta 120.

Nós encontramos os números de Catalan C_n num problema anterior. Com uma pequena reorganização, a definição dada lá pode ser reescrita na forma

$$C_n = \begin{cases} 1 & \text{if } n = 0, \\ \dfrac{4n-2}{n+1} C_{n-1} & \text{if } n > 0. \end{cases}$$

Escreva uma função em Python, usando recursão, que calcula C_n. Use sua função para calcular e imprimir C_{100}.

Exercício 2.14: A fórmula da massa semi-empírica

Em física nuclear, a fórmula da massa semi-empírica é uma fórmula para calcular, aproximadamente, a energia de ligação nuclear B de um núcleo atômico com número atômico Z e número de massa A:

$$B = a_1 A - a_2 A^{2/3} - a_3 \frac{Z^2}{A^{1/3}} - a_4 \frac{(A-2Z)^2}{A} + \frac{a_5}{A^{1/2}},$$

onde, em unidades de milhões de eletron volts, as constantes são $a_1 = 15.67$, $a_2 = 17.23$, $a_3 = 0.75$, $a_4 = 93.2$, e

$$a_5 = \begin{cases} 0 & \text{se } A \text{ for ímpar,} \\ 12.0 & \text{se } A \text{ e } Z \text{ forem ambos pares,} \\ -12.0 & \text{se } A \text{ for par e } Z \text{ for ímpar.} \end{cases}$$

a) Escreva um programa que recebe como input os valores de A e Z e imprime a energia de ligação para o átomo correspondente. Use o seu programa para acha a energia de ligação de um átomo com $A = 58$ e $Z = 28$. (Dica: a resposta correta está em torno de 490 MeV.)

b) Modifique seu programa para que ele imprima não a energia total de ligação B, mas sim a energia de ligação por núcleon, que vale B/A.

c) Agora modifique seu programa para que ele receba como input um único valor do número atômico Z e então percorra todos os valores de A entre $A = Z$ e $A = 3Z$, para encontrar aquele que tem a maior energia de ligação por núcleon. Este é o núcleo mais estável para um dado número atômico. Faça seu programa escrever, para este núcleo mais estável, o valor de A e o valor da energia de ligação por núcleon.

d) Modifique novamente seu programa de modo que, ao invés de receber Z como input, ele percorra todos os valores de Z entre 1 e 100 e imprima o valor mais estável de A para cada caso. Para qual valor de Z ocorre a máxima energia de ligação por núcleon? (A resposta correta, na vida real, é $Z = 28$, o níquel. Você deve achar que a fórmula da massa semi-empírica encontra uma resposta próxima, mas não exatamente correta.)