Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação

Disciplina: Programação com Interfaces Gráficas

AP3 2^o semestre de 2019.

Nome -

Assinatura -

Algumas das questões a seguir usam o canvas do tkinter para desenhar. Para quem não se lembra, eis uma lista das funções usadas. Lembre-se também que em tkinter o eixo y aponta para baixo, e que o ponto de coordenadas (0,0) é o do canto superior esquerdo da janela

`create_line(`x_0`,`y_0`,`x_1`,`y_1`)` Desenha um segmento de reta entre os pontos (x_0,y_0) e (x_1,y_1).

`create_oval(`$x-w$`,`$y-w$`,`$x+w$`,`$y+h$`)` Desenha uma elipse com centro em (x,y), largura w e altura h.

1. (3 pontos) Esboce o que é produzido pelo código abaixo. Assuma que a janela gráfica é grande o suficiente para conter o desenho. Não é necessário dar as coordenadas dos pontos, apenas manter as proporções.

```
def f(x,y,w,h,l):
    def q(p):
        return y+(p-a)/(b-a)*h

    a,b = 1.0*min(l),1.0*max(l)
    x0,y0 = x,q(l[0])
    for i in range(1,len(l)):
        x1 = [x,x+w][i%2]
        y1 = q(l[i])
        canvas.create_line (x0,y0,x1,y1, fill='white')
        x0,y0 = x1,y1
    for p in l:
        canvas.create_oval(x-10,q(p)-10,x+10,q(p)+10,outline='white')
        canvas.create_oval(x+w-10,q(p)-10,x+w+10,q(p)+10,outline='white')

f(50,50,400,400,[3,1,0,2,4,3,5])
```

2. (7 pontos) No código parcial abaixo, define-se a classe `Intervalo` e a função `adiciona(l,i)` As funcionalidades de cada método e da função estão expressas nos comentários respectivos. Pede-se completar o código, isto é, escrever as porções marcadas com ". . .", de forma a satisfazer as especificações.

```
class Intervalo:
    u"""Representa um intervalo [a,b] entre 2 valores a e b (números
    inteiros ou ponto-flutuante)."""

    def __init__(self,a,b):
        u"""Cria um intervalo entre a e b. Os valores a e b podem
        ser dados em qualquer ordem."""
        ...

    def __repr__(self):
        u"""Retorna uma representação string deste objeto."""
        ...
```

```
    def intersecao (self,i):
        u"""Retorna um objeto da classe Intervalo com a interseção
        entre self e o intervalo i, caso exista, ou causa uma exceção
        ValueError, caso contrário. Assume-se que 2 intervalos se
        intersectam se têm ao menos um ponto em comum."""
        ...

    def uniao(self,i):
        u"""Retorna um objeto da classe Intervalo com a união entre
        self e o intervalo i. Assume-se que união de dois intervalos
        é o menor intervalo que contém ambos"""
        ...

def adiciona(l,i):
    u"""Dada l, uma lista de intervalos que não se intersectam,
    retorna uma nova lista contendo o novo intervalo i dado.
    Se i intersecta algum intervalo j de l, este é substituído pela
    união de i e j, de tal maneira que a lista resultante
    só contenha intervalos que não se intersectam."""
    for k,j in enumerate(l):
        try:
            x = i.intersecao(j)
            return ...
        except ValueError:
            pass
    return ...
```

Eis um exemplo de utilização:

```
print Intervalo(4,1).uniao(Intervalo(10,2))
print Intervalo(1,4).intersecao(Intervalo(2,10))
try:
    print Intervalo(1,4).intersecao(Intervalo(6,10))
except ValueError:
    print "Deu Bode"
l = adiciona([Intervalo(2,3)],Intervalo(0,1))
print l
l = adiciona(l,Intervalo(1,2))
print l
l = adiciona(l,Intervalo(7,9))
print l
l = adiciona(l,Intervalo(4,5))
print l
l = adiciona(l,Intervalo(2,8))
print l
```

Eis o que é impresso pelo exemplo acima:

```
Intervalo(1,10)
Intervalo(2,4)
Deu Bode
[Intervalo(2,3), Intervalo(0,1)]
[Intervalo(0,3)]
[Intervalo(0,3), Intervalo(7,9)]
[Intervalo(0,3), Intervalo(7,9), Intervalo(4,5)]
[Intervalo(0,9)]
```