

INF100 – Introdução à Programação I
Roteiro da Aula Prática 07 – 05 de novembro de 2020
Comandos repetitivos while, for e arranjos
Valor: 5 pontos

Introdução

O uso de arranjos permite armazenar vários valores associados a uma variável estruturada. O acesso a cada valor individual é feito usando o nome da variável e o índice, ou a posição, dentro da estrutura do arranjo. A figura abaixo mostra a representação de um arranjo de nome nota de dimensão 6.

nota	90	63	58	95	86	
	0	1	2	3	4	← índice (ou posição)

A atribuição de valores é individual e funciona da mesma forma que a atribuição para variáveis comuns (ou escalares) com as quais já lidamos há algum tempo. A diferença é que a posição (ou índice) na qual o valor será armazenado deve ser especificada entre colchetes. Ex:

```
nota[0] = 90      # atribui o valor 90 para a primeira posição (zero)
nota[1] = 63      # atribui o valor 63 para a segunda posição (1)
pos = 4
nota[pos] = 86    # atribui o valor 86 para a posição 4 (quinta posição)
nota[pos-1] = 95  # atribui o valor 95 para a posição "pos-1" ou 3 (quarta posição)
nota[2] = nota[1] - 5 # lê o valor da posição 1 (63) e calcula a expressão (63-5) e o
                   # resultado (58) é atribuído para a posição 2
```

Assim, os comandos acima definiriam os valores dos elementos do arranjo nota conforme mostrado na figura acima.

O comando repetitivo for é ideal para a manipulação de arranjos pois ocorre um casamento natural dos índices limites da faixa (range) de execução do comando for com os índices de um arranjo de dimensão N.

Os índices de um arranjo de dimensão N variam de 0 a N-1 da mesma forma que um for i in range (0, N) também executará para valores de i de 0 a N-1. Desta forma, com um comando for i in range (0, N) é possível percorrer todo o arranjo, posição por posição (apontada pela variável i – que controla as repetições do comando for).

Assim, para imprimir os elementos do arranjo nota (um elemento por linha) bastaria usar um comando repetitivo for para percorrer todas as posições do arranjo (de 0 a 4), apontadas (ou indexadas) pela variável (i) de controle do for :

```
for i in range(0, 5) :
    print( nota[i] )
```

O resultado da execução seria o seguinte:

```
90.0
70.0
65.0
95.0
86.0
>>>
```

Note que os valores foram impressos como sendo números de ponto flutuante. Por que?

Isso ocorreu devido a forma padrão de criação de um arranjo em Python. Mas, como criar um arranjo em Python?

Primeiramente é necessário importar a biblioteca numérica numpy. Na primeira semana de aula, a prática foi a instalação do ambiente de programação IDLE-Python e das bibliotecas numérica (numpy) e gráfica (PIL). A partir de agora, iremos usá-las (começando com a numpy). O uso de uma biblioteca é feito com a diretiva de importação (import):

```
import numpy as np # importa a biblioteca numpy e define um apelido (np)
```

Uma das formas de criar um arranjo é usando a função empty, como mostrado abaixo:

```
nota = np.empty(5) # cria um arranjo de nome nota com 5 posições
```

Quando não se especifica o tipo, o padrão da linguagem Python é definir como float. Se fosse um arranjo de inteiros, a definição deveria ser:

```
nota = np.empty(5, dtype=int) # cria um arranjo de nome nota com 5 posições
```

Caso desejássemos classificar as notas previamente armazenadas no arranjo e imprimir o conceito associado, considerando as faixas:

- 90-100: conceito A
- 75-89: conceito B
- 60-74: conceito C
- 0-59: conceito R.

Para resolver tal problema, bastaria percorrer o arranjo e comparar cada uma das notas com as faixas e imprimir o resultado. Note a manipulação dos índices para mostrar o resultado para o usuário. Para os seres humanos normais, a nota na primeira posição é a primeira nota. Apenas para nós, conhecedores de Python, é que faz sentido a primeira posição ser a posição 0.

```
for i in range(0, 5) :
    # imprime a primeira parte da mensagem
    print('%da nota = %d, conceito ' % (i+1, nota[i]), end='')
    # faz a classificação da nota[i] e imprime o conceito correspondente
    if nota[i] >= 90 :
        print('A. ')
    elif nota[i] >= 75 :
        print('B. ')
    elif nota[i] >= 60 :
        print('C. ')
    else :
        print('R. ')
```

Observe também que a impressão da posição e valor da nota é sempre feita. A impressão condicional é apenas da informação do conceito (A, B, C ou R). A saída deste trecho de código é mostrada a seguir.

```
1a nota = 90, conceito A.  
2a nota = 63, conceito C.  
3a nota = 58, conceito R.  
4a nota = 95, conceito A.  
5a nota = 86, conceito B.  
>>>
```

Importante lembrar, mais uma vez, que apesar do arranjo permitir armazenar vários valores associados a um mesmo nome de variável (nota, nos exemplos anteriores), a manipulação dos valores, em geral, é individual, ou seja, feita com uma única posição *i*.

Veja novamente os comandos de atribuição e as comparações que usaram sempre `nota[indice]`, onde “índice” na maioria das vezes foi a variável de nome *i*. Mas também foram usados números inteiros, a variável de nome *pos* e a expressão “*pos-1*”, que sempre expressaram um valor inteiro no intervalo 0 a 4, que são os índices válidos para o arranjo de tamanho 5.

Um professor bonzinho que quisesse dar um bônus de 5 pontos ao final do semestre para cada um dos 5 alunos poderia fazê-lo executando o seguinte comando:

```
for i in range (0, 5) :  
    nota[i] = nota[i] + 5
```

Na verdade, o código mais correto deveria incluir um teste para verificar se a nota não ultrapassaria a nota máxima 100:

```
for i in range (0, 5) :  
    nota[i] = nota[i] + 5  
    if nota[i] > 100 :  
        nota[i] = 100
```

Ou alternativamente, fazendo o teste antes de somar o bônus:

```
for i in range (0, 5) :  
    if nota[i] < 95 :  
        nota[i] = nota[i] + 5  
    else :  
        nota[i] = 100
```

Uma última informação necessária para entender o código da prática é sobre uma outra forma de criação de um arranjo usando valores aleatórios. Neste caso, cada posição será preenchida com um valor aleatório em um determinado intervalo. Isso pode ser feito utilizando duas funções da biblioteca `numpy` (chamada de `np` no nosso código):

```
# iniciar gerador de números (pseudo) aleatórios para que os valores sejam  
# OS MESMOS sempre que o programa for executado.  
np.random.seed( 0 )  
  
# gerar um arranjo de nome nota com números inteiros entre 0 e 100  
nota = np.random.randint( 0, 101, 5 )
```

Na função `randint` os dois primeiros parâmetros se referem a faixa na qual os números inteiros gerados estarão limitados, incluído o limite inferior (0) e excluído o limite superior (101). E o terceiro parâmetro se refere à dimensão do arranjo (5).

Já a função `random.seed(n)` define a semente a ser usada para a geração de números pseudo aleatórios. São chamados de pseudo (falso) aleatórios porque se usarmos o mesmo valor de semente, o programa gera sempre os mesmos valores.

Instruções


Nesta prática DEVE ser utilizado o comando repetitivo *for* para manipulação/utilização dos elementos do arranjo.


Nome do arquivo a ser entregue: **p07.py**

Importante: Como qualquer outra prática de INF100 você deve:

1. Criar o cabeçalho obrigatório.
2. Após finalizar o cabeçalho salve o arquivo com o nome correto
3. Leia as instruções até o final e, após finalizar sua leitura, inicie sua programação.

Obs.: Recomenda-se salvar o arquivo com certa frequência para não perder a digitação já feita em caso de uma falha na rede elétrica.

 A saída do programa deve obedecer à formatação **exata** mostrada nos exemplos acima.

 Não esqueça de preencher o cabeçalho com seus dados e uma breve descrição do programa.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p07.py**) através do sistema do LBI.

Questão a ser Resolvida

- 1) No servidor do LBI há um arquivo de nome `p07.py` com código (ainda incompleto) para gerar `N` números inteiros de maneira aleatória e calcular a média de todos os valores.

Complete o programa do arquivo `p07.py` disponível no servidor do LBI para que ele execute as seguintes tarefas:

- a) Criar um arranjo de 60.000 posições, preenchendo as posições com números inteiros aleatórios no intervalo `linf` e `lsup` (JÁ ESTÁ IMPLEMENTADO)

Atribua os dois valores acima para as variáveis `linf` e `lsup` no início do código:

**`linf = 75733`
`lsup = 103124`**

- b) Informar o valor do maior elemento encontrado e quantas vezes este maior valor apareceu. A tarefa de percorrer o arranjo e determinar o maior valor e o número de vezes que ele aparece JÁ ESTÁ PRONTA. Falta apenas o comando de impressão.

Falta também calcular o somatório, para depois calcular a média (item d).

d) Informar a média inteira dos valores, **usando %.0f no comando de impressão da média.**

e) Imprimir a média, o maior valor e o número de vezes que este maior valor aparece (veja o exemplo de execução).

f) Após ter feito todas as tarefas acima, o programa deverá executar de maneira repetitiva, usando a estrutura “while True:” (que já está escrita no código) para:

- Ler um valor do teclado e, de acordo, com o valor lido executar a seguinte tarefa:

1) terminar a execução do programa, quando o valor fornecido for 0 (zero); ou

2) informar quantas vezes o número aparece no arranjo, quando ele está na faixa [linf, lsup] (conforme valores definidos no item “a”) e voltar para ler outro valor; ou


3) informar que o número não pertence à faixa [linf, lsup] (conforme valores definidos no item “a”), exceto para o número 0 (zero), e voltar para ler outro valor

A seguir é apresentado um exemplo de execução do programa completo.

```
Criando um arranjo com 60.000 posições
Valores no intervalo 75733 a 103124.

A média dos valores é 89437
O maior valor 103124 aparece 1 vez(es).

Entre com um valor ente 75733 e 103124 ou 0 para terminar: 103124
O valor 103124 aparece 1 vez(es).
Entre com um valor ente 75733 e 103124 ou 0 para terminar: 103125
Este valor está fora da faixa de valores gerados.
Entre com um valor ente 75733 e 103124 ou 0 para terminar: 102000
O valor 102000 aparece 1 vez(es).
Entre com um valor ente 75733 e 103124 ou 0 para terminar: 75732
Este valor está fora da faixa de valores gerados.
Entre com um valor ente 75733 e 103124 ou 0 para terminar: 0
>>>
```

 A saída do seu programa (em ambas as questões) deve obedecer à formatação **exata** mostrada nos exemplos.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p07.py**) através do sistema do LBI.

A entrega deverá ser feita até às 20h20 (1h50 após o início da aula prática)