Python

Introdução

Tipos de Dados em Python

Um programa python é denominado de **script** e é uma sequencia de definições e comandos. As definições são avaliadas e os comandos são executados pelo interpretador python, chamando de **shell**.

Objetos são os elementos fundamentais que a linguagem manipula. Cada objeto tem um **tipo** que define a variedade de manipulações (operações) que o programa pode fazer com este objeto.
Os tipos podem ser de duas classes: **Escalares** e **não escalares**

- **Escalares**: Objetos indivisiveis ou atómicos.
  1. **int** representa números inteiros.
  2. **float** representa números reais.
  3. **bool** usada para representar valores booleanos True e False.
- **Não escalares**: Objetos que têm uma estrutura interna, por exemplo, **strings**

Uma **Expressões** é a combinação de objetos e operadores. Uma expressão retorna um **valor** de algum **tipo**.

```
# soma: +
10 + 4
# divisao
10/4
# divisao inteira
10//4
# resto da divisao
10%4
# potencia de um numero
10**4
```

```
# comparacao: <, >, <=, >=, !=
```

```
10 < 4
```

```
10 != 4
```

```
# como saber o tipo da expressao
print(type(10.3))
```

```
print(type(3))
print(type('s'))
```

Operadores booleanos

- **a and b**: True, se a e b são verdadeiros e False nos outros casos.
- **a or b**: True, se a e b são True ou algum valor de a ou b são verdadeiros, nos outros casos False.
- **not a**: True se for False e False de for True.

Variáveis e Atribuição

As **variáveis** fornecem uma forma de associar **nomes** com objetos. Pode ser usada qualquer palavra para determinar uma variável com exceção das palavras reservadas da linguagem, tipo *for*, *else*, *int*, *float*, etc.

```
# exemplo de variavel (nome), atribuicao (=) e valor daatribuição
pi = 3.14
r = 4
area = pi*(r**2)
#area
print('area:>',area)
x = [1,2,3,4,5]
#x
print('x=',x)

x1 =['a',2,'aqui',40,'calor']
x1
x1[2]

# atribuicao de valores boolianos
a = 4
b = False
r= a and b
print(r)
```

Visualização de variáveis

In [ ]:

```
# visualizacao de variaveis
nome = 'Rio de Janeiro é considerada a Cidade Maravilhosa'
# visualizar - 1: nome da variavel
nome
```

```
# visualizar - 2: usando o comando print()
print(nome)
# visualizar - 3: usaando comando print()
print('Um fato:',nome)
#visualizacao - 4: usando o comando print()
print(nome,' ...',' sengundo a Revista Fofoca')
```

Tipos de Dados

Um **tipo** de dato é um conjuntos de valores e operações que podem ser realizados com esses valores.

Tipos numéricos

- **Inteiro**: *int*

```
# exemplo de int
n_i = 4
n_i
# visualizar o tipo de dado
type(n_i)
```

- **Real**:*float*

```
# exemplo de float
f = 3.9
print(f)
# visualizar o tipo de dados
type(f)
```

- **Complexo**: *complex*

```
# exemplo de numero complexo
nc1 = 3+4j
print(nc1)
type(nc1)
```

Strings

Uma **string** é uma sequência de caracteres fechados entre aspas simples ou duplas.

```
# exemplo de string
cidade1 = 'Rio de Janeiro'
cidade2 = "Niterói"
print(cidade1)
print(cidade2)
# outra forma de apresentar?
```

```
print(cidade1+' '+cidade2)
```

- **função**: **str(objeto)** - retorna um objeto de tipo *string*

```
# exemplo de uso de str() - numero2string
str(10)
# exemplo de criacao de str()
string_vazia = str()
type(string_vazia)
```

Operações Básicas com Strings

- **Concatenação de strings**:+

```
# exemplo de concatenacao
print(cidade1+cidade2)
# outra forma?
print(' ..',cidade1,' ',cidade2)
# tipo ?
type(cidade1)
```

- **Sequências repetidas de string**: *

```
tres_vezes = cidade1*3 + '-'+str(50)
print(tres_vezes)
```

- Verificar a existência de uma string em outras string. Retorna **True** ou **False**

```
# exemplo
s1 = 'O corona vírus esta solto na cidade do Rio de Janeiro'
s2 = 'cidade'
# exemplo
s3='Araruama'
r1 = s2 in s1
r2 = s3 in s1
print(r2)
```

- Comparação de **strings**: <, >, >=, <=, ==, !=

```
# exemplo de comparacao
print(s1 == s2)
#print(s1 > s2) # why?
```

- **Funções de strings**: *len(),max(),min()*

```
# tamanho da string
l1 = len(s1)
print(l1)
# max()
mx = max(s1)
print(mx)
# min()
mi = min(s1)
print(mi)
```

Índices - indica a posição de um caracter na string

```
# exemplo do tamanho de uma string
tam = len(s1)
p0 = s1[51]
print(p0)
```

Listas

Uma **lista** é uma sequência de zero ou mais objetos. Usa-se os "[]" para indicar uma lista.

Operadores de Listas

```
# exemplo de lista
lista = [1,4,"oi",'Teste']
print(lista)
lista0 = [10, 'a',4-5j,3.1]
print(lista0)
print(len(lista0))
print(lista0[3])
```

Operador	**ação**
[]	lista vazia
append()	adiciona elementos

Operador	**ação**
sort()	ordena os elementos
insert()	insere elementos
pop()	elimina o último elemento
remove()	remove um elemento

```
# exemplo de listas
Lista = []
print('Lista vazia L:',Lista)
Lista.append(34)
print('append(34)-> L:',Lista)
Lista.append(22)
print('append(22)-> L:',Lista)
Lista.append(50)
print('append(50)-> L:',Lista)
Lista.append(-10)
print('append(-10)-> L:',Lista)
Lista.sort()
print('sort(L)-> L:',Lista)
Lista.pop()
print('pop()-> L:',Lista)
Lista.insert(0, 40)
print('insert(0,40)-> L:',Lista)
Lista.insert(1, 55)
print('insert(1,55)-> L:',Lista)
Lista.pop(1)
print('pop(1)-> L:',Lista)
Lista.remove(22)
print('remove(22)-> L:',Lista)
print('O que acontece com os elementos da lista L?')
Lista.remove(55) #
print('remove(55)-> L:',Lista)
```

```
# split(): converte uma string em lista
s = 'Rio de Janeiro está em lockdown por causa da pandemia COVID-19'
r = s.split()
print(r)


# acesar elementos da lista usando o indice
elemento = r[5]
print('Elemento na posição 6:> ',elemento)

# join(): Converte uma lista em string
ns =" ".join(r)
print('ns:',ns)
```

Indexação e Fatiamento de listas¶

```
# list index
print('Indexação de uma Lista')
#print('')
texto =['Python','para','Data Science','curso','introdutório']
t0 = texto[0]
t4 = texto[4]
print('  texto-0:>',t0)
print('  texto-4:>',t4)

# posicao do elemento numa lista
elemento = texto[3]
print('Elemento :>',elemento)
pos = texto.index(elemento)
print('Posição do elemento :>',pos)

# numero de elementos de uma lista
l = len(texto)
print('Número de elemento :>',l)

# sum(). max(); min()
numeros = [23,25,28,45,33,20]
s1 = sum(numeros)
print('Soma :>',s1)
print('Máximo :>',max(numeros))
print('Mínimo :>',min(numeros))

# all() - Retorna TRUE se todos os items de um objeto são "iterable" são
TRUE
```

```
al1 = all([1,1,1,1])
print('AL1:>',al1)
al2 = all([1,1,1,0])
print('AL2:>',al2)
al3 = all([1,1,1,''])
print('AL3:>',al3)
al4 = all([1,1,1,'J'])
print('AL4:>',al4)
# ordena uma lista
#L = ['Z','z','a','A']
L = [5,3,2,-1,1,4]
lo = sorted(L)
print('Lista ordenada :>',lo) # menor -> maior
lr = sorted(L, reverse=True) # maior -> menor
print('Lista ordenada :>',lr)

# ordenando palavras
texto = "COVID-19, Araruama"
T = sorted(texto)
print('T:>',T)

# criar uma lista
items_da_lista = []
total_de_items = int(input("Ingressar o número de items:"))
for i in range(total_de_items):
    item = input(" Item da lista:> ")
    items_da_lista.append(item)
print(f"Lista de items: {items_da_lista}")
print()
print('Número de elementos da lista :> ',len(items_da_lista))

# fatiamento (slices) usando os indices
txt ="O transporte público está um caos"
print('texto :>',txt)
txt1 = txt[3:]
print('txt:>',txt1)
txt2 = txt[3:20]
print('txt:>',txt2)

# 5 valores iniciais
txt5 = txt[:5]
print(txt5)
```

```
# valores
txt4 = txt[12:20]
print(txt4)

# slice(inicio:fim,step)
lv = [0, 10, 20, 30, 40, 50, 60, 70, 80]
#
print('Lista de valores :>',lv)
sl = slice(2,8,2)
print('valores :> ',lv[sl])

# Eliminar um elemento
del lv[0]
print(lv)


# indice negativo
t1 = lv[-1]
print(t1)

# indice negativo
t2 = lv[:-1]
print(t2)
```

Dicionários¶

Um **dicionário** tem zero ou mais entradas. A cada entrada tem associada a uma chave única e um valor **{key:valor}**

- {} : dicionário vazio
- {"nome":"Jorge"} : uma entrada
- {"nome":"Jorge", "Idade":57} : duas entradas
- {"hobbies":["Leitura","viajar"]} : uma entrada e o valor é uma lista

```
# exemplo de dicionario
mesNumero = {'Janeiro':1, 'Fevereiro':2, 'Março':3, 'Abril':4, 'Mai
          o':5, 1:'Janeiro', 2:'Fevereiro', 3:'Março', 4:'Abril'
          , 5:'Maio'}
#
print('O terceiro mês é ' + mesNumero[3])
distancia = mesNumero['Abril'] - mesNumero['Janeiro']
print('De Abril a Janeiro são', distancia, 'mêses de espaço ')

# verificar chaves
key = 'três'
```

```
res = key in chaves
print('Resultado da consulta :>',res)

# verificar valor do key = 2
valor = mesNumero[2]
print('Resultado da consulta :>',valor)


# iteracao sobre as chaves
for key in mesNumero:
    print('Chave:>',key)
```

Funções

```
# def nome_funcao (lista_de_parametros):
#     <sequencia_de_expressoes>

# exemplo de funcao
def soma(a,b):
    return a+b
# uso da funcao
so = soma(7,9)
print('Soma:>',so)
def valorx(a):
    if a > 3:
        x='verdade'
    x='falso'
    return x
#
t = valorx(23) # chama a funcao
#
print('Valor :>',t)
```

Exercícios

Prática dos conteúdos estudados: construindo e operando listas e strings

1. Seja x='cachorro' e y='gato'. Quais são os valores retornados pelas operações:
   - x + y
   - "O" + x + "não gosta de " + y
   - x*6
1. Escreva uma string que contenha nome, endereço, bairro e cidade:
   - Em linhas diferentes

- Na mesma linha

1. Quais das seguintes variáveis são nomes válidos?
   - a) length
   - b) _width
   - c) firstBase
   - d) 2MaisDois
   - e) halt!
1. Que tipo de dados seria mais apropriado para representar os seguintes valores?
   - o número de meses de um ano
   - a área de um círculo
   - o salário mínimo atual
   - a idade do universo
   - seu nome
1. Seja x = 4 e y = 0.5. Escrever os valores das seguintes expressões:
   - a) x + y * 3
   - b) (x + y) * 3
   - c) x ** y
   - d) x % y
   - e) x / 12.0
   - f) x / 6
1. Seja x = 5.33. Escrever os valores das seguintes expressões:
   - a) round(x)
   - b) int(x)

   Têm alguma diferença?
1. Se pode concatenar um valor numérico e uma string?. Apresente exemplos.
1. Definir funções para as operações matemáticas básicas.

Fontes:

Professor: Sergio Serra - UFRRJ

Professor: Jorge Zavaleta - UFRRJ