Introdução à Programação Orientada a Objetos
Prof. Elder Rizzon Santos
Universidade Federal de Santa Catarina
Sistemas de Informação

Prova 2 - 2023/1

Elabore soluções na linguagem Python para as seguintes questões. Utilize somente recursos da linguagem abordados em aula, em caso de dúvida sobre poder ou não utilizar algum recurso, pergunte. É possível utilizar somente as funções len, list e keys.

1. Desenvolva uma **função** a qual recebe uma matriz quadrada (caso não seja, retorne uma lista vazia) e **retorna uma lista contendo o maior elemento de cada diagonal acima da principal.** Considere que a diagonal mais próxima da principal tem índice 1 e os índices das demais aumentam de 1 em 1. *(4,0)*

Exemplos de parâmetro e retorno:

Parâmetro	**Retorno**	**Parâmetro**	**Retorno**
1 1 -4 2 -1 -2 3 -2 1	[1,-4]	8 -3 1 4 -4 2	[]
1 2 4 2 1 2 3 1 1	[2,4]	1 2 3 4 6 1 2 4 5 7 1 2 5 7 8 0 5 7 1 9 1 2 5 7 8	[9, 8, 7, 6]

2.1 Continue a implementação do construtor da classe Data (código especificado a seguir) de acordo com as seguintes especificações: *(2,0)*

- os parâmetros do construtor e os valores armazenados nos atributos são números inteiros (não é necessário verificar).
- utilizando, **obrigatoriamente**, o dicionário meses (já presente no construtor), verifique se o parâmetro dia é válido, ou seja, é um valor maior do que 0 e não pode ser maior do que a quantidade de dias do mês - essa informação está no dicionário meses.
- além do dia ser válido, o parâmetro mes precisa ser um valor entre 1 e 12 (inclusive) e o parâmetro ano precisa ser maior do que 1900
- caso os parâmetros dia, mes e ano estejam válidos, seus valores devem ser atribuídos aos atributos privados dia, mes e ano.
- caso os parâmetros não estejam válidos, os atributos (privados) devem receber os valores 26, 6 e 2023 respectivamente.

2.2 Sobrescreva o método __str__ na classe Data de modo a retornar a data armazenada no seguinte formato: dia/mes/ano. Exemplo de retorno: '3/4/2023'. *(1,0)*

3. Construa a classe Bolsista de acordo com as especificações a seguir. A classe Bolsista é uma subclasse da classe Estudante (código especificado a seguir). Atributos: além dos atributos herdados, tem o atributo protegido dataInicio, representando a data em que a bolsa iniciou.

a) Construtor: possui parâmetros para inicializar todos os atributos da classe. O parâmetro dataInicio é opcional e tem como valor padrão um objeto da classe Data representando a data 26/6/2026. É obrigatório executar o construtor da superclasse (usando o *super*) dentro desse construtor para inicializar os atributos vindos da superclasse, o restante deve ser inicializado conforme o argumento recebido. *(1,5)*

4. Adicione o método obtemEstudantesIAA na classe Cadastro (código especificado a seguir) conforme as seguintes especificações: *(1,5)*

- o método retorna uma lista com objetos da classe Estudante ou Bolsista cujos IAA são superiores ao parâmetro iaa deste método.
- os estudantes ou bolsistas são armazenados no atributo cadastro, que é um dicionário. Observe no método para adicionar um estudante (adicionaEst - que recebe um objeto da classe Estudante) qual é a chave e qual é o valor do cadastro.

Não serão consideradas modificações nas linhas de código a seguir:

```
class Data:
    def __init__(self, dia=1, mes=1, ano=2023):
       meses={1:31, 2:28, 3:31, 4:30, 5:31, 6:30, 7:31, 8:31, 9:30,
10:31, 11:30, 12:31}
```

```
class Estudante:
      def  __init__(self,  mat,
iaa):
       self.__mat = mat
       self.__iaa = iaa
   def obtemMatricula(self):
       return self.__mat
   def obtemIAA(self):
       return self.__iaa
```

```
class Cadastro:
  def __init__(self):
     self.__cadastro = {}

  def adicionaEst(self, est):
     mat = est.obtemMatricula()
     self.__cadastro[mat] = est
```