Inatel
Instituto Nacional de Telecomunicações

AC-309 – Atividades Complementares

(Engenharias Biomédica e de Telecomunicações)

3. Estruturas de Controle de Fluxo e Funções

Prof. Evandro Luís Brandão Gomes

Listas de Dados

Lista

É um conjunto sequencial de valores, onde cada valor é identificado através de um índice. O primeiro valor tem índice 0.

Sintaxe:

Nome_Lista= [valor1, valor2, ..., valorN]

Uma lista pode ter valores de qualquer tipo, incluindo outras listas.

Exemplo:

L = [3,'abacate',9.7,[5,6,3],"Python",(3,'j')]

```
print(L[2])
9.7
print(L[3])
[5,6,3]

print(L[3][1])
6
```

Listas de Dados

Para alterar um elemento da lista, basta fazer uma atribuição de valor através do índice. O valor existente será substituído pelo novo valor.

Exemplo:

```
L = [3,'abacate',9.7,[5,6,3],"Python",(3,'j')]
...
...

L[3]= 'morango'
print(L)
L = [3 , 'abacate' , 9.7 , 'morango', "Python" , (3 , 'j')]
```

Listas de Dados

Função	Descrição	Exemplo
len	retorna o tamanho da lista.	L = [1, 2, 3, 4] len(L) → 4
min	retorna o menor valor da lista.	L = [10, 40, 30, 20] min(L) → 10
max	retorna o maior valor da lista.	L = [10, 40, 30, 20] max(L) → 40
sum	retorna soma dos elementos da lista.	L = [10, 20, 30] sum(L) → 60
append	adiciona um novo valor na no final da lista.	L = [1, 2, 3] L.append(100) L → [1, 2, 3, 100]
extend	insere uma lista no final de outra lista.	L = [0, 1, 2] L.extend([3, 4, 5]) L → [0, 1, 2, 3, 4, 5]
del	remove um elemento da lista, dado seu índice.	L = [1,2,3,4] del L[1] L→ [1, 3, 4]
in	verifica se um valor pertence à lista.	L = [1, 2 , 3, 4] 3 in L → True
sort()	ordena em ordem crescente	L = [3, 5, 2, 4, 1, 0] L.sort() L → [0, 1, 2, 3, 4, 5]
reverse()	inverte os elementos de uma lista.	L = [0, 1, 2, 3, 4, 5] L.reverse() L → [5, 4, 3, 2, 1, 0]

Listas de Dados

Operações com listas:

Concatenação (+)

```
a = [0,1,2]
b = [3,4,5]
c = a + b
print(c)
```

```
[0, 1, 2, 3, 4, 5]
```

Repetição (*)

```
L = [1,2]
R = L * 4
print(R)
```

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

Listas de Dados

Criação de listas com range()

A função **range()** define um intervalo de valores inteiros. Associada ao tipo **list()**, cria uma lista com os valores do intervalo.

A função range() pode ter de 1 a 3 parâmetros:

range(n) → gera um intervalo de **0 a n-1**

range(i , n) → gera um intervalo de **i a n-1**

range(i , n, p) → gera um intervalo de **i a n-1** com intervalo p entre os números

Listas de Dados

Criação de listas com range()

Exemplos:

```
L1 = list(range(5))
print(L1)
[0, 1, 2, 3, 4]
L2 = list(range(3,8))
print(L2)
[3, 4, 5, 6, 7]
L3 = list(range(2,11,3))
print(L3)
[2, 5, 8]
```

Listas de Dados

Exercícios:

1 –Dada a lista L= [5, 7, 2, 9, 4,1, 3], escreva um programa que imprima as seguintes informações:

a) tamanho da lista.
b) maior valor da lista.
c) menor valor da lista.
d) soma de todos os elementos da lista.
e) lista em ordem crescente.
f) lista em ordem decrescente.

2 –Gere uma lista de contendo os múltiplos de 3 entre 1 e 50.

Estruturas de Decisão

Em Python temos as seguintes estruturas de decisão:

Estrutura if

O comando **if** é utilizado quando para decidir se um trecho do programa deve ou não ser executado.

Sintaxe: *if condição:*
Bloco de comandos

Exemplo:

```
valor = int(input("Qual sua idade?"))
if valor < 18:
    print("Você ainda não pode dirigir!")
```

Estruturas de Decisão

Estrutura if..else

Nesta estrutura, um trecho de código será executado se a condição for verdadeira e outro se a condição for falsa.

Sintaxe:

if condição:
    Bloco de comandos para condição verdadeira
else:
    Bloco de comandos para condição falsa

Exemplo:

```
valor = int(input("Qual sua idade? "))
if valor < 18:
    print("Você ainda não pode dirigir!")
else:
    print("Você é o cara!")
```

Estruturas de Decisão

Comando if..elif..else

Se houver diversas condições, cada uma associada a um trecho de código, utiliza-se o **elif.**

Sintaxe:

```
if condição1:
    Bloco de comandos 1
elif condição2:
    Bloco de comandos 2
elif condição3:
    Bloco de comandos 3
.....
else:
    Bloco de comandos
```

Estruturas de Decisão

Exemplo:

```
valor = int(input("Qual sua idade? "))
if valor < 6:
    print("Que coisa fofa!")
elif valor < 18:
    print("Você ainda não pode dirigir!")
elif valor > 60:
    print("Você está na melhor idade!")
else:
    print("Você é o cara!")
```

Estruturas de Decisão

Exercício:

1 – Faça um programa que leia 2 notas de um aluno, calcule a média e imprima o conceito aprovado (média superior a 60), em exame (média entre 30 e 60) ou reprovado (média inferior a 30).

Estruturas de Repetição

A Estrutura de repetição é utilizada para executar uma mesma sequência de comandos várias vezes. A repetição está associada ou a uma condição, que indica se deve continuar ou não a repetição,ou a uma sequência de valores, que determina quantas vezes a sequência deve ser repetida.

Laço while

No **while**, o trecho de código da repetição está associado a uma condição. Enquanto a condição tiver valor verdadeiro, o trecho é executado. Quando a condição passa a ter valor falso, a repetição termina.

Sintaxe:

```
while condição:
    Bloco de comandos
```

Estruturas de Repetição

Exemplos:

```
senha = "54321"
leitura =" "
while (leitura != senha):
   leitura = input("Digite a senha: ")
   if leitura == senha :
       print('Acesso liberado ')
   else:
       print('Senha incorreta. Tente novamente')
```

```
contador = 0
somador = 0
while contador < 5:
    contador = contador + 1
    valor = float(input('Digite o '+str(contador)+'° valor: '))
    somador = somador + valor
print('Soma = ', somador)
```

Estruturas de Repetição

Laço for

O **for** pode ser utilizado com uma sequência numérica (gerada com o comando range) ou associado a uma lista. O trecho de código da repetição é executado para cada valor da sequência numérica ou da lista.

Sintaxe:

```
for variável in range(início, limite, passo):
    Bloco de comandos
```

ou

```
for variável in lista:
    Bloco de comandos
```

Estruturas de Repetição

Exemplos:

1. Encontrar a soma S = 1+4+7+10+13+16+19

```
S=0
for x in range(1,20,3):
    S = S+x
print('Soma = ',S)
```

2. As notas de um aluno estão armazenadas em uma lista. Calcular a média dessas notas.

```
Lista_notas= [3.4,6.6,8,9,10,9.5,8.8,4.3]
soma=0
for nota in Lista_notas:
    soma = soma+nota
média = soma/len(Lista_notas)
print('Média = ', média)
```

Estruturas de Repetição

Exercícios:

1 – Escreva um programa para encontrar a soma S = 3 + 6 + 9 +…. + 333.

2 – Escreva um programa que leia 10 notas e informe a média dos alunos.

3– Escreva um programa que leia um número de 1 a 10, e mostre a tabuada desse número.

Funções

Funções são pequenos trechos de código reutilizáveis. Elas permitem dar um nome a um bloco de comandos e executar esse bloco, a partir de qualquer lugar do programa.

Como definir uma função

Funções são definidas usando a palavra-chave **def**,conforme sintaxe a seguir:

```
def nome_função (definição dos parâmetros):
    Bloco de comandos da função
```

Obs.: A definição dos parâmetros é opcional.

Funções

Exemplo: Função sem parâmetros

```
def hello():
    print ("Olá Mundo!!!")

Para usar a função, basta chamá-la pelo nome:

>>> hello()
Olá Mundo!!!
```

Exemplo: Função com parâmetros

```
def soma(x,y):
    total = x+y
    return total

#programa principal
s=soma(3,5)
print("soma = ",s)
```

→ Resultado da execução:

```
soma = 8
```

Funções

Valor padrão

É possível definir um valor padrão para os parâmetros da função. Neste caso, quando o valor é omitido na chamada da função, a variável assume o valor padrão.

Exemplo:

```
def calcula_juros(valor, taxa=10):
    juros = valor*taxa/100
    return juros
```

```
>>> calcula_juros(500)
50.0
```

Escopo das Variáveis

Toda variável utilizada dentro de uma função tem escopo local, isto é, ela não será acessível por outras funções ou pelo programa principal. Se houver variável com o mesmo nome fora da função, será uma outra variável, completamente independentes entre si.

Exemplo:

```
def soma(x,y):
    total = x+y
    print("Total soma = ",total)

#programa principal
total = 10
soma(3,5)
print("Total principal = ",total)
```

→ Resultado da execução:

```
Total soma = 8
Total principal = 10
```

Escopo das Variáveis

Para uma variável ser compartilhada entre diversas funções e o programa principal, ela deve ser definida como **variável global.**

Exemplo:

```
def soma(x,y):
    global total
    total = x+y
    print("Total soma = ",total)

#programa principal
global total
total = 10
soma(3,5)
print("Total principal = ",total)
```

→ Resultado da execução:

```
Total soma = 8
Total principal = 8
```

Funções

Exercícios:

1 - Crie uma função para desenhar uma linha, usando o caractere ´_´. O tamanho da linha deve ser definido na chamada da função. Se o tamanho for omitido deve-se fazer a linha com 10 caracteres ´_´

2 - Crie uma função que receba como parâmetro uma lista,com valores de qualquer tipo. A função deve imprimir todos os elementos da lista numerando-os.

3 - Crie uma função que receba como parâmetro uma lista com valores numéricos e retorne a média desses valores.

Lista de Exercícios

Lista de Exercícios 1 – Python

Orientações:

- **Faça cada exercício em um arquivo (.py) diferente;**

- **Ao terminá-los, coloque-os em uma pasta, transforme ela em .zip e envie o arquivo ao professor com o tema: Lista-1-Python – seuNome;**

- **Data Limite para entrega:** **Turma A - dia 23/03/2020 até às 12:00h**
  Turma C - dia 30/03/2020 até às 12:00h

- **Email: evandro@inatel.br;**