

## 課題演習

### 基本事項・リスト・関数

初心者用設問 1 0 FirstQ00.py

キーボードからの3桁の文字整数を入力する。（ここでは、必ず入力すると仮定する。）  
まず、1から3桁の整数までを sum1 に格納する。  
つぎに、3の倍数ならば  
    sum3 に 奇数の和を  
そうでないならば  
    sum2に 偶数の和を  
格納する。

```
# 課題 FirstQ00.py
```

```
# ----- キーボード入力 -----
```

```
numberStr = input("3桁の文字整数 = ")
```

```
num = int(numberStr) # 文字整数を整数に変換
```

```
sum1 = 0 # 和 :
```

```
sum2 = 0 # 偶数和 :
```

```
sum3 = 0 # 奇数和 :
```

```
# ----- 和 : sum1 = 0 + 1 + 2 + ... + num -----
```

```
# 3の倍数か否かの判定
```

```
if num%3==0:
```

```
    # ----- 奇数和 : sum3 = 1 + 3 + 5 + 7 + ... + < num + 1 -----
```

```
else:
```

```
    # ----- 偶数和 : sum2 = 0 + 2 + 4 + 6 + ... + < num + 1 -----
```

```
print(" (%d)までの和 = (%d)" %(num, sum1) )
```

```
print(" (%d)までの偶数和 = (%d)" %(num, sum2) )
```

```
print(" (%d)までの奇数和 = (%d)" %(num, sum3) )
```

## 初心者用設問 1 1 FirstQ00.py

判定の追加（仮定：文字整数を必ず入力する）  
3桁の文字整数であるか否かの判定。

```
num = 0

# ★★★★★★★★★★★★★★★★★★

count = 0

while True :
    # ----- キーボード入力 -----
    numberStr = input("3桁の文字整数 = ")

    # 文字整数を整数に変換
    num = int(numberStr)

    count = count + 1
    # ◆◆◆3桁の判定 100以上 かつ(and) 1000未満◆◆◆
    if 100<=num and num<1000:
        break
    else:
        print("3回で終了です。Count =", count); print();print()
        if count==3:
            print("強制終了"); print();print()
            break

# ★★★★★★★★★★★★★★★★★★
```


## 初心者用設問 1 2 FirstQ00.py

上記プログラムの第2番目の

```
# ★★★★★★★★★★★★★★★★★★
```

以降を関数 def SumFunc( ??? ) に修正しなさい。ただし、引数 ??? は自分で考えなさい。

```
if 100<=num and num<1000:
    break
```



```
if 100<=num and num<1000:
    sumFunc(???)
    break
```

下記を満足するプログラムを FirstQ11.py に作成しなさい。

(1)

変数 x に10、変数 y に9.8、変数 z に 'Hello World!' をそれぞれ代入し  
これらのx, y, z を用いて

```
変数 x = (10), 変数 y = (9.8), 変数 z = ("Hello World!")
```

と一行に表示するプログラムを作成しなさい。

(2)

キーボードから

```
str = input("2桁の文字整数=")
number = int( str )
```

のように整数を取得して、

$0 \leq \text{number} < 33$  (数学表記)

```
変数 sum33 に sum33 = 0 + 1 + 2 + ... + number を代入し、
print( "和の合計=", sum33 )
```

$33 \leq \text{number} < 66$  (数学表記)

変数 sum66 に(5の倍数)を除いて、つぎを満足する加算を行う。

```
sum66 = 0 + 1 + 2 + 3 + 4 + 6 + 7 + 8 + 9 + 11 + ... + number < 500
```

または

```
sum66 = 0 + 1 + 2 + 3 + 4 + 6 + 7 + 8 + 9 + 11 + ... + < 500
```

```
print( "入力=(%d) (%d)までの加算の合計=(%d)" %(number, ????, sum66) )
```

$66 \leq \text{number} < 100$  (数学表記)

```
print( "66以上の入力数値=", number )
```

とするプログラムを作成しなさい。

(3)

リスト list0 = [ 1, 2, 'AA' ] と宣言・初期化して

```
list0 = [1, 2, "AA", 3, [4, 5], 6 ]
```

となるように追加し、

```
list0 = [1, 2, 3, [4, 5], 6 ]
```

となるように"AA"を削除するプログラムを作成しなさい。

(4)

まず、つぎの関数をプログラムしなさい。

```
関数 def MaxValueCal( list1 ):
    引数 list1 は2個以上の整数が格納されているリストである。
    最大値 maxvalue を求めて、戻り値としなさい。
```

つぎに、リスト変数

```
list2 = [23, 12, 45, 50, 6, 10, 8]
```

を宣言・初期化して、

```
GetMaxValue = MaxValueCal( list2 )
print(" リストlist 2 の最大整数=", GetMaxValue )
```

とプログラムしなさい。

使用するデータ：

選手名	打率	ホームラン数	球団名
吉田正選手	328	23	オリックス
浅村選手	265	28	楽天
筒香選手	273	26	DeNA
中田選手	243	23	日本ハム
秋山選手	309	19	西武
坂本選手	304	33	巨人
井上選手	251	23	ロッテ
柳田選手	316	7	ソフトバンク
山田選手	274	32	ヤクルト
鈴木選手	333	25	広島
近本選手	267	9	阪神
福田選手	265	15	中日

いま、FirstQ22.pyはつぎのようになっている。

```
# 課題 FirstQ22.py ( FirstQ22_AAA.py BaseBallPlayer.dat )

from FirstQ22_AAA import ReadBaseBallPlayer as ReadBaseBallPlayer

# 名前,打率,ホームラン数,球団
Data0 = ReadBaseBallPlayer()
for row in Data0: print( row )

# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
```

Data0[]には、つぎのようにランダムに**何個か**データが入っている。

```
"吉田正選手,328,23,オリックス"
"浅村選手,265,28,楽天"
"筒香選手,273,26,DeNA"
" . . . . . "
```

プログラム中の★印の間に

```
# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
```

下記のプログラムを作成しなさい。

#### 設問 1：

Data0のデータを、Data00=[]に

Data00[ ["吉田正選手", 328, 23, "オリックス"], ["浅村選手", 265, 28, "楽天"], … ]  
のように、代入しなさい。

## 設問2：

打率、ホームランの上位3名の名前、打率、ホームラン数、所属球団をそれぞれ求めて表示しなさい。  
ただし、同率、同本の場合には、すべて表示しなさい。

## クラス

### 1 設問1 FirstQClass00.py 設問1・設問2の解答：：：FirstQClass55.py

面積を求める、クラス: Area() を作成し、検証しなさい。

なお、メソッドでは、当該面積を求めて、面積を表示しなさい。

面積：

対称物体	関数名	計算式
円 Circle：	En	半径(Hankei) x 半径 x パイ(np.pi)
長方形 Square：	Shikaku	縦(tate) x 横(yoko)
三角形 Triangle：	Sankaku	底辺(teihen) x 高さ(takasa)

パイ：np.pi

検証：

オブジェクト：area
円の面積： 半径：10
長方形の面積： 縦：4、横：5
三角形の面積： 底辺：4、高さ：10

### # 課題 FirstQClass00.py

```
import numpy as np

# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
class AreaClass:
    def __init__(self):
        print("+++++コンストラクタ+++++")
        pass
    def __str__(self):
        return "クラス名=AreaClass(面積)"
    def __del__(self):
        print("+++++解放+++++")

    def En(self): #円：半径 x 半径 x パイ
        pass
    def Shikaku(self): #長方形：縦 x 横
        pass
    def Sankaku(self): #三角形：底辺 x 高さ
```

引数は 省略してある

[illegible]

解答：FirstQClass11.py である。

2 設問2 FirstQClass33.py

体積を求める、クラス: Area\_Capacity\_Class() を作成し、検証しなさい。

なお、メソッドでは、当該体積を求めて、体積を表示しなさい。

体積：

対称物体	関数名	計算式
直方体 Rectangular :	TyokuhouTai	底面積(長方形:tate, yoko) x 高さ(takasa)
三角錐 Cone :	SankakuSui	底面積(三角形(teihen, teiTakasa)) x 高さ(takasa)/ 3
円柱 Pillar :	EnTyu	底面積 (円(hankei)) x 高さ(takasa)
球 Sphere:	KyuTai	$(4/3) \times \text{パイ} \times \text{半径(hankei)}^3$

検証：

オブジェクト : areaCapacity
直方体の体積 : 縦 : 4、横 5、高さ : 1 0
三角錐の体積 : 底辺 : 4、底辺高さ : 5、高さ : 1 0
円柱の体積 : 半径 : 1 0、高さ : 1 0
球体の体積 : 半径 : 1 0

```
# 課題 FirstQClass33.py
import numpy as np

# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
class AreaClass:
    def __init__(self):
        print("+++++コンストラクタ+++++")
        pass
    def __str__(self):
        return "クラス名=AreClass(面積)"
    def __del__(self):
        print("+++++解放+++++")

    def En(self): #円：半径 x 半径 x パイ
        pass
    def Shikaku(self): #長方形：縦 x 横
        pass
    def Sankaku(self): #三角形：底辺 x 高さ
        pass

# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
class Area_Capacity_Class:
    # -----体積：volume、capacity
    def TyokuhouTai(sel ): #直方体：底面積(長方形) x 高さ
        pass
    def SankakuSui(self ): #三角錐：底面積(3角形) x 高さ/ 3
        pass
    def EnThu(self ): #円柱：底面積（円） x 高さ
```

```
import numpy as np
```

[illegible]

```
class AreaClass:
```

```
def __init__(self):
```

```
print("+++++コンストラクタ+++++")
```

pass

```
def __str__(self):
```

```
return "クラス名=AreClass(面積)"
```

```
def __del__(self):
```

```
print("+++++解放+++++")
```

```
def En(self): #円 : 半径 x 半径 x パイ
```

pass

```
def Shikaku(self): #長方形：縦 x 横
```

pass

```
def Sankaku(self): #三角形：底辺 x 高さ
```

pass

# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

```
class Area_Capacity_Class:
```

# -----体積 : volume、capacity

```
def TyokuhouTai(sel): #直方体：底面積(長方形) x 高さ
```

pass

```
def SankakuSui(self): #三角錐：底面積(3角形) x 高さ/ 3
```

pass

```
def EnThu(self): #円柱：底面積（円） x 高さ
```

[illegible]

解答：FirstQClass44.py である。

### 3 設問3 FirstQClass44.py のクラスの継承

スーパークラス：AreaClaa  
サブクラス：Area\_Capacity\_Class

クラスの継承を行いなさい。

4 設問 4 FirstQClass44.py のクラスの継承 ( import )

```
import : : スーパークラス : AreaClaa
        サブクラス : Area_Capacity_Class
```

クラスの継承( import )を行いなさい。

```

# 課題 FirstQClass77.py ( FirstQClass77_AAA.py BaseBallPlayer.dat )

from FirstQClass11_AAA import ReadBaseBallPlayer as ReadBaseBallPlayer

# ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
class Player_00:
    def __init__(self):
        print("+++++++コンストラクタが呼び出された+++++")
        self.Data0 = ReadBaseBallPlayer()
        print(); print()

    def __str__(self):
        return "+++++++クラス名=Player_00+++++"

    def __del__(self):
        print("+++++++オブジェクト:Playerの解放+++++")

    def Make2D_Data(self):
        # ++++++設問 1 ++++++
        self.Data00 = []
        for row in self.Data0:      # row <= "柳田選手,316,7,ソフトバンク"
            list0 = row.split( "," )
            list0[1] = int( list0[1] )
            list0[2] = int( list0[2] )
            self.Data00.append( list0 )

    def HitRatio(self):
        # +++++名前[0] reverse = False
        self.Data00.sort(key=lambda x:x[0] )
        # +++++打率[1] reverse = True
        print("+++++++打 率+++++")
        self.Data00.sort(key=lambda x:x[1], reverse = True )
        count = 0
        nn = -1
        mm = 0
        for row in self.Data00:
            if nn != row[1]:
                nn = row[1]
                mm += 1
                count += 1
                if count== 4 or mm >= 4:
                    break
            print( row )
        else:
            mm += 1

```



[illegible]

