Listas

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

Listas

Em Python, uma lista é representada como uma sequência de objetos separados por vírgula e dentro de **colchetes []**, assim, uma lista vazia, por exemplo, pode ser representada por colchetes sem nenhum conteúdo. Pelo Python ser uma linguagem que não força a tipagem das variáveis, nas listas temos um caso incomum, podemos adicionar diferentes tipos de variáveis as suas inicializações.

Listas

```
1   lista1 = []
2
3   lista2 = [1,2,3,4,5]
4
5   lista3 = ['Yan', 'Fernando', 'Jorge']
6
7   lista4 = [1,'Yan', 3, 4, 'Fernando']
```

Operadores nas listas

O Python facilita algumas funções para os devs disponibilizando operadores e métodos para manipulação das listas definidas. Os operadores conseguem identificar elementos que estão presentes ou não, menores e maiores valores no array e outras operações, podemos ainda concatenar e multiplicar o tamanho da nossa listagem como se fossem métodos matemáticos.

Operadores (lista)

```
1   numeros = [1,2,-3,4,5]
2
3   # Captura o menor número da lista
4   print(min(numeros))
5   # Captura o maior número da lista
6   print(max(numeros))
7   # Soma todos os itens da lista
8   print(sum(numeros))
9
10  # Verificando se existe o elemento no array
11  print(6 in numeros)
12  print(-3 in numeros)
```

Métodos (lista)

```
1   numeros = [1,2,3,4,5]
2
3   numeros.append(6)
4   print(numeros)
5
6   numeros.insert(0, 0)
7   print(numeros)
8
9   numeros.pop()
10
11  numeros.remove(3)
```

Estrutura de Controle

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

Estruturas de Controle

As estruturas de controle servem para definir quais blocos de construção serão executados, para o Python temos acessível a estrutura ***if/elif/else***. O conceito é similar aos de outras linguagens, como Javascript, mas com a adição do Elif que interpreta como sendo uma possível execução para a estrutura, ou seja, checa mais uma possível condição e é uma abreviatura para ***else if***.

if/elif/else (Estrutura de controle)

```python
numeroDigitado = int(input('Informe um número: '))

if numeroDigitado == 0:
    print('A entrada é 0')
elif numeroDigitado > 0 and numeroDigitado <= 5:
    print('A entrada está entre 0 e 5')
else:
    print('A entrada é maior que 5 ou abaixo de 0')
```

Estruturas de Repetição

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

Estruturas de Repetição

As estruturas de repetição são utilizadas quando queremos que um bloco de código seja executado várias vezes. Em Python existem duas formas de criar um laço de repetição:

- O ***for*** (repetição contável) é usado quando se quer iterar sobre um bloco de código um número determinado de vezes. Ex: percorrer todos os itens de uma lista

- O ***while*** (repetição condicional) é usado quando queremos que o bloco seja repetido até que uma condição seja satisfeita, ou seja, é necessário que uma expressão seja declarada como ponto de parada do laço.

for (Repetição contável)

```
1  lista = [ 1, 2, 3, 4, 5 ]
2
3  for item in lista:
4      if item % 2 == 0:
5          print('{0} - Número par'.format(item))
6          continue
7      print('{0} - Número ímpar'.format(item))
```

for (Repetição contável)

```
while True:
    palavra = input('Digite uma palavra: ')
    if palavra.lower() == 'sair':
        print('Fim')
        break
    if len(palavra) < 2:
        print('Palavra curta.')
        continue
    print('Tente digitar \"sair\"')
```

Break e Continue

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

Break e Continue

Quando desejamos alterar o fluxo de execução de um laço de repetição, podemos utilizar as cláusulas break e continue.

- ***Break*** gera uma ação para interromper a execução, independente se a condição está satisfazendo ou não. Pode quebrar o laço que está ocorrendo com ***while*** ou ***for***.

- O ***continue*** é usado para prosseguir para próxima iteração, seu uso faz com o que o programa “pule” a execução posterior e volte a executar o laço a partir do bloco superior (incremento).

Exemplo Break

```
i = 0
while i < 50:
    print('Laço de repetição realizado {0} vez(es)'.format(i+1))
    if i == 9:
        break;
    i+=1
```

Exemplo Continue

```
1    i = 0
2    while i < 20:
3        i+=1
4        if i % 2 == 0:
5            print('{0} é par'.format(i))
6            continue
7        print('{0} é ímpar'.format(i))
8
```

Classes e Métodos

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

Classes

Classes proporcionam uma forma de organizar os dados e funcionalidades juntos, criar uma nova classe cria um novo “tipo” de objeto, permitindo que novas instâncias desse tipo sejam produzidas. Cada instância da classe pode ter atributos anexados a ela, para manter seu estado.

Classes

```
1   class Cachorro:
2       # variável da classe aplicada a todas as instâncias
3       tipo = "canino"
4
5       def __init__(self, nome):
6           # variável única para cada instância
7           self.nome = nome
8
9   c1 = Cachorro('Buzzy') # instância 1
10  c2 = Cachorro('Layka') # instância 2
11  print('{0} - {1}'.format(c1.tipo, c1.nome))
12  print('{0} - {1}'.format(c2.tipo, c2.nome))
```

Classes

```
1   class Cachorro:
2       # variável da classe aplicada a todas as instâncias
3       tipo = "canino"
4
5       def __init__(self, nome):
6           # variável única para cada instância
7           self.nome = nome
8
9   c1 = Cachorro('Buzzy') # instância 1
10  c2 = Cachorro('Layka') # instância 2
11  print('{0} - {1}'.format(c1.tipo, c1.nome))
12  print('{0} - {1}'.format(c2.tipo, c2.nome))
```

Definição da classe com um método __init__, que é o construtor da classe. Este método é chamado automaticamente na criação de uma nova instância.

Classes

```
1   class Cachorro:
2       # variável da classe aplicada a todas as instâncias
3       tipo = "canino"
4
5       def __init__(self, nome):
6           # variável única para cada instância
7           self.nome = nome
8
9   c1 = Cachorro('Buzzy') # instância 1
10  c2 = Cachorro('Layka') # instância 2
11  print('{0} - {1}'.format(c1.tipo, c1.nome))
12  print('{0} - {1}'.format(c2.tipo, c2.nome))
```

Criamos instâncias da nossa classe Cachorro, passando o nome do cachorro como parâmetro para o “construtor” (__init__).

Classes

```
1   class Cachorro:
2       # variável da classe aplicada a todas as instâncias
3       tipo = "canino"
4
5       def __init__(self, nome):
6           # variável única para cada instância
7           self.nome = nome
8
9   c1 = Cachorro('Buzzy') # instância 1
10  c2 = Cachorro('Layka') # instância 2
11  print('{0} - {1}'.format(c1.tipo, c1.nome))
12  print('{0} - {1}'.format(c2.tipo, c2.nome))
```

Acessando nossas instâncias a partir de suas propriedades, uma instância pode ser vista como um objeto e a classe sendo a interface desse objeto criado.

Classes (Conceitos)

classe : Define uma estrutura de dados para o modelo, com métodos, propriedades, etc.

instância : uma representação da classe que foi inicializada.

__*init*__ : “Construtor” da classe, é o método chamado da inicialização de uma nova instância

self : Pode ser compreendido como a referência da própria instância que está sendo inicializada

Métodos

```
class Cachorro:
    tipo = "canino"

    def __init__(self, nome, idade = -1):...

    # método sem inicialização automática
    def setCor(self, cor):
        self.cor = cor

c1 = Cachorro('Buzzy', 3)

try:
    print(c1.cor)
except:
    print('Cor do cachorro não foi definida.')

c1.setCor('preto') # Chamo o método setCor

print('{0} tem a cor {1}'.format(c1.nome, c1.cor))
```

Podemos definir novos métodos (funções) na classe para receber ou trabalhar um outro conjunto de informações e adicionar novas propriedades a nossa instância.

Exercício 1 (20min)

Descrição Editar

Criar uma aplicação que pergunte o usuário que função matemática deseja realizar.

Crie uma aplicação em Python, utilizando apenas um arquivo, que receba três informações do usuário.

1. Função matemática que deseja (soma, subtração, divisão, multiplicação)
2. Número "a"
3. Número "b"

Passe os dois números como parâmetro para a função matemática informada e imprima o retorno da função na tela.

Modificar o exercício descrito ao lado para que as funções matemáticas fiquem dentro de uma classe e as operações sejam os métodos dessa classe. Defina o construtor para receber os números **“a”** e **“b”**

Import

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

Import

Grande parte das funcionalidades do Python estão em suas bibliotecas, sejam as chamadas “nativas” ou os pacotes que são instalados por meio do gerenciador ***pip***. A importação das bibliotecas ou arquivos seguem o mesmo padrão, ver modelo a seguir.

Import (Exemplos)

```python
# do arquivo/biblioteca importe a classe/funcao
from calculadora import Calculadora

# importando uma biblioteca
import os.path

# importando pacote e declarando que estará acessível pelo nome np
import numpy as np

from numpy import sum
# Printando o diretório e arquivo atual
print(os.path.curdir)

s = sum(np.arange(10))
print(s)
```

Biblioteca Matplotlib

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

Matplotlib

Matplotlib é uma biblioteca de software para criação de gráficos e visualizações de dados em geral. O pacote transforma os dados em figuras, cada uma contendo uma ou mais eixos, especificando os pontos gráficos em coordenadas x-y. Para cada tipo de gráfico existe um função específica, que a depender da forma em que os dados necessita se dispor no gráfico, deve ser indicado pelos seus argumentos.

Momento Prática

- Criar um ambiente virtual para a semana 2 do módulo 2 (Ignore caso já tenha um ambiente criado)
- Ativar o ambiente por meio do comando **activate**.
- Instalar pacote ***matplotlib*** ao ambiente virtual e referencia-lo (importar) em um arquivo.

Material Complementar

- 15 comandos Matplotlib - https://paulovasconcellos.com.br/15-comandos-de-matplotlib-que-talvez-voc%C3%AA-n%C3%A3o-conhe%C3%A7a-17cf88a75119
- Matplotlib - https://pypi.org/project/matplotlib/
- Numpy - https://algoritmosempython.com.br/cursos/programacao-python/strings/

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>

AGENDA

- Informações do template
- Exemplo de texto (Light mode)
- Exemplo de texto (Dark mode)
- Exemplo de tópicos (Light mode)
- Exemplo de tópicos (Dark mode)
- Exemplo com imagens (Light mode)
- Exemplo com imagens (Dark mode)
- Exemplo com tabelas (Light mode)
- Exemplo com tabelas (Dark mode)

INFORMAÇÕES DO TEMPLATE

- Título da Apresentação:
  - Fonte: Ubuntu Bold
  - Formato: Maiúsculo
  - Tamanho: 34
  - Cor: Branco
- Título do Slide:
  - Fonte: Ubuntu Bold
  - Formato: Maiúsculo
  - Tamanho: 22
  - Cor: Branco
- Parágrafos:
  - Fonte: Open Sans Normal
  - Tamanho: 14 a 18
  - Cores: Branco (Dark Mode) ou Preto (Light Mode)

- Marcadores de tópicos:
  - Formatos: Símbolos ou Alfanuméricos
  - Cor: Laranja
- Padrão de Cores:
  - Cinza - #868584
  - Preto - #1C1C19
  - Branco - #FAFAFA
  - Laranja - #F08305
  - Rosa - #c71d81
  - Azul - #0e1d8e

EXEMPLO DE TEXTO (LIGHT MODE)

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

EXEMPLO DE TEXTO (DARK MODE)

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

EXEMPLO DE TÓPICOS (LIGHT MODE)

- Lorem Ipsum is simply dummy text of the printing and typesetting industry;
- Lorem Ipsum has been the industry's standard dummy text ever since the 1500s;
- Lorem Ipsum is simply dummy text of the printing and typesetting industry;
- Lorem Ipsum has been the industry's standard dummy text ever since the 1500s;
- Lorem Ipsum is simply dummy text of the printing and typesetting industry;
- Lorem Ipsum has been the industry's standard dummy text ever since the 1500s;

EXEMPLO DE TÓPICOS (DARK MODE)

- Lorem Ipsum is simply dummy text of the printing and typesetting industry;
- Lorem Ipsum has been the industry's standard dummy text ever since the 1500s;
- Lorem Ipsum is simply dummy text of the printing and typesetting industry;
- Lorem Ipsum has been the industry's standard dummy text ever since the 1500s;
- Lorem Ipsum is simply dummy text of the printing and typesetting industry;
- Lorem Ipsum has been the industry's standard dummy text ever since the 1500s;

EXEMPLO COM IMAGENS (LIGHT MODE)

Título

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Título

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

EXEMPLO COM IMAGENS (DARK MODE)

Título

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Título

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

EXEMPLO COM TABELAS (LIGHT MODE)

Lorem	Ipsum	Ipsum	Ipsum	Ipsum	Ipsum
	1	2	3	4	5
xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx

EXEMPLO COM TABELAS (DARK MODE)

Lorem	Ipsum	Ipsum	Ipsum	Ipsum	Ipsum
	1	2	3	4	5
xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx