Ciências de Dados e Big Data

Agenda

- Listas
- Laço *While*
- Laço *For*

Ciências de Dados e Big Data
Programando em Python

Listas

O Python conhece vários tipos de dados *compostos (Conjunto de Dados)* , usados para agrupar outros valores. A mais versátil é a *lista* , que pode ser escrita como uma lista de valores separados por vírgula (itens) entre colchetes. As listas podem conter itens de tipos diferentes, mas geralmente os itens têm o mesmo tipo.

Exercício

```
impares = [1, 3, 5, 7, 9]
impares
[1, 3, 5, 7, 9]
impares[0]
1
impares[1]
3
impares[-1]
9
impares[1:]
[3,5,7,9]
impares[3:]
[9]
```

Exercício

```
impares[3:]
[7,9]
impares[1:4]
[3,5,7]
impares[2:5]
[5,7, 9]
impares[:]
[1, 3, 5, 7, 9]
impares.append(13)
impares
[1, 3, 5, 7, 9, 13]
```

```
impares[5]
13
impares[5]=11
impares
[1, 3, 5, 7, 9, 11]
impares=impares+[15,19,13,17]
impares
[1, 3, 5, 7, 9, 15, 19, 13, 17]
impares.sort()
impares
[1, 3, 5, 7, 9, 13, 15, 17, 19]
```

Ciências de Dados e Big Data
Programando em Python

Exercício

```
letras = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
letras
['a', 'b', 'c', 'd', 'e', 'f', 'g']
letras[2:5] = ['C', 'D', 'E']
letras
['a', 'b', 'C', 'D', 'E', 'f', 'g']
letras[2:5] = []
letras
['a', 'b', 'f', 'g']
letras[:] = []
letras
[]
```

Exercício

```
letras=["a","e","i","o","u"]
len(letras)
5
letras.insert(1,"b")
letras
['a', 'b', 'e', 'i', 'o', 'u']
>len(letras)
6
letras.insert(3,"b")
letras
['a', 'b', 'e', 'b', 'i', 'o', 'u']
letras.remove("b")
letras
['a', 'e', 'b', 'i', 'o', 'u’]
letras.remove("b")
letras
['a', 'e', 'i', 'o', 'u']
```

ESTRUTURA DE REPETIÇÃO (LAÇOS E LOOPS)

Muitos problemas exigem que determinado conjunto de instruções sejam executadas várias vezes ***WHILE*** (ENQUANTO) uma condição seja satisfeita, formando uma LAÇO DE REPETIÇÃO.

WHILE <CONDIÇÃO VERDADEIRA> **:**

<INSTRUÇÃO 1>

<INSTRUÇÃO 2>

Exemplo : Somar os 10 primeiros números inteiros. Soma=1+2+3+4+5+6+7+8+9+10

```
count=1
soma=0
while (count<=10) :
    soma=soma+count
    count=count+1
print(soma)
```

Ciências de Dados e Big Data
Programando em Python

ESTRUTURA DE REPETIÇÃO (LAÇOS E LOOPS)

Exemplo: Soma os primeiros N números inteiro

PROGRAMA: SomaInteiros

OBJETIVO: Calcular a SOMA dos N primeiros números inteiros

ENTRADA: N (Quantidade de Números a Somas)

SAÍDA:

```
N=int(input("Entre com a qtde de números:"))
count=1
soma=0
while (count<=N) :
        soma=soma+count
        count=count+1
print(soma)
```

Ciências de Dados e Big Data
Programando em Python

ESTRUTURA DE REPETIÇÃO (LAÇOS E LOOPS)

Exemplo: Uma loja coleta diariamente o valor da compra de cada cliente para calcular o valor do Ticket Médio de Venda. Faça o programa que calcule o Ticket Médio, com o usuário informando quantas vendas ele coletou e o valor de cada venda.

```
N=int(input("Entre com a qtde de VENDAS:"))
count=1
soma=0
while (count<=N) :
   vlVenda=float(input("Entre com o valor da venda:"))
   soma=soma+vlVenda
   count=count+1

media=soma/N
print(media)
```

Listas - Manipulando listas com programação

Exemplo 1 – Imprime todos os elementos de uma lista

```
lista = ["A", "B", "C", "D", "E", "F", "G","H","I", "J" ]
i=0
while (i<len(lista)):
      print ("Item ",i," : ",lista[i])
      i=i+1
print("Fim!!")
```

Exemplo 2 – Contanto um determinado item em uma lista e imprimindo o resultado

```
lista = ["A", "B", "C", "A", "E", "A", "C","H","A", "C"]
i=0
elemento="C‟
contador=0
while (i<len(lista)):
       if (lista[i]==elemento):
              contador=contador+1;
       i=i+1
print("Foram encontrado(s) ", contador, " elemento(s) igual a ", elemento)
```

Exemplo 3 – Soma e calcula a média de uma lista

```
lista = [10.5, 9.98, 11.0, 10.3, 9.9, 9.8, 10.3,11.0,9.7, 10.0]
soma=0
i=0
while (i<len(lista)):
        soma=soma+lista[i]
        i=i+1
media=soma/i
print("soma=",soma)
print("Total de itens=",i)
print("media=",media)
```

Exemplo 4 – Determina o Maior e o Menor número de uma lista (Máximo e Mínimo)

```
lista = [10.5, 9.98, 11.0, 10.3, 9.9, 9.8, 10.3,13.0,9.7, 10.0]
maior=-1
menor=1000
i=0
while (i<len(lista)):
    if (lista[i]>maior) :
            maior=lista[i]
    if (lista[i]<menor) :
            menor=lista[i]
    i=i+1
print("Maior=",maior)
print("Menor=",menor)
```

Exemplo 5 – Determina o Maior e o Menor número de uma lista (Máximo e Mínimo)

```
lista = [10.5, 9.98, 11.0, 10.3, 9.9, 9.8, 10.3,13.0,9.7, 10.0]
lista.sort()#ordena lista em ordem crescente
menor=lista[0]#pega o primeiro elemento da lista
maior=lista[len(lista)-1]#pega o ultimo elemento da lista
print("Maior=",maior)
print("Menor=",menor)
```

Exemplo 6 – Acrescendo 10% no valor de cada item da lista

```
lista = [10.5, 9.98, 11.0, 10.3, 9.9, 9.8, 10.3,11.0,9.7, 10.0]
i=0
while (i<len(lista)):
    lista[i]=lista[i]+0.10*lista[i]
    i=i+1
print(lista[:])
```

Exemplo 7 – Criando uma nova lista a partir da lista acrescendo 10% no valor de cada item da lista

```
lista = [10.5, 9.98, 11.0, 10.3, 9.9, 9.8, 10.3,11.0,9.7, 10.0]
lista2=[]#Cria uma lista vazia
i=0
while (i<len(lista)):
    lista2.append(lista[i]+0.10*lista[i])
    i=i+1
print(lista2[:])
```

Exemplo 8 – Criando uma lista de 10 PREÇOS COLETADOS a partir do input do usuário, e calculando o PREÇO MÉDIO, o Maior e o Menor preço

```
lista=[]#Cria uma lista vazia
i=0
soma=0
while (i<10):
  preco=float(input("Entre com o preco :"))
  lista.append(preco)
  soma=soma+preco
  i=i+1
media=soma/len(lista)
lista.sort()
menor=lista[0]
maior=lista[len(lista)-1]
print(media)
print(maior)
print(menor)
```

ESTRUTURA DE REPETIÇÃO (LAÇOS E LOOPS)

Como já foi visto, muitos problemas exigem que determinado conjunto de instruções sejam executadas várias vezes. A instrução ***WHILE*** (ENQUANTO) permite isto, gerando um LAÇO DE REPETIÇÃO até que uma condição seja satisfeita.
Uma outra instrução que permite criar um laço de repetição é a instrução ***FOR***, que executa um número fixo de repetições.

No exemplo abaixo LAÇO DE REPETIÇÃO irá ser executado exatamente 10 vezes.

```
for i in range(10):
  print(i)
```

ESTRUTURA DE REPETIÇÃO utilizando FOR

Exercício 1

Faça um programa que gere uma lista com os N primeiros pares

```
pares=[]
N=int(input("Entre com a qtde de números pares que deseja gerar: "))
for i in range(N):
  par=i*2
  pares.append(par)
print(pares)
```

ESTRUTURA DE REPETIÇÃO utilizando FOR

Exercício 2

Faça um programa que leia dois conjuntos de números inteiros distintos e imprima a interseção destes dois conjuntos (os números presentes em ambos os conjuntos).

Exemplo:

Primeiro conjunto: 1 2 3 4 5
Segundo conjunto: 2 5 7 1 9 18
Resultado: 1 2 5

ESTRUTURA DE REPETIÇÃO utilizando FOR

Exercício 2 - RESPOSTA

```
entrada1=input("Entre com a primeira lista com os elementos separados por espaço")
entrada2=input("Entre com a segunda lista com os elementos separados por espaço")
lista1=entrada1.split()
lista2=entrada2.split()
print(lista1)
print(lista2)
lista3=[]
for i in range(len(lista1)):
  num=lista1[i]
  for j in range(len(lista2)):
    if (lista2[j]==num):
      lista3.append(num)

print(lista3)
```

ESTRUTURA DE REPETIÇÃO utilizando FOR

Exercício 3

Desenvolva um gerador de tabuada, capaz de gerar a tabuada de qualquer número inteiro entre 1 a 10.
O usuário deve informar de qual numero ele deseja ver a tabuada. A saída deve ser conforme o exemplo abaixo:

Tabuada do 5:
5 X 1 = 5
5 X 2 = 10
...
5 X 10 = 50

Ciências de Dados e Big Data

Dúvidas????

Certezas????

Obrigado!!!!