Universidade do Minho
Licenciatura em Física

Tarefa 4

Professores:	Cristiana Neto, Pedro Oliveira, Vitor Alves
Disciplina:	Linguagens para Computação Numérica
Tema:	Strings, Listas e Tuplos
Data:	Março de 2022

1 Introdução

Nas aulas anteriores foram introduzidos diferentes tipos de dados em Python, tendo sido principalmente utilizados: `int`, `float`, and `str`.

Inteiros e floats são tipos numéricos, o que significa que contêm números. Podemos usar os operadores numéricos como vimos nas aulas anteriores para construir expressões numéricas. O interpretador consegue então avaliar essas expressões para produzir valores numéricos, tornando o Python numa calculadora muito poderosa.

Por sua vez, strings, listas e tuplos são tipos de sequências, uma vez que se comportam como uma sequência - uma coleção ordenada de objetos.

Os tipos de sequência são qualitativamente diferentes dos tipos numéricos porque são tipos de dados compostos - o que significa que são compostos de partes menores.

1.1 String

No caso das strings estas são compostas por strings menores, cada uma contendo um caractere que pode ser um número, uma letra, um espaço ou um símbolo. Em Python, as strings são delimitadas por aspas, tanto duplas (") como simples (') e já têm vindo a ser usadas nas aulas anteriores. São várias as operações que podem ser feitas sobre as strings:

1.1.1 Concatenação (+)

O operador + concatena (junta) strings. É necessário prestar atenção pois o símbolo do operador concatenação é o mesmo que o de soma, porém a soma é feita com tipos numéricos (inteiros ou decimais). Veja os seguintes exemplos:

```
8 + 7     # retorna 15 (soma)
'8' + '7' # retorna '87' (concatenação)
'8' + 7   # ERRADO!
```

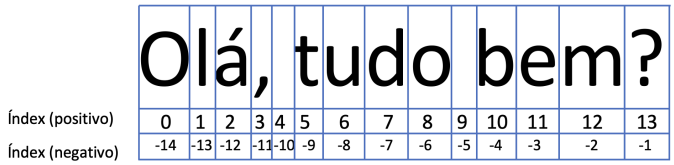
1.1.2 Repetição (*)

O operador * repete strings. Uma string será repetida tantas vezes quanto indicado pelo inteiro que a acompanha na operação. Novamente, é necessário ter atenção. Com números, este operador funciona de forma diferente do que com strings. Veja este comportamento nos seguintes exemplos:

```
3 * 4     # retorna 12 (multiplicação)
3 * '4'   # retorna '444' (repetição)
'3' * '4' # ERRADO!
```

1.1.3 Indexação(string[pos])

As strings são indexadas. Isto significa que podemos obter um caractere da string a partir de uma determinada posição. No exemplo apresentado de seguida, na posição 0 temos o primeiro caractere ("O"), na posição 1 o segundo ("l"), e assim por diante. Na posição 5 temos o sexto caractere, que é "t". Os espaços também são caracteres.




```
texto = "Olá, tudo bem?"
print(texto[5])              #Imprime t
```

1.1.4 Fatiamento (string[start:stop:step])

Com esta operação podemos obter uma string que seja uma cópia de parte de outra. A sintaxe desta operação é a seguinte:

- start: Índice inicial onde começa o fracionamento.
- stop: Índice final onde o corte pára.
- step: Argumento opcional que determina o incremento entre cada índice para fracionamento.

```
texto = "Olá, tudo bem?"
print(texto[:3])             #Imprime "Olá"
print(texto[5:13])           #Imprime "tudo bem"
print(texto[10:])            #Imprime "bem?"
print(texto[-14:-11])        #Imprime "Olá"
print(texto[0:10:2])         #Imprime "Oá uo"
print(texto[::-1])           #Imprime toda a string a começar do fim: "?meb odut ,álO"
```

É necessário perceber que a cópia será feita desde a posição que vem antes dos ":" (INCLUINDO o que estiver nessa posição) até a posição que vem depois dos ":" (EXCLUINDO o que estiver nessa posição).

1.1.5 Métodos

O Python apresenta um conjunto de métodos que podem ser usados nas strings. Os mais comuns são:

- stringVar.count('x') - conta o número de ocurrências de 'x' na stringVar
- stringVar.find('x') - retorna a posição do caractér 'x' na stringVar

- stringVar.lower() - retorna a stringVar em letras minúsculas (isto é temporário)
- stringVar.upper() - retorna a stringVar em letras maiúsculas (isto é temporário)
- stringVar.replace('a', 'b') - substitui todas as ocorrências de a com b na stringVar
- stringVar.split() - divide a string numa lista, em que cada palavra da string se torna um elemento da lista

```
stringVar = 'Hello'
print(stringVar.count('l'))            #retorna 2
print(stringVar.find('H'))             #retorna 0
print(stringVar.lower())               #retorna 'hello'
print(stringVar.upper())               #retorna 'HELLO'
print(stringVar.replace('H','R'))      #retorna 'Rello"
```

1.2 Listas

No caso de listas, estas podem ser compostas por vários elementos, que, por sua vez, podem ser valores de qualquer tipo de dados, incluindo outras listas. As listas são colocadas entre parênteses rectos (`[` e `]`) e os seus elementos são separados por vírgulas. A figura seguinte mostra alguns exemplos de listas:

```
lista_int = [1,2,3]
lista_str = ['azul', 'verde', 'vermelho']
notas = [6.2, 5.7, 8.3, 7.3]
mistureba = [1, 2.0, '3', '4.0', 'cinco', [6]]
lista_vazia = []
```

1.2.1 Operações

Tal como nas strings, também nas listas podemos usar concatenação, repetição, indexação e fatiamento como pode ser observado na figura seguinte. Nesta figura, é também mostrado como substituir um elemento da lista e como verificar o tamanho da lista (função `len`).

```
lista = ['abc', 'def','qwerty']
print(lista + [1, 2, 3])   # retorna ['abc', 'def', 'qwerty', 1, 2, 3]
print(2 * lista)           # ['abc', 'def', 'qwerty', 'abc', 'def', 'qwerty']
print(lista[1])            # retona 'def'
print(lista[1:])           # retorna ['def', 'qwerty']
print(len(lista))          #retona 3

#substituir elemento na lista:
lista[1] = 'novo'
print(lista)               #retorna ['abc', 'novo', 'qwerty']
```

1.2.2 Métodos

O Python oferece um conjunto de métodos que facilitam as operações sobre listas:

Método	Descrição
`append()`	Adiciona um elemento ao fim da lista
`clear()`	Remove todos os elementos da lista
`copy()`	Retorna uma cópia da lista
`count()`	Retorna o número de elementos com o valor especificado
`extend()`	Adiciona elementos de uma lista ao final da lista atual
`index()`	Retorna o index do primeiro elemento com o valor especificado
`insert()`	Adiciona um elemento na posição especificada
`pop()`	Remove o elemento da posição especificada
`remove()`	Remove o primeiro elemento com o valor especificado
`reverse()`	Reverte a ordem da lista
`sort()`	Ordena a lista

O método `join()` tem a capacidade de juntar todos os elementos da lista/tuplo numa string, separados por um caractere definido.

```
lista = ['abc', 'def','qwerty']
lista.append('abab')                #lista passa a ser ['abc', 'def', 'qwerty', 'abab']
print(lista.count('abc'))           # retorna 1
print(lista.index('abc'))           # retorna 0
lista.remove('abab')                #lista passa a ser ['abc', 'def', 'qwerty']
lista.insert(1,'hello')             #lista passa a ser ['abc', 'hello', 'def', 'qwerty']
lista.pop(1)                        #lista passa a ser ['abc', 'def', 'qwerty']
lista.reverse()                     #lista passa a ser ['qwerty', 'def', 'abc']
lista.sort()                        #lista passa a ser ['abc', 'def', 'qwerty']
print((', ').join(lista))           #retorna "abc, def, qwerty"
lista.clear()                       #lista passa a ser []
```

Ao contrário das strings, as listas são exemplos de valores mutáveis. Acabamos de ver que é possível adicionar, remover e modificar um elemento de uma lista qualquer.

1.3 Tuplos

Os tuplos são similares às listas, porém, não podem ser modificados (são imutáveis), ou seja, não se podem colocar, trocar nem retirar elementos de um tuplo. Estes são representadas com parênteses e seus elementos devem ser separados por vírgulas. A figura seguinte mostra alguns exemplos de tuplos:

```
tupla1 = ('eu', 'tu', 'ele')
tupla2 = (1, '1', 1.0)
tupla3 = (5,) # tuplo com um elemento (note a vírgula)
tupla_vazia = ( ) # tuplo vazia
```

As mesmas operações que se aplicam às strings, também se aplicam aos tuplos, como pode ser observado na figura seguinte:

```
tup = ('abc', 'def')
print(tup + (1, 2, 3)) #concatenação -> retorna ('abc', 'def', 1, 2, 3)
print(tup * 3)         #repetição -> ('abc', 2, 'abc', 2, 'abc', 2)
print(tup[1])          #indexação -> retorna 'def'
print(tup[0:1])        #fatiamento -> retorna ('abc')
print(len(tup))        #retorna 2
```

Além disso, também os métodos `count()` e `index()` podem ser usados nos tuplos.

2 Exercícios

Os exercícios desta ficha de trabalho pretendem fornecer aos alunos um conjunto de problemas básicos utilizando strings, listas e tuplos. Pretende-se que apliquem os conhecimentos adquiridos também nas aulas anteriores.

1. Crie um programa que receba o primeiro e último nome de uma pessoa (num só input), e imprima a abreviação do nome. Exemplo:

   - Entrada: Pedro Moura
   - Resultado: Moura, P.

2. Crie um programa que receba um texto separado por underscores e onde todas as letras são maiúsculas, e o converta para um texto separado por espaços e onde apenas a primeira letra é maiúscula. Adicionalmente, diga quantas substituições underscore->espaço foram feitas. Exemplo:

   - Entrada: HOJE_VAI_CHOVER.
   - Resultado: Hoje vai chover. Conversões '_'->' ': 2

3. Um palíndromo é uma sequência de caracteres cuja leitura é idêntica se feita da direita para esquerda ou viceversa. Por exemplo: OSSO e OVO são palíndromos. Em textos mais complexos os espaços são ignorados. A frase "A mala nada na lama"é o exemplo de uma frase palíndroma onde os espaços foram ignorados. Faça um programa que leia uma sequência de caracteres, mostrea e diga se é um palíndromo ou não.

4. Crie um programa onde o utilizador insere uma expressão qualquer que use parênteses e analisa se a expressão colocada está com os parênteses abertos e fechados corretamente.

5. Crie um programa que simule a criação de uma conta (username e password) num determinado site. A password tem de ter um mínimo de 5 caracteres e o username não pode existir na lista de usernames já inseridos. Armazene também todos os utilizadores/passwords numa lista de tuplos.

6. Desenvolva um programa que receba as notas de um aluno em três frequências, que as guarde numa lista e que apresente a média final (pode usar o método `sum()`). Caso a média seja igual ou superior a 10, apresentar a mensagem "APROVADO", caso contrário, pedir ao utilizador a nota do recurso. Caso a nota do recurso seja maior ou igual a 10, apresentar a mensagem "APROVADO", caso contrário, apresentar a mensagem "REPROVADO"

7. Escreva um programa que permita ao utilizador inserir o seu número de telemóvel numa lista de bloqueados, ver a lista, remover um número e limpar a lista. O número só deve ser inserido na lista se ainda não existir, caso contrário, o utilizador deve ser avisado que o seu número já se encontra na lista. É ainda obrigatório que o número inserido contenha o indicativo de Portugal (+351) e 9 digitos após o indicativo. Assim, deve ser apresentado ao utilizador as seguintes opções:

   (a) Adicionar número à lista de bloquados.

   (b) Remover número da lista de bloqueados.

   (c) Limpar a lista de bloqueados.

   (d) Ver lista de bloqueados.

8. Escreva um programa que receba um url e retorne as suas componentes. Indique ainda de o url inserido pertence ao domínio de topo na internet de Portugal (.pt). Exemplo:

    - Entrada: http://www.google.com
    - Resultado: Protocolo: http, subdomínio: www, domínio: google, TLD: com. Este url não pertence ao domínio de topo na internet de Portugal.

9. Escreva um programa que de acordo com o horário fornecido pelo utilizador, retorna um cumprimento adequado para o horário fornecido, ou seja:

    - 00:00 até às 11:59 - "Bom dia"
    - 12:00 até às 17:59 - "Boa tarde"
    - 16:00 até às 23:59 - "Boa noite"

10. Escreva uma função que recebe um tuplo de números e um número, e devolve a posição da primeira ocorrência desse número no tuplo. Caso o número não ocorra no tuplo deverá devolver `falso`. Exemplo:

    - Entrada: (4, 3, 2, 2, 1, 4), 1
    - Resultado: 4

11. Faça um programa que leia dois vetores de 3 posições, que representam forças sobre um ponto no espaço 3D, e retorne a força resultante. Dica: a força resultante é obtida através da soma dos valores das coordenadas correspondentes nos dois vetores: (x1 + x2), (y1 + y2), (z1 + z2).

12. Crie um tuplo preenchido com os 18 clubes da Primeira Liga de Futebol portuguesa, ordenados pela sua classificação. Depois crie um programa capaz de mostrar as seguintes informações, consoante a opção do utilizador:

    (a) Os 5 primeiros clubes.

    (b) Os últimos 4 clubes.

    (c) Equipas ordenadas por ordem alfabética.

    (d) Em que posição está o Vizela.