Universidade Federal de Santa Maria
UFSM
1960

CT
CENTRO DE TECNOLOGIA - UFSM

Departamento de Processamento de Energia Elétrica

Disciplinas de *Deep Learning* e *Sistemas Inteligentes*
F01 – Estudo Dirigido de Python

Prof. Rodrigo da Silva Guerra

26 de outubro de 2020

Esta lista de exercícios foi elaborada para servir como uma espécie de estudo dirigido, para que você avalie seu nível de conhecimento de Python e NumPy para um bom acompanhamento da disciplina. Estes exercícios não serão avaliados, mas problemas semelhantes aos aqui apresentados podem aparecer nos trabalhos e provas da disciplina. Mesmo que já tenhas alguma experiência programando em Python, tente ler os enunciados e refletir se saberias resolver cada um destes exercícios. Se houver algum exercício para o qual não tenhas certeza, então tente resolvê-lo e se não conseguir, busque revisar o conteúdo pertinente. Se ainda assim encontrar dificuldades, traga suas dúvidas ao professor.

Atenção: Demonstre algum esforço tentando resolver o exercício antes de procurar ajuda do professor. Explique seu raciocínio e traga dúvidas específicas.

Exercícios

1. Crie uma função que recebe dois números inteiros e retorna o produto, mas caso o produto seja maior que 1000, retorna a soma.
2. Escreva um programa capaz de gerar o padrão abaixo utilizando laços.

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

3. Escreva uma função que, recebendo um número inteiro `n`, e retorne os `n` primeiros números da sequência de Fibonacci.

4. Crie uma função que, recebendo como argumento uma lista e um número, adicione o número ao final da lista apenas se a lista ainda não contém aquele número.

5. Escreva uma função que recebe uma lista e retorna um dicionário representando quantas vezes cada elemento aparece na lista.

6. Faça uma função que recebe duas listas de igual tamanho e retorna um conjunto de tuplas dos pares de elementos na mesma posição em cada lista.

7. Crie uma função que receba uma frase e retorne o conjunto de todas palavras da frase cujas iniciais sejam maiúsculas. Inicialmente considere apenas frases compostas de palavras separadas por espaços. Depois, considere empregar expressões regulares utilizando o módulo `re` para remover pontuação também.

8. Transforme a brincadeira da figura abaixo em um programa que gera o "nome japonês" automaticamente:

9. Faça um programa que recebe uma sequência genética de aminoácidos na forma de uma string longa com as letras A, C, G, T (por exemplo "`ACGAATTCCGCGAATTC`") e retorna todas as substrings de exatamente 10 aminoácidos que se repete pelo menos uma vez.

Ex.1: `"AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"`
⇨ `["AAAAACCCCC","CCCCCAAAAA"]`

Ex.2: `"AAAAAAAAAAAAA"`
⇨ `["AAAAAAAAAA"]`

10. Escreva um programa que seja capaz de unir esses dois dicionários abaixo em um único dicionário

```
d1 = {'Dez': 10, 'Vinte': 20, 'Trinta': 30}
d2 = {'Trinta': 30, 'Quarenta': 40, 'Cinquenta': 50}
```

11. Escreva código capaz de acessar a chave '`history`' no dicionário abaixo

```
sampleDict = {
   "class":{
      "student":{
         "name":"Mike",
         "marks":{
            "physics":70,
            "history":80
         }
      }
   }
}
```

12. Crie uma classe `Onibus` que herda todos métodos e atributos da classe `Veiculo` abaixo.

```
class Veiculo:

    def __init__(self, nome, max_vel, quilometragem):
        self.nome = nome
        self.max_vel = max_vel
        self.quilometragem = quilometragem
```

13. Escreva uma classe `Retangulo` que é inicializada com as propriedades lado e altura, e uma classe `Circulo` que é inicializada com a propriedade raio. Ambas classes devem conter seus métodos `area()` que retorna a área de cada respectiva figura geométrica. Em seguida crie uma lista de dez objetos contendo objetos aleatórios de ambas classes e faça uma função que calcula a soma das áreas de todos objetos da lista.

14. Escreva um programa que gere uma senha aleatória que atenda as seguintes condições: (1) deve ter 10 caracteres, (2) deve conter duas letras maiúsculas, (3) deve conter 1 dígito, (4) deve conter um símbolo

15. Crie uma função que recebe um *array* do `NumPy` de duas dimensões e retorna duas listas, a primeira contendo os maiores elementos de cada coluna e a segunda contendo os maiores elementos de cada linha.

16. Crie uma função que recebe um *array* do `NumPy` e que substitua todos números pares por `-1` (em única linha, sem utilizar laços)

17. Transforme um *array* do `NumPy` de uma dimensão e comprimento igual a `10` em um *array* de duas dimensões, com `2` linhas e `5` colunas (em única linha, sem utilizar laços)

18. Crie um *array* do `NumPy` de uma dimensão com inteiros aleatórios entre 0 e 10. Em seguida escreva código que mostre o *array* resultante depois de remover todos elementos no intervalo inclusivo de 4 a 7 (é possível fazer em apenas duas linhas de código, sem laços).

19. Utilizando o código abaixo como ponto de partida, normalize o *array* `sepallength` para todos valores fiquem na escala 0 a 1.

```
import numpy as np
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
sepallength = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0])
# Seu Código aqui abaixo
```

20. Utilize a função `plot` do submódulo `matplotlib.pyplot` para plotar o gráfico das funções $f(x) = e^{-x/10} \cdot sin(\pi x)$ e $g(x) = xe^{-x/3}$ no intervalo [0, 10]. Inclua legendas explicando qual curva representa qual função.

Entrega

Este é um estudo dirigido, portanto não haverá entrega.