Título: Funções e métodos integrados de trabalho com listas

CENÁRIO DE APRENDIZAGEM		
Escola:	*Duração (minutos):*	90
Professor(a):	*Idade dos alunos:*	13

Ideia Chave:	Vamos conhecer funções e métodos integrados de trabalho com listas

Tópicos:

- Os alunos aprofundam sua compreensão do uso de vários softwares e políticas.

Objetivos:

- Os alunos serão capazes de projetar e criar programas que utilizam sub-rotinas, estruturas e tipos de dados apropriados, expressões, variáveis e comandos iterativos e condicionais.
- Linguagens de programação gerais são usadas para criar programas.
- Os alunos compreendem as diferentes maneiras de usar simulações e algoritmos de organização passo a passo para resolver problemas.

Resultados:

- Os alunos criam um jogo, aplicativo ou aplicativo móvel mais complexo que resolve um problema particular de um assunto ou tópico específico.
- Os alunos aprendem a delinear a operação de um programa mais complexo em vários padrões e generalizações.

Formas de trabalho:

- trabalho individual
- Trabalho de pares
- trabalho de grupo

Métodos:

- apresentação
- discussão
- exercício interativo

ARTICULAÇÃO

Linha de atuação (duração, minutos)

INTRODUÇÃO

O professor explica e inicia a discussão com os alunos:

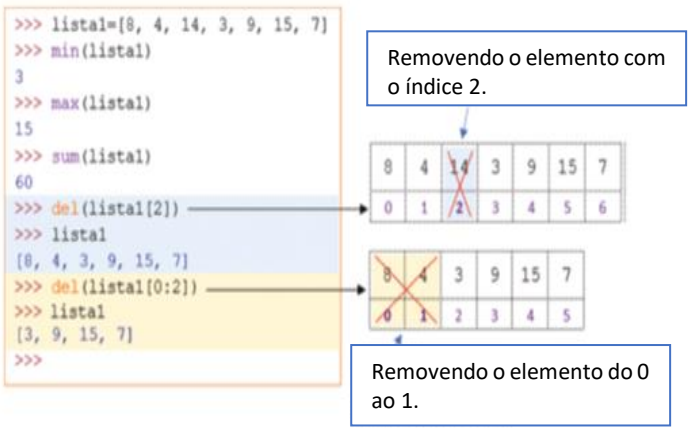
Aprendemos como armazenar dados com características comuns (strings) com a mesma variável e como ter acesso a cada um dos elementos dessa variável. Vamos agora conhecer algumas funções e métodos que podem tornar o nosso trabalho com as listas ainda mais simples.

PARTE PRINCIPAL

Ao usar as strings para armazenar diferentes tipos de dados com atributos comuns, tornámos mais simples aceder e ver dados input. Muitas vezes precisamos de determinar o tamanho (número de elementos) na string, de encontrar o menor ou o maior valor, ou até apagar um elemento. Muitas linguagens de programação oferecem diferentes funções que fazem isso mesmo. Apresentamos na tabela abaixo algumas das funções disponibilizadas no Python.

Função	Descrição
len(a)	Retorna o comprimento da lista
min(a)	Retorna o menor elemento da lista
max(a)	Retorna o maior elemento da lista
sum(a)	Retorna a soma de todos os elementos
del(a[i])	Remove o elemento com o índice
del(a[i:j])	Remove todos os elementos de i a j-1

Aplicar estas funções depende do tipo de dados dos elementos na string.



EXERCÍCIO 1

Cria uma lista com a altura dos alunos de uma turma. Usando funções, encontra e visualiza a altura mais baixa e a mais alta.

```
>>> lista=[156,160,145,147,160,157]
>>> print('The height of the shortest student is',min(lista))
The height of the shortest student is 145
>>> print('The height of the highest student is',max(lista))
The height of the highest student is 160
>>>
```

EXERCÍCIO 2

Cria uma lista com as notas dos alunos num teste. Usando funções, publica a média das classificações.

```
>>> lista=[1,2,3,4,5,1,2,3,4,5]
>>> print(sum(lista)/len(lista))
3.0
>>>
```

```
>>> prazna=[]
>>> print(prazna)
[]
>>>
```

Tal como com as strings, uma lista pode estar vazia. Podemos criar uma lista vazia atribuindo parêntesis retos vazios a uma variável, por exemplo: **a=[]**.

Métodos de construção para trabalho com as listas

Além de operações e de funções, quando trabalhamos com as listas, podemos usar métodos. Estes são sempre atribuídos a um determinado objeto (uma variável ou uma lista) e aparecem sempre depois do objeto a que estão atribuídos. Alguns dos métodos que usamos no Python são:

Nome do método	Maneira de usar	Descrição
append()	a.append(b)	Método adiciona o elemento b ao final da lista a
insert()	a.insert(i,b)	O método insere o elemento b antes do elemento i da lista a
remove()	a.remove(b)	O método exclui o elemento b da lista. Se houver vários b-s, aquele com o índice mais baixo será excluído.
reverse()	a.reverse()	Método faz uma ordem inversa dos elementos na lista a.

sort()	a.sort()	Método classifica a lista do menor para o maior valor.
count()	a.count()	O método retorna o número de repetições do objeto b na lista a como resultado
index()	a.index(b)	Método encontra o índice do elemento b na lista.

Vamos mostrar esses métodos em alguns exemplos.

```
>>> a=[9,4,6,12,16,3]
>>> a.append(13)
>>> a
[9, 4, 6, 12, 16, 3, 13]
>>>
```

Acrescenta um novo elemento com valor 13 e coloca-o no fim da lista

```
>>> a.insert(3,17)
>>> a
[9, 4, 6, 17, 12, 16, 3, 13]
>>>
```

Acrescenta um novo elemento com o valor 17 e coloca-o no índice 3

```
>>> a.remove(16)
>>> a
[9, 4, 6, 17, 12, 3, 13]
>>>
```

Apaga o elemento com o valor 16 de uma lista a. A lista é mais curta por 1 elemento.

```
>>> a.reverse()
>>> a
[13, 3, 12, 17, 6, 4, 9]
>>>
```

Os elementos da lista a são revertidos.

```
>>> a.sort()
>>> a
[3, 4, 6, 9, 12, 13, 17]
>>>
```

Os elementos da lista a são ordenados a partir dos mais pequenos para os mais altos.

```
>>> a.append(6)
>>> a
[3, 4, 6, 9, 12, 13, 17, 6]
>>> a.count(6)
2
>>>
```

Acrescentamos um elemento com os valores 6. contamos quantas repetições o valor 6 tem na lista a.

```
>>> a.index(12)
4
>>> a
[3, 4, 6, 9, 12, 13, 17, 6]
```

Imprimimos o índice de um elemento com o valor 12.

EXERCÍCIO 3

Escreve uma programação que tenha uma lista de **n** números naturais como input. Imprime a lista dos números numa única linha.

```
n=int(input('Enter the number of elements in the list:'))
a=[]
for i in range(n):
    m=int(input('Enter a number: '))
    a.append(m)
print(a)
```

Explicação:

Explicação: na tarefa, podemos ver que o comprimento da lista está especificado – tem n elementos. Escrevemos os elementos usando a função para a qual escreverá um valor **n** vezes. Com o método **append()**, acrescentamos o elemento no final da lista a. Tivemos de inicializar a lista no início da programação. No final, visualizamos a lista usando simplesmente o comando print.

Fazendo uma lista dos valores de entrada:

Na última atividade, mostrámos como criar uma lista a partir de valores dados – caracteres ou números. No entanto, muitas vezes não sabemos o número de elementos na lista. Neste caso, não podemos usar o **for loop** com um número atribuído de funções. Neste caso, temos de usar o método **split()**, com o qual se divide uma sequência numa lista, ou seja, podemos separar palavras com um espaço em branco, dentro da string, e assim criar uma nova lista de palavras. Os elementos de uma lista feita com o método **split()** são sempre caracteres.

```
>>> names=input('Enter the names of the students: ').split()
Enter the names of the students: Steve, John, Katie, Sarah
>>> print(names)
['Steve,', 'John,', 'Katie,', 'Sarah']
>>>
```

EXERCÍCIO 4

Escreve um programa no qual daremos entrada a várias palavras e depois criamos uma nova lista que contém apenas a primeira letra de cada palavra.

```
a=input('Enter a list of words: ').split()
b=[]
for i in range(len(b)):
    x=a[i]
    b.append(x(0))
print(b)
```

Explicação:

a=input('Enter a list of words: ').split()	Insere as palavras da lista a
b=[]	Cria uma lista vazia b na qual iremos mais tarde armazenar as letras iniciais da lista a
for i in range(len(b)):	Define a repetição dependendo do comprimento da lista a
x=a[i]	Cada palavra em a é temporariamente armazenada na variável x
b.append(x(0))	Estamos adicionando novos elementos à lista, a primeira letra da palavra armazenada em x
print(b)	Imprimimos a nova lista b

EXERCÍCIO 5

Escreve um programa que dará entrada a uma lista de letras numa única linha e depois inverte a ordem dos elementos na lista. Publica a nova lista formada.

Exemplo de teste

```
Unesi slova: a b c d e f
['f', 'e', 'd', 'c', 'b', 'a']
>>>
```

Programa de computador

```
a=input('Unesi slova: ').split()
a.reverse()
print(a)
```

Podemos usar o último programa para reverter a lista dos números? Se os valores numéricos forem inseridos como caracteres (**sem int()**), então precisamos de os converter em números, usando essa função (**int()**).

```
>>> lista=input('Upiši brojeve: ').split()
Upiši brojeve: 7 6 3 12 4 15 8
>>> print(lista)
['7', '6', '3', '12', '4', '15', '8']
>>>
```

Lista de números armazenados como uma lista de caracteres.

Usando a função int() converte os caracteres da lista original em números.

```
>>> lista=input('Upiši brojeve: ').split()
Upiši brojeve: 7 6 3 12 4 15 8
>>> nova=[]
>>> for i in range(len(lista)):
        nova.append(int(lista[i]))
>>> nova.reverse()
>>> print(nova)
[8, 15, 4, 12, 3, 6, 7]
>>>
```

EXERCÍCIO 6

Depois de cada teste, os professores corrigem as respostas e analisam o trabalho dos alunos. Escreve um programa que introduza todas as classificações que os alunos obtiveram no teste. Usando as funções que aprendeste, imprime o número de alunos que teve A (ou 100) como nota, e o número de alunos que não passou no teste.

Exemplo de teste

```
Unesi ocjene: 1 2 3 5 4 3 2 5 3 5 1
3 su učenika dobila ocjenu odličan.
2 su učenika dobila ocjenu nedovoljan.
>>>
```

Programa de computador

```
lista=input('Unesi ocjene: ').split()
nova=[]
for i in range(len(lista)):
    nova.append(int(lista[i]))
print(nova.count(5),'su učenika dobila ocjenu odličan.')
print(nova.count(1),'su učenika dobila ocjenu nedovoljan.')
```

EXERCÍCIO 7

Escreve um programa que irá inserir uma lista arbitrária de números. O programa precisa criar e escrever uma nova lista de números que conterá apenas os números pares daqueles que foram inseridos inicialmente.

```
list=input('input the numbers: ').split()
new=[]
for in in range(len(list)):
    if int(list[i])%2==0:
        new.append(int(list[i]))
print(new)
```

Explicação:
Como na tarefa anterior, temos que criar uma nova lista, mas não a partir de todos os valores de entrada. Estamos apenas a usar aqueles que atendem à condição de serem iguais. Ao resolver esta tarefa, temos que nos lembrar do uso do operador % (resto) para testar se um número é par. É por isso que o comando crucial em nossa solução é:

if int (list [i])% 2 == 0:
list [i] - muda o índice, leva-nos através da lista
int - transforma o elemento da lista em um número
% 2 == 0 - verifica se o número é par
new.append - insere o número par no final da nova lista

EXCERCÍCIO 8

De acordo com o exemplo anterior, os alunos podem projetar, criar e testar seus próprios exemplos.

CONCLUSÃO

Alunos e professor discutem e avaliam as soluções apresentadas.

Métodos		***Formas de trabalho***
apresentação	*entrevista*	*Trabalho individual*
discussão	*demonstração*	*Trabalho em pares*
trabalhar no texto	*representação*	*trabalho em equipa/grupo*
trabalho gráfico		*trabalho frontal*
exercício interativo / simulação no computador		

Material:

-

Bibliografia

OBSERVAÇÕES PESSOAIS, COMENTÁRIOS E NOTAS