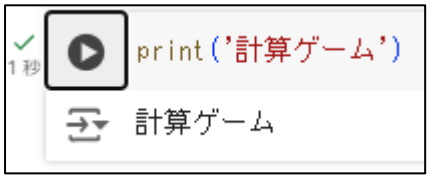
プログラミング演習３（Python）③　実践問題

■計算問題を出すプログラムを作る

　２つの値を足した数を答えさせる問題を出し、５問回答させて、正解した数を表示するゲームを作り、徐々にゲーム性を高めていく。

１－１　タイトルを表示する

と入力して、  をクリックすると



「(’)と「’)」の間の文字が表示される。

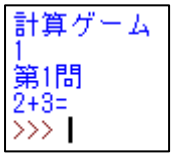
この要領で、コードを入力して、ボタンを押して実行結果を見ていく

１－２　変数に値を入れて結果を表示する（変数）

py3-2.py

```
001 toiban=1
002 kazu1=2
003 kazu2=3
004 print('計算ゲーム')
005 print(toiban)
006 print(f'第{toiban}問')
007 print(f'{kazu1}+{kazu2}=')
```

変数に値を入れると



と表示される

文字列を表示させるときは
print(’ ～ ’)
と
文字を「’」で挟んで入れ、
数値や変数の中身を表示させるときは
print(～)
と「’」で挟まずに変数や数値を入れる。

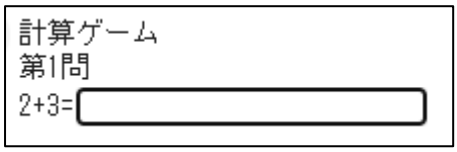
文字列と変数が混在しているときは
print(f’{変数}文字列’)
として、変数を{}で挟んで埋め込む

３－３　答えを入力できるようにして、解答を表示させる（INPUT）

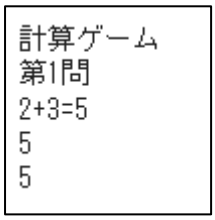
py3-3.py

```
001 toiban=1
002 kazu1=2
003 kazu2=3
004 print('計算ゲーム')
005 print(f'第{toiban}問')
006 kotae=input (f'{kazu1}+{kazu2}=')
007 print(kotae)
008 print(kazu1 + kazu2)
```

を実行すると



と表示され、５を入力して Enter を押すと



と表示される

３－４　足す値をランダムにする（ライブラリ関数）

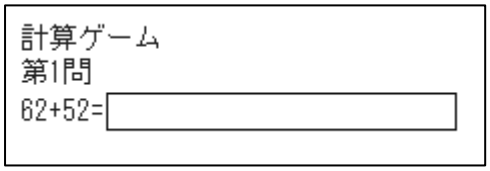
関数には「組み込み関数」と「ライブラリ関数」の２種類あり、今回はライブラリ関数を使う。

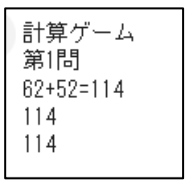
ライブラリ関数をインポートするには
import モジュール名
と記述し、ランダムな数値を作る関数を含むライブラリは「random」なので、
inport random と入力する
ライブラリには、そのジャンルの関数が複数種類含まれ、それらの関数を使うコードの書式は
モジュール名・関数名（引数）
となる
ランダムな整数を作るには、
random.randrange(範囲の上限値)
となり、実行すると０から指定した値の範囲でランダムな整数が生成される
例えば範囲の上限値に５を指定すると、０～４の範囲でランダムな整数が表示される。
ここでは１～９９までの整数を表示させたいので、
random.randrange(99)
とすると、０～９８までの整数になるので、
random.randrange(99)+1
とする
変数の設定でこれらのコードを入れると

py3-4.py

```
001 import random
002 toiban=1
003 kazu1=random.randrange(99)+1
004 kazu2=random.randrange(99)+1
005 print('計算ゲーム')
006 print(f'第{toiban}問')
007 kotae=input (f'{kazu1}+{kazu2}=')
008 print(kotae)
009 print(kazu1 + kazu2)
```

実行すると





と表示される

３－５　採点をする（IF 文）

入力された回答と解答を照らし合わせて、同じだったら正解、間違っていたら間違いと表示させるときは、
分岐のコードを入れる

書式は
If 条件式:
　成立する場合の処理
else:
　成立しない場合の処理
となる。
インデント（左からの余白）を入れないと、if 文が成立しないので注意する。

条件式の条件は「比較演算子」という仕組みを使って書き、２つの値を比較して、成立するか成立しないかを判定する。

表　主な比較演算子

演算子	意味
==	等しい
!=	等しくない
>	大きい
>=	以上
<	小さい
<=	以下

書式は
値１　比較演算子　値２
となり、
例をあげると
1 == 1　　の結果は　　True
1 != 1　　の結果は　　False
1 > 2　　の結果は　　False
1 < 2　　の結果は　　True
となる。
Print(1 == 1)

を実行すると
True が表示される
計算ゲームでは
kotae の値と kazu1＋kazu2 の値が同じだったら正解、そうでなかったら不正解と表示すればいいのだから、条件式を kotae == (kazu1 + kazu2)として、True のところに正解と表示するコード、False のところに不正解と表示するコードを書けばいいので、

py3-5.py

```
001 import random
002 toiban=1
003 kazu1=random.randrange(99)+1
004 kazu2=random.randrange(99)+1
005 print('計算ゲーム')
006 print(f'第{toiban}問')
007 kotae=input (f'{kazu1}+{kazu2}=')
008 print(kotae)
009 print(kazu1 + kazu2)
010 if kotae == (kazu1+kazu2):
011   print('正解')
012 else:
013   print('不正解')
```

として実行すると

```
計算ゲーム
第1問
86+2=88
88
88
不正解
```

となる
これは input 関数の戻り値は文字列と決められているので、kotae の 88 は文字列の 88 なので違いものと判定されて不正解となる。
なので、
kotae=input (f'{kazu1}+{kazu2}=')
で kotae に文字列が入らないように
数値に変換する int 関数を使う
int 関数の書式は

int(文字列)
なので、
kotae=int(input (f'{kazu1}+{kazu2}='))
と変更すればよい。
実行すると

```
計算ゲーム
第1問
94+76=170
170
170
正解
```

同じ値だったら正解

```
計算ゲーム
第1問
25+49=999
999
74
不正解
```

間違っていたら不正解が表示される

３－６　繰り返し処理で５問に増やす（for 文）

第１問の答え合わせが終わったら、第２問の問題が表示されて、第５問まで繰り返す処理を考える。
２問目になったら、toiban の値を１つ増やし後は同じ処理をする。
反復のコードは

```
for 変数名　in range(回数)：
  処理
```

で表す。処理の前のインデントを忘れないように注意する。

```
for toiban in range(5)
  print (toiban)
```

とすると、

```
0
1
2
3
4
```

と表示される。toiban は 1,2,3,4,5 としたいので、
まず、5 まで繰り返すように

```
for toiban in range(6)
  print (toiban)
```

```
0
1
2
3
4
5
```

として、１から始めるように

```
for toiban in range(1,6):
  print(toiban)
```

```
1
2
3
4
5
```

と記述する。

タイトルの「計算ゲーム」の文字列は初めだけ入れればいいので、繰り返し文の前に記述して、

py3-6.py

```
001  import random
002  print('計算ゲーム')
003  for toiban in range(1,6):
004    kazu1=random.randrange(99)+1
005    kazu2=random.randrange(99)+1
006    print(f'第{toiban}問')
007    kotae=int(input
008  (f'{kazu1}+{kazu2}='))
009    print(kotae)
010    print(kazu1 + kazu2)
011    if kotae == (kazu1 + kazu2):
012      print('正解')
013    else:
014      print('不正解')
```

と入力して、実行すると右のようになる。

実行した結果

```
計算ゲーム
第1問
3+34=37
37
37
正解
第2問
2+55=57
57
57
正解
第3問
6+86=92
92
92
正解
第4問
33+10=43
43
43
正解
第5問
86+46=999
999
132
不正解
```

３－７　結果の表示

最後に５問中何問正解できたか表示させるために、正解した数を変数 seikaisu を用意して、正解するとその数が１つ増えるようにする。
初めに seikaisu=0 として、正解するたびに数が１ずつ増えるようにすればいいので、if 文の同じだった場合の処理に seikaisu += 1 を入れる。seikaisu = seikaisu + 1 でもいいが、「+=」の演算子を使った方が早く入力ができる。なお１ずつ減らしたいときは a -= 1 と入力する

py3-7.py

```
001 import random
002 seikaisu = 0
003 print('計算ゲーム')
004 for toiban in range(1,6):
005   kazu1=random.randrange(99)+1
006   kazu2=random.randrange(99)+1
007   print(f'第{toiban}問')
008   kotae=int(input (f'{kazu1}+{kazu2}='))
009   print(kotae)
010   print(kazu1 + kazu2)
011   if kotae == (kazu1 + kazu2):
012     print('正解')
013     seikaisu += 1
014   else:
015     print('不正解')
016 print(f'{toiban}問中{seikaisu}問正解しました')
```

と入力して、実行すると右のようになる。

実行した結果

```
計算ゲーム
第1問
63+81=144
144
144
正解
第2問
6+66=72
72
72
正解
第3問
86+34=120
120
120
正解
第4問
9+33=42
42
42
正解
第5問
7+8=15
15
15
正解
5問中5問正解しました
```

３－８　結果の詳細の表示（リスト）

　採点のたびに結果をリストに保管して、最後に表示する

空のリストは

　リスト　= []　　または　　リスト　=　list()

と書ける

このリストに値を代入するときは、

要素を追加していく append(値)でもいいし、場所を指定して値を入れていく insert(位置,値)を使っても書ける

ここでは、○と×を記録するためのリスト marubatu を作るために

　marubatu = []

と初めに書いておき、そのリスト内の場所を指定して値を入れる insert メソッドを使って○または×を記録していく。

Insert メソッドの書式は

　リスト.insert(位置, 要素)

と書くので、正解したときに、marubatu リストの要素番号 toiban の場所に○を代入するには

marubatu.insert(toiban,'○')

と書き、

不正解の時は、

marubatu.insert(toiban,'×')

と書く。

なお、リストの要素番号は０から始まるが、toiban 変数は１から始まるので、marubatu[1]の場所から記録されていき、matubatu[0] は空要素になる。また、リスト名[要素番号]=値　という書き方は、すでに値が入っているときに上書きする書き方なので、ここでは使えない。

次にリストの結果を表示する場合、リストと for 文を組み合わせた書式の

for 変数名 in リスト名:

　処理

が使えるので、

```
for listout in marubatu:
  print(listout)
```

と書いて、結果の詳細を出力する。

初めに問題数を設定できるようにコードを加えて実行すると、

py3-8.py

```
001 import random
002 seikaisu = 0
003 marubatu = []
004 print('計算ゲーム')
005 mondaisu=int(input ('問題数は？'))
006 for toiban in range(1,mondaisu+1):
007   kazu1=random.randrange(99)+1
008   kazu2=random.randrange(99)+1
009   print(f'第{toiban}問')
010   kotae=int(input (f'{kazu1}+{kazu2}='))
011   print(kotae)
012   print(kazu1 + kazu2)
013   if kotae == (kazu1 + kazu2):
014     print('正解')
015     seikaisu += 1
016     marubatu.insert(toiban, '○')
017   else:
018     print('不正解')
019     marubatu.insert(toiban, '×')
020 for listout in marubatu:
021   print(listout)
022 print(f'{toiban}問中{seikaisu}問正解しました')
```

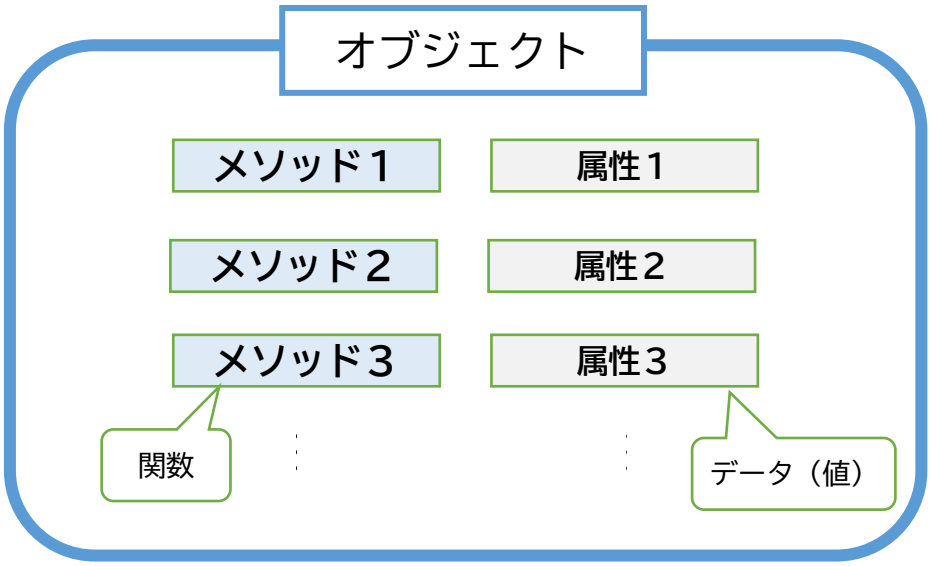
実行した結果

```
計算ゲーム
問題数は？3
第1問
12+30=42
42
42
正解
第2問
68+47=115
115
115
正解
第3問
46+94=999
999
140
不正解
○
○
×
3問中2問正解しました
```

と入力して、実行すると右のようになる。

[＋α]オブジェクト指向

　Python にはオブジェクトの仕組みが使われており、これまで登場した文字列とリストはオブジェクトの形式になっている。オブジェクトはデータと関数をまとめて扱う仕組みで、データとは数値や文字列などで、関数とはよく使う処理をまとめてソースコードを簡潔に書けるようにしたものである。イメージとしては下図のようになる。



メソッドの基本的な書式は

オブジェクト．メソッド名（引数）
となり、
marubatu.insert(toiban,'○')
と書いた場合、
matubatu というリストのオブジェクトに対して、insert というメソッドを使っていて、そのメソッドの仕様により、第１引数のリストの要素番号 toiban の場所に、第２引数の「○」という文字列を入れる。という操作をしている。

３－９　四則演算のそれぞれの計算をするオリジナルの関数を作って、結果を出力する。(組み込み関数)

　ここでは、初めに四則演算のどの演算をするのか決めて、その演算をするための関数を呼び出して、計算結果を出す。

関数の書式は

```
  def 関数名（引数名１,引数名２…）
      処理
      return 戻り値
```

と書いて定義をして、

```
  関数名（引数）
```

と書いて実行をする。

初めに演算の種類を判別できるように、enzan 変数を用意して、演算に合わせて、足し算なら１，引き算なら２というように値を設定する。その設定の値に合わせて、呼び出す関数を変える。最後の結果で使う seikaisu や toiban の変数は関数の中でも外でも使っているので、関数で global seikaisu,toiban と宣言してグローバル変数として定義している。割り算の答えは割り切れなくことがあるので、商（整数部）のみ答えさせている。商は「//」の演算子で求められ、他に商を取り出す方法として、math モジュールをインポートして math.modf()などで読み込ませてから取り出す方法など、いくつかある。

Py3-9.py

```
001 import random
002 seikaisu = 0
003 toiban = 1
004 marubatu = []
005 def wa():
006   global seikaisu,toiban
007   for toiban in range(1,mondaisu+1):
008     kazu1=random.randrange(99)+1
009     kazu2=random.randrange(99)+1
010     print(f'第{toiban}問')
011     kotae=int(input (f'{kazu1}+{kazu2}='))
012     print(kotae)
013     print(kazu1 + kazu2)
014     if kotae == (kazu1 + kazu2):
015       print('正解')
016       seikaisu += 1
017       marubatu.insert(toiban, '○')
018     else:
```

```
019         print('不正解')
020         marubatu.insert(toiban, '×')
021 def sa():
022   global seikaisu,toiban
023   for toiban in range(1,mondaisu+1):
024     kazu1=random.randrange(99)+1
025     kazu2=random.randrange(99)+1
026     print(f'第{toiban}問')
027     kotae=int(input (f'{kazu1}-{kazu2}='))
028     print(kotae)
029     print(kazu1 - kazu2)
030     if kotae == (kazu1 - kazu2):
031       print('正解')
032       seikaisu += 1
033       marubatu.insert(toiban, '○')
034     else:
035       print('不正解')
036       marubatu.insert(toiban, '×')
037 def seki():
038   global seikaisu,toiban
039   for toiban in range(1,mondaisu+1):
040     kazu1=random.randrange(10)+1
041     kazu2=random.randrange(10)+1
042     print(f'第{toiban}問')
043     kotae=int(input (f'{kazu1}*{kazu2}='))
044     print(kotae)
045     print(kazu1 * kazu2)
046     if kotae == (kazu1 * kazu2):
047       print('正解')
048       seikaisu += 1
049       marubatu.insert(toiban, '○')
050     else:
051       print('不正解')
052       marubatu.insert(toiban, '×')
053 def shou():
054   global seikaisu,toiban
055   print('整数部のみ（例；3.5なら3)')
056   for toiban in range(1,mondaisu+1):
```

```
057     kazu1=random.randrange(10)+1
058     kazu2=random.randrange(10)+1
059     print(f'第{toiban}問')
060     kotae=int(input (f'{kazu1}/{kazu2}の商は？'))
061     print(kotae)
062     print(int(kazu1 // kazu2))
063     if kotae == int(kazu1 // kazu2):
064       print('正解')
065       seikaisu += 1
066       marubatu.insert(toiban, '○')
067     else:
068       print('不正解')
069       marubatu.insert(toiban, '×')
070 print('計算ゲーム')
071 mondaisu=int(input ('問題数は？'))
072 enzan=int(input ('どの種類？(和は 1,差は 2,積は 3,商は 4)'))
073 if enzan == 1:
074   wa()
075 elif enzan == 2:
076   sa()
077 elif enzan == 3:
078   seki()
079 elif enzan == 4:
080   shou()
081 else:
082   print('1-4 の数値を入力してください')
083 for listout in marubatu:
084   print(listout)
085 print(f'{toiban}問中{seikaisu}問正解しました')
```

として、実行すると次のようになる。

実行した結果

```
計算ゲーム
問題数は？2
どの種類？(和は1,差は2,積は3,商は4)4
整数部のみ（例；3.5なら3)
第1問
8/1の商は？8
8
8
正解
第2問
1/6の商は？0
0
0
正解
○
○
2問中2問正解しました
```

【課題】
これまで作ってきたプログラムを改良して、どのように改良したか、わかりやすく説明しなさい。

例えば、演算の種類に合わせて、関数を呼び出して、関数側でほとんどの処理をしているが、入力と出力をメインのコードで処理させおいて、関数側に値を受け渡して、演算の結果を返すようにすると、全体的に簡潔なコードが少なくなる。

（参考）和を出す関数を実装したソース

```
def wa(kanx,kany):
  kanz = kanx + kany
  return kanz

x = int(input ('x='))
y = int(input ('y='))
z = wa(x,y)
print(z)
```

```
x=2
y=3
5
```

他にも、input に制限時間を設けてゲーム性を高めたり、詳細のリストや結果をファイルに書き出して、過去の履歴を残せるようにして、成長の度合いを確認したり、よく間違う場所を把握したりすることができるようにしてもいいだろう。

参考　立山秀利著「高校生からの Python 入門」（株式会社ジャムハウス）
※計算ドリルアプリより本教材を着想した