Python講座

2016/6/30(木)

今回の内容

- 期末試験対策
- 問題を解く、予定は1時間30分
- 解答・解説

※プログラムの試験なので、実際に(Pythonで)コードを書いて実行できれば、それが答え。
つまり、難しく考えてはいけない。

[1]. (1)

- 空欄に数値のリストvaluesの各要素を2倍したリストを作成する処理を書きなさい。
- この例では、最後の行で変数resultの内容が[2,4,8,16,8]になっていること

```
values = [1,2,4,8,4]
result = []

print("結果:{}".format(result))
```

[1]. (2)

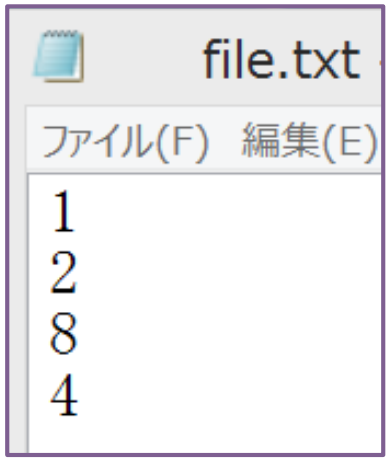
- 空欄に数値のリストvaluesの要素の合計値を求める処理を書きなさい。ただし、組み込み関数sumを利用せず書くこと
- この例では、最後の行で変数totalの内容が19になっていること

```
values = [1,2,4,8,4]

print("結果:{}".format(total))
```

[1]. (3)

- 数値が一行に一つ記載されたファイル(“C¥file.txt”)を読み、各行の数値を4倍したリストを表示させるプログラムを作成せよ。
- この例の場合はリスト[4,8,32,16]を表示させる




```
print("結果:{}".format(result))
```

[2]

- タートルに与える「命令」を、属性「”command(動作名)”」「”param”(パラメータ)」の二つからなるオブジェクトとして、モデリングすることを考える。動作名は”fd”と”lt”の二種類であり、それぞれ、タートルの前進と左回転を意味する。パラメータは動作”fd”に対してはタートルの進むステップ数、”lt”に対してはタートルの回転角度(度数法)である。
- P.7　　解答欄
- P.8.9　問題(1)~(4)

```
import turtle
def new_inst(command, param):
```

(a)

```
def execute(inst_list):
    t = turtle.Pen()
    for
```

(d)

```
        c = inst["command"]
        p = inst["param"]
        if
```

(e)

```
            t.forward(p)
        elif
```

(f)

```
            t.left(p)

inst_list = []
while True:
```

(b)

```
    inst = i.split(":")
```

(c)

```
    inst_list.append(new_inst(c, p))

execute(inst_list)
```

[2]

(1) 空欄(a)を埋め、関数new_instを完成させよ。

(2) 以下の説明に合うよう、空欄(b)に適切な処理を書け。

- このプログラムのwhile文では、タートルに与える命令を「fd:200」や「lt:90」のように、動作名:パラメータ、という形式の入力を受け、対応する命令オブジェクトをリストinst_listに追加し、命令列を登録させる処理を行う。空行を入力された場合は、whileループを打ち切り、命令列の登録処理を終了し、その後、登録された命令列をexecute関数によって実行される。

[2]

(3) whileの本体の最後の文では、命令オブジェクトをinst_listに追加している。空欄(c)で行うべき処理を記入せよ。

(4) executeはリストinst_listにある各命令列を順に処理する関数である。空欄(d)~(f)に適切なコードを記入し関数を完成させよ

[3]

- 次の疑似コードに関して次の問いに答えよ。

```
関数F(数値のリスト):
  X = 数値リストの最初の要素 - 1
  「数値のリスト」のそれぞれの数値について:
    もし、数値 > Xならば:
      X = 数値
    そうでなければ:
      FALSEを返す
  TRUEを返す
```

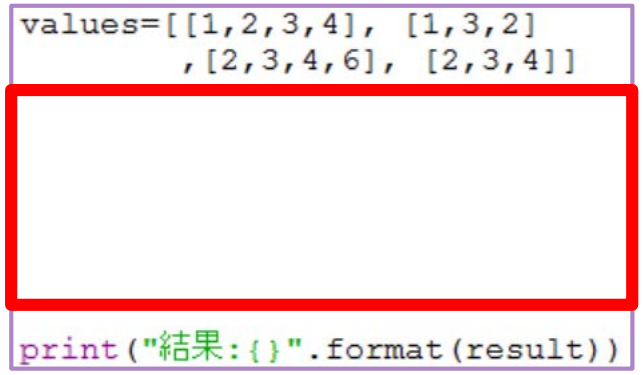
(1) 関数Fに引数としてそれぞれリスト[1,3,4,5]、[1,3,2,4]、[1,1,2,3]を与えて実行された場合の返り値を答えよ

(2) 上記の疑似コードをPythonの関数で記せ。
関数名はis_ascendingとせよ。

[3]

(3) 複数の数値リストが与えられた時に、各リストのうち数値の並びが単調増加となっているもののみを残して表示するプログラムを書け。(2)の関数を利用せよ。

- この例の場合、四つのリストのうち、単調増加となっていないリスト[1,3,2]を取り除いた[[1,2,3,4],[2,3,4,6],[2,3,4]]というリストを新たに生成し、表示すればよい。



[4]

- 平面上の円を表すオブジェクトを、Pythonの辞書を利用して、プログラミングすることを考える。次のnew_circle(x,y,r)がオブジェクトを生成する関数である。

```
def new_circle(x,y,r):
    return {"x":x, "y":y, "r":r}
```

(1) (a)円を表すオブジェクトを受け取り、その面積を求める関数areaを書け。円周率は3.14としてよい。

(b)area関数を利用してリスト内の円の面積の合計を求める処理を書け。組み込み関数sumを用いてもよい。

- P.12,13 (1)、P.14,15 (2) 、P.16 (3)、P.17 (4)

[4]

(1) 解答欄

```
def
```

(a)

```
circle_list = [new_circle(0,0,5),new_circle(10,10,3),
               new_circle(0,5,2),new_circle(-10,-0, 6)]
```

(b)

```
print("結果:{}".format(area_total))
```

[4]

(2) 円のリストのおのおのの円について、指定した基準の数値より面積が小さいものと、その面積以上のものに分類する関数divideByAreaを作成せよ。空欄 (c)を埋めよ。

- プログラムのcircle_listの例を用いると、半径4の円が面積の基準となるので、以下のように表示される。

```
面積が基準より小さい:[{'x': 10, 'y': 10, 'r': 3}, {'x': 0, 'y': 5, 'r': 2}]
面積が基準以上である:[{'x': 0, 'y': 0, 'r': 5}, {'x': -10, 'y': 0, 'r': 6}]
```

[4]

(2) 解答欄

```
def divideByArea(circle_list, v):
    small = []
    large = []
```

(c)

```
    return [small, large]

circle_list = [new_circle(0,0,5),new_circle(10,10,3),
               new_circle(0,5,2),new_circle(-10,-0, 6)]

criteria = 4*4*3.14
result = divideByArea(circle_list, criteria)
print("面積が基準より小さい:{}".format(result[0]))
print("面積が基準以上である:{}".format(result[1]))
```

[4]

(3) 円c1,c2に対して、それらが重なっているかどうかを判定するis_intersectを書け。

- ここで、c1とc2が接している場合も、それらは重なっているものと見なす。「c1とc2が接する」とは、c1とc2の中心間の距離をdとし、c1とc2の円の半径の合計をdrとしたとき、d==drあるいは、d*d==dr*drの場合を指す。平方根を求める関数はmath.sqrtとし、必要であればこれを利用して良い。たとえばmath.sqrt(2)の結果は、2の平方根(の近似値)である。

[4]

(4) 円のリストcircle_listに含まれるの組のうち、重なっているもの同士の組の数を求める処理を記入せよ。ただし、同じ円は組の数には数えない。プログラムのcircle_listの例では、リストの各要素の円を先頭から順にc0,c1,c2,c3とすると、次の表で重なりの関係を表せる。

	c0	c1	c2	c3
c0	ー	重ならない	重なる	重なる
c1	重ならない	ー	重ならない	重ならない
c2	重なる	重ならない	ー	重ならない
c3	重なる	重ならない	重ならない	ー

- 重なりのある組は{c0,c2},{c0,c3}の二通りであるから、結果は2と表示される。

[5]

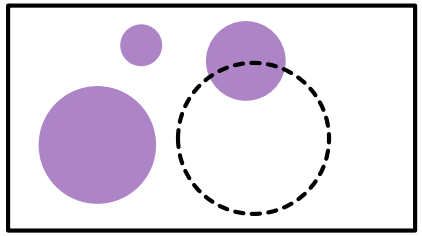
- プレイヤーが円を次々と平面上に配置していくようなゲームを作ることを考える。このゲームの流れは以下である。

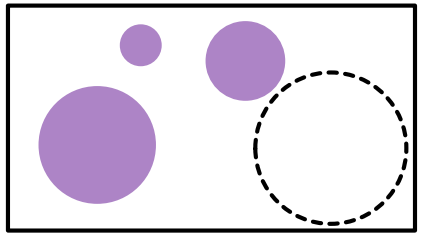
```
ステップ0.  初期設定を行う
            (a)一つ目に置く円の半径を決める
            (b)「円リスト」を初期化する
            (c)初期の点数を0点とする
ステップ1.  プレイヤーは中心座標を指定し、新たに配置する円NEW_Cを用意する
ステップ2.  画面に円NEW_Cを描画する
            (それまでに置いた円は、そのまま描画されている)
ステップ3.  円NEW_Cが、それまでに配置した円と重なったかどうかを判定する。
            重なった場合、点数とゲーム終了のメッセージを表示して
            プログラムを終了する。
ステップ4.  重ならなかった場合、円NEW_Cの面積を点数に加え、「円リスト」に
            円NEW_Cを追加する。また、次に配置する円の半径を前回のものより
            C だけを増やす。
ステップ5.  ステップ1に戻る
```

- P.18,19 説明、P.20 解答欄、P.21 (1),(2)

[5]

- 下記の左図はプレイヤーが置こうとする円(点線で表示)がそれまでに配置した円に重なる状態であり、右図は重ならない状態である。





- 以上の説明に基づいたゲームのプログラムの一部を解答欄[5]のように記述した。このプログラム中では、円をモデリングするために、[4]で説明した円オブジェクトやその関数を利用している。一部の関数の内部は省略しており説明のみを記述している。

```
def new_circle(x, y, r):#円オブジェクトを作成する。
def is_intersect(c1, c2):#円c1,円c2が重なっているかどうかを判定する。
def area(c):#円cの面積を求める。
#以上3つは[4]で作成済

def redraw(c):#円cを（これまでの円に加えて）画面に描画する。
def print_score(score):#点数scoreを表示しなおす。
def print_gameovaer(score) :#ゲーム終了時のメッセージを表示する。点数も表示する。

def detect_collision(new_circle, circles):
```

(1)

```
r = 20 #一つ目の円の半径を20とする。
C = 10 #半径の増加量
circles = []
score = 0
while True:
    x = int(input("x座標>"))
    y = int(input("y座標>"))
    New_circle = new_circle(x, y, r)
```

(2)

[5]

(1) ステップ3において、プレイヤーが新しく配置した円が、それまでに配置した円のどれかと重なったかどうかを判定するために、以下の関数detect_collisionを記述せよ

- 「円new_circleが、円のリストcirclesに含まれる円のどれかと重なる場合Trueを返し、どれとも重ならない場合Falseを返す」

(2) 解答欄[5]のプログラムでは、ステップ2~5が未完成である。プログラムを記述し、ゲームを完成させよ。