別紙4－55

実習「シミュレーション3（トランプを使った得点の計算）」のデータ（疑似言語）

```
i ← 1
score ← 0
i (<= 3 の間くりかえす :
    | もし 乱数() < 0.5 ならば :
    |     | 表示する("ハート")
    |     | score ← score + 1
    | そうでなければ :
    |     L 表示する("スペード")
    L i ← i + 1
表示する(score, "点")
```

※ここでは，下記の「乱数」という関数を作成し使用する。
乱数 () …0以上1未満のランダムな数を出力する関数

別紙4－56

複合代入演算子の種類と使用例

演算子	意味	使用例	使用例の意味
+=	たし算	a+=3	変数aに3をたす
-=	ひき算	a-=5	変数aから5をひく
*=	かけ算	a*=11	変数aに11をかける
/=	わり算	a/=3	変数aを3でわる

- 複合代入演算子は，累算代入演算子ともよばれる。
- 代入演算子と算術演算子を使った記述を短くしたいときに使用される。
  たとえば，「a+=3」は「a=a+3」と同じ意味である。

別紙4－57

実習「シミュレーション4（円の面積）」のデータ（Python）

```
import random
n = 0
i = 0
a = 0.5
b = 0.5
r = 0.5
while i < 10000:
    x = random.random();
    y = random.random();
    if ((x-a)**2 + (y-b)**2) < (r**2):
        n = n + 1;
    i = i + 1
print("(", n, ",", i, ")");
print("(", 3.14 * r ** 2, ",", (2*r) ** 2, ")");
```

別紙4－58

実習「シミュレーション4（円の面積）」のデータ（疑似言語）

```
import random
n ← 0
i ← 0
a ← 0.5
b ← 0.5
r ← 0.5
i < 10000 の間くりかえす :
    | x ← 乱数()
    | y ← 乱数()
    | もし ((x-a)**2 + (y-b)**2) < (r**2) ならば :
    |     L n ← n + 1
    L i ← i + 1
表示する("(", n, ",", i, ")")
表示する("(", 3.14 * r ** 2, ",", (2*r) ** 2, ")")
```

別紙4－59

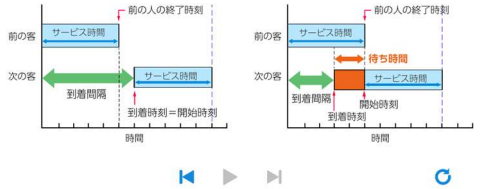
複合代入演算子の種類と使用例

演算子	意味	使用例	使用例の意味
+=	たし算	a+=3	変数aに3をたす
-=	ひき算	a-=5	変数aから5をひく
*=	かけ算	a*=11	変数aに11をかける
/=	わり算	a/=3	変数aを3でわる

- 複合代入演算子は，累算代入演算子ともよばれる。
- 代入演算子と算術演算子を使った記述を短くしたいときに使用される。
  たとえば，「a+=3」は「a=a+3」と同じ意味である。

別紙4－60

● 待ち行列ができない場合／待ち行列ができる場合



別紙4－61

実習「シミュレーション5（待ち行列）」のデータ（Python）

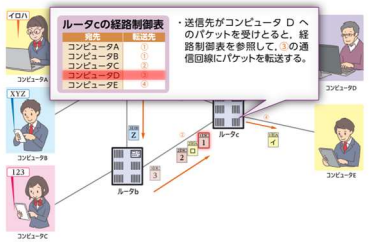
```
import random

interval = arrivaltime = starttime = endtime = waittime = 0
servicetime = 5

print("(客，到着間隔，到着時間，開始時間，サービス時間，終了時刻，待ち時間)")

for i in range(10):
    if i == 0:
        interval = 0
    else:
        interval = random.randint(1, 12)
    arrivaltime = arrivaltime + interval
    starttime = max(arrivaltime, endtime)
    endtime = starttime + servicetime
    waittime = starttime - arrivaltime
    print("(", i+1, interval, arrivaltime, starttime, servicetime, endtime, waittime, ")")
```

別紙4－62

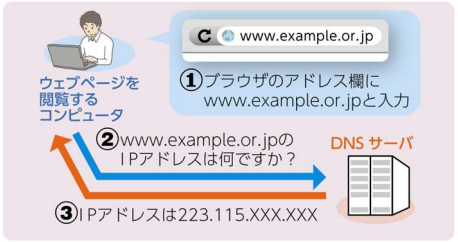
別紙4－63

別紙5－1

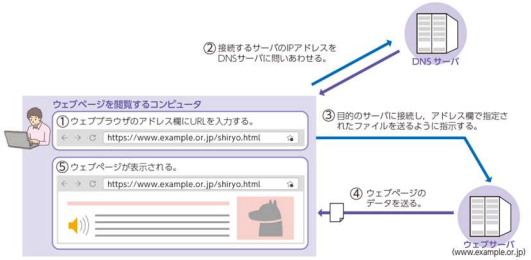
別紙5－2



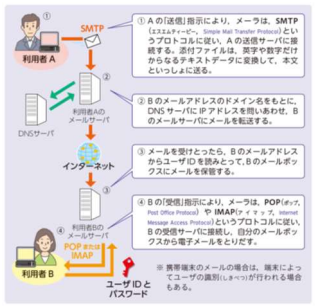
別紙5－3



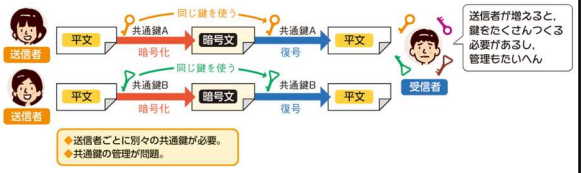
別紙5－4



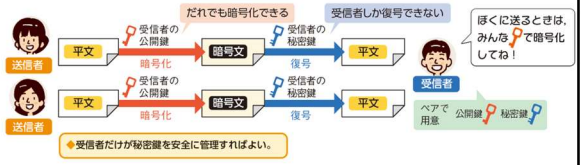
別紙5－5



別紙5－6



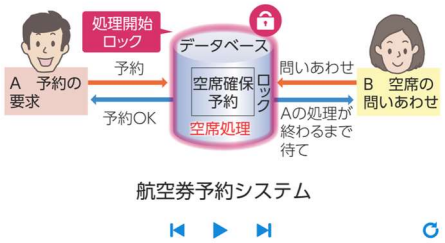
別紙5－7



別紙5－8

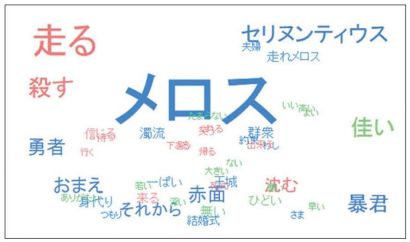
別紙5－9

別紙5－10

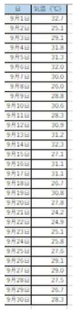


別紙5－11

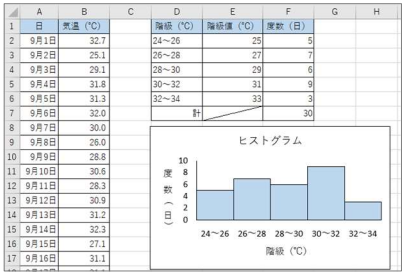
別紙5－12



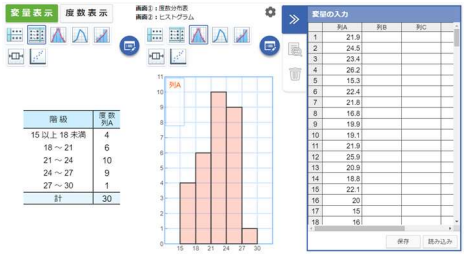
別紙5－13



別紙5－14



別紙5－15



別紙5－16

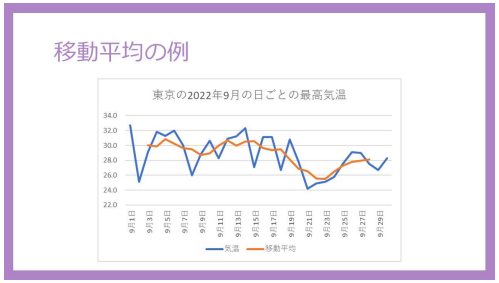
A町における商品の価格	
店舗	価格（円）
店舗1	260
店舗2	270
店舗3	280
店舗4	280
店舗5	300

B町における商品の価格	
店舗	価格（円）
店舗1	100
店舗2	260
店舗3	270
店舗4	280
店舗5	280
店舗6	280

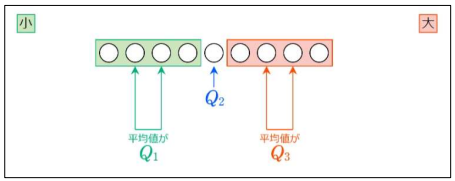
別紙5－17

	A	B	C	D	E
1	A町における商品の価格			B町における商品の価格	
2	店舗	価格（円）		店舗	価格（円）
3	店舗1	260		店舗1	100
4	店舗2	270		店舗2	260
5	店舗3	280		店舗3	270
6	店舗4	280		店舗4	280
7	店舗5	300		店舗5	280
8				店舗6	280
9					
10	平均値	278		平均値	245
11	中央値	280		中央値	275

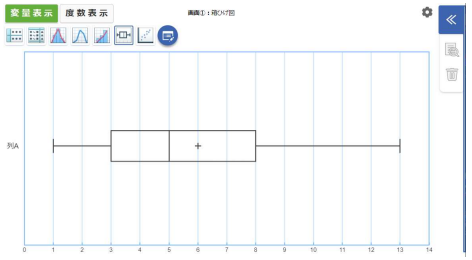
別紙5－18



別紙5－19



別紙5－20



別紙5－21

元データ

年代	評価
中学生	2
高校生	3
大学生	2
大学生	3
高校生	3
大学生	3
中学生	1
中学生	1
高校生	2
高校生	3

別紙5－22

	A	B	C	D	E	F	G	H
1	元データ			クロス集計表				
2	年代	評価		評価	人数	年代		
3	中学生	2				中学生	高校生	大学生
4	高校生	3		1 悪い	2	2	0	0
5	大学生	2		2 普通	3	1	1	1
6	大学生	3		3 よい	5	0	3	2
7	高校生	3						
8	大学生	3						
9	中学生	1						
10	中学生	1						
11	高校生	2						
12	高校生	3						

別紙5－23

10人の漢字テストの得点（A組）

生徒	得点（点）	（得点－平均値）
生徒1	9	4
生徒2	3	16
生徒3	4	9
生徒4	10	9
生徒5	10	9
生徒6	5	4
生徒7	7	0
生徒8	9	4
生徒9	10	9
生徒10	3	16
合計	70	80

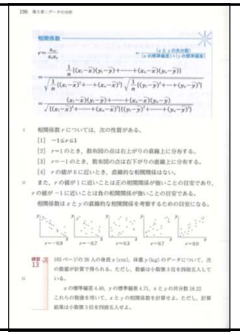
10人の漢字テストの得点（B組）

生徒	得点（点）	（得点－平均値）
生徒1	6	1
生徒2	10	9
生徒3	7	0
生徒4	7	0
生徒5	5	4
生徒6	4	9
生徒7	9	4
生徒8	10	9
生徒9	5	4
生徒10	7	0
合計	70	40

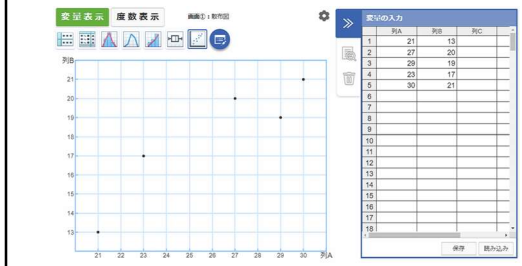
別紙5－24

	A	B	C	D	E	F	G
1	10人の漢字テストの得点（A組）				10人の漢字テストの得点（B組）		
2	生徒	得点（点）	（得点－平均値）		生徒	得点（点）	（得点－平均値）
3	生徒1	9	4		生徒1	6	1
4	生徒2	3	16		生徒2	10	9
5	生徒3	4	9		生徒3	7	0
6	生徒4	10	9		生徒4	7	0
7	生徒5	10	9		生徒5	5	4
8	生徒6	5	4		生徒6	4	9
9	生徒7	7	0		生徒7	9	4
10	生徒8	9	4		生徒8	10	9
11	生徒9	10	9		生徒9	5	4
12	生徒10	3	16		生徒10	7	0
13	合計	70	80		合計	70	40
14							
15	平均値	7			平均値	7	
16	分散	8			分散	4	
17	標準偏差	2.8			標準偏差	2.0	

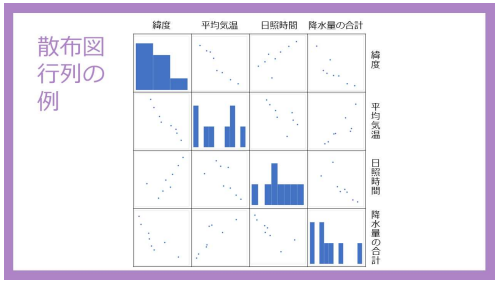
別紙5－25



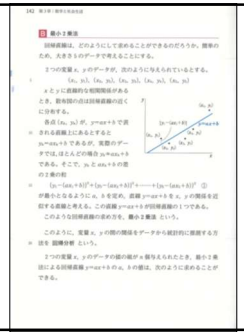
別紙5－26



別紙5－27



別紙5－28

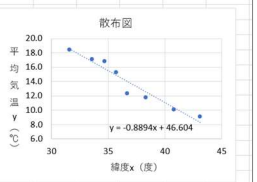


別紙5－29

	A	B	C	D	E	F	G	H	I
1	地点	札幌	青森	仙台	東京	長野	大阪	高知	鹿児島
2	緯度x（度）	43.1	40.8	38.3	35.7	36.7	34.7	33.6	31.6
3	平均気温y（℃）	9.1	10.1	11.8	15.3	12.3	16.8	17.1	18.4

別紙5－30

	A	B	C	D	E	F	G	H	I	J
1	各地点の緯度と2022年4月の平均気温									
2	地点	緯度x（度）	平均気温y（℃）							
3	札幌	43.1	9.1							
4	青森	40.8	10.1							
5	仙台	38.3	11.8							
6	東京	35.7	15.3							
7	長野	36.7	12.3							
8	大阪	34.7	16.8							
9	高知	33.6	17.1							
10	鹿児島	31.6	18.4							
11										
12	相関係数	-0.97								
13										
14										



別紙5－31

別紙5－32

別紙6−1

巻末実習2「プログラミング（1）」のデータ（Python）

④プログラムを作成する

```
motonosu = 2                    #平方根の近似値を求めたい値を2として代入する
kuwaerusu = 0.00001             #加える数を0.00001として代入する
heihokon1 = 0                   #平方根の近似値を，初期値を0として代入する
heihokon2 = 0                   #平方根の近似値を，初期値を0として代入する

while heihokon2**2 < motonosu:  #2乗した数が求めたい数より小さいときくりかえし
    heihokon1 = heihokon2       #くりかえしで小さい数をたす前の数を変数に代入する
    heihokon2 += kuwaerusu      #小さな数をたす

print(heihokon1)                #近似値が含まれる範囲の下限の数を表示
print(heihokon2)                #近似値が含まれる範囲の上限の数を表示
```

⑥求める値の精度をあげる

```
import time                  #timeモジュールをインポート

time1 = time.time()          #プログラム開始時の時間を代入する
motonosu = 2
kuwaerusu = 0.00001
heihokon1 = 0
heihokon2 = 0

while heihokon2**2 < motonosu:
    heihokon1 = heihokon2
    heihokon2 += kuwaerusu
```

別紙6−2

巻末実習2「プログラミング（1）」のデータ（疑似言語）

●手順4

```
motonosu = 2
kuwaerusu = 0.00001
heihokon1 = 0
heihokon2 = 0

heihokon2**2 < motonosu の間くりかえす：
  | heihokon1 = heihokon2
  L heihokon2 += kuwaerusu

表示する(heihokon1)
表示する(heihokon2)
```

●手順6

```
time1 = 時刻()
motonosu = 2
kuwaerusu = 0.00001
heihokon1 = 0
heihokon2 = 0

heihokon2**2 < motonosu の間くりかえす：
  | heihokon1 = heihokon2
  L heihokon2 += kuwaerusu
```

別紙6−3

巻末実習3「プログラミング（2）」のデータ（Python）

①プログラムをつくる

```
def fib(n):
    if (n == 1 or n == 2):
        return 1
    else:
        return fib(n-1) + fib(n-2)

print(fib(10))
```

③プログラムを改良する

```
def fib2(n):
    mem = [0] * (n+1)
    def fib1(n):
        if (n == 1 or n == 2):
            return 1
        elif mem[n] != 0:
            return mem[n]
        else:
            mem[n] = fib1(n-1) + fib1(n-2)
            return mem[n]
    return fib1(n)

print(fib2(10))
```

別紙6−4

巻末実習3「プログラミング（2）」のデータ（疑似言語）

①プログラムをつくる

```
関数 フィボナッチ(n) を
  | もし (n == 1 or n == 2) ならば：
  |   | フィボナッチ(n) = 1
  | そうでなければ：
  |   L フィボナッチ(n) = フィボナッチ(n-1) + フィボナッチ(n-2)
と定義する

表示する(フィボナッチ(10))
```

③プログラムを改良する

```
関数 フィボナッチ２(n) を
  | Memのすべての値を0にする
  | 関数 フィボナッチ１(n) を
  |   | もし (n == 1 or n == 2) ならば：
  |   |   | フィボナッチ１(n) = 1
  |   | そうでなくもし Mem[n] != 0 ならば：
  |   |   | フィボナッチ１(n) = Mem[n]
  |   | そうでなければ：
  |   |   | Mem[n] = フィボナッチ１(n-1) + フィボナッチ１(n-2)
  |   |   L フィボナッチ１(n) = Mem[n]
  | と定義する
  | フィボナッチ２(n) = フィボナッチ１(n)
と定義する

表示する(フィボナッチ２(10))
```

別紙6−5

巻末実習4「プログラミング（3）」のデータ（Python）

```
hai = [4,2,6,8,1,0,9,3,5,7]     #整列させたい配列を定義する
taihi = 0                        #値を退避する変数の初期値を0として代入する

a = 0                            #aに0を代入する
while a <= len(hai)-1:           #整列前の配列の値(hai[0]～hai[9])を順に表示
    print(hai[a])
    a = a+1

i = 0                            #iの初期値を0として代入する
j = len(hai)-1                   #配列の要素数より1小さい数を初期値としてjに代入する
while j >= 1:                    #jが1以上の間くりかえす
    while i <= j-1:              #iがj-1以下の間くりかえす
        if hai[i] > hai[i+1]:    #配列でとなりあう値の左側の数字が大きければ，以下を実行する
            taihi = hai[i]       #左側の数字をtaihiに代入する
            hai[i] = hai[i+1]    #左側の数字の位置に右側の数字を代入する
            hai[i+1] = taihi     #右側の数字の位置にtaihiの値（もともと左側の数字）を代入する
        i = i+1                  #iを1ずつ増やす
    i = 0                        #配列の端まで比較が終わったら，iの値を0で初期化する
    j = j-1                      #jを1ずつ減らす

print("-")                       #整列前後の数列を区別する記号を入れる
```

別紙6−6

巻末実習4「プログラミング（3）」のデータ（疑似言語）

```
Hai = [4,2,6,8,1,0,9,3,5,7]
taihi = 0

a = 0
a <= 長さ(Hai)-1 の間くりかえす:
  | 表示する(Hai[a])
  L a = a+1

i = 0
j = 長さ(Hai)-1
j >= 1 の間くりかえす:
  | i <= j-1 の間くりかえす:
  |   | もし Hai[i] > Hai[i+1] ならば:
  |   |   | taihi = Hai[i]
  |   |   | Hai[i] = Hai[i+1]
  |   |   L Hai[i+1] = taihi
  |   L i = i+1
  | i = 0
  L j = j-1

表示する("-")
```

別紙6−7

巻末実習5「プログラムを用いた待ち行列のシミュレーション」のデータ（Python）

```
import random     #randomモジュールのインポート

ninzu = 10        #到着する客の人数

r_kankaku = 0     #次の客の到着間隔を，初期値を0として代入する
r_service = 0     #サービス時間を，初期値を0として代入する
p_toctime= 0      #客ごとの到着時刻を，初期値を0として代入する
p_end = 0         #前の客のサービス終了時刻を，初期値を0として代入する
p_start = 0       #客ごとのサービス開始時刻を，初期値を0として代入する
s_waitsum = 0     #待ち時間積算値を，初期値を0として代入する
i = 0             #くりかえしの範囲を決める変数iの初期値を0として代入する

while i <= ninzu-1:                         #人数分くりかえし
    print("--客番号",i,"--")                #客ごとに区切りを表示
    r_kankaku = int(random.random()*10)     #次の客の到達間隔を乱数で決める（0～9分とする）
    print("次の客の到着間隔",r_kankaku)      #次の客の到着間隔を表示
    print("到着時刻",p_toctime)              #到着時刻を表示
    r_service = 1 + int(random.random()*5)  #サービス時間を乱数で決める（1～5分とする）
    print("サービス時間",r_service)          #サービス時間を表示
    if p_toctime > p_end:                   #もし到着時刻が前の客のサービス終了時刻より遅ければ
        p_start = p_toctime                 #サービス開始時間は到着時間と同じ（待たない）
    else:                                   #そうでなければ（到着時刻が早ければ）
        p_start = p_end                     #サービス開始時刻は前の客のサービス終了時刻になる
    print("サービス開始時刻",p_start)        #サービス開始時刻を表示
    p_end = p_start + r_service             #この客のサービス終了時刻を求める
    print("サービス終了時刻",p_end)          #サービス終了時刻を表示
```

別紙6−8

巻末実習5「プログラムを用いた待ち行列のシミュレーション」のデータ（疑似言語）

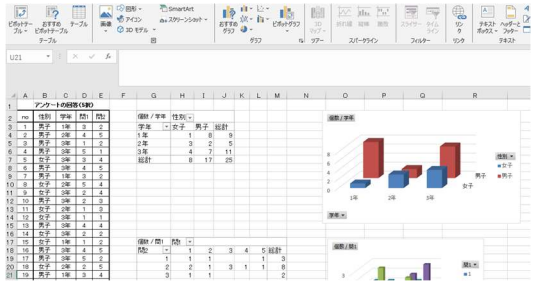
```
ninzu = 10

r_kankaku = 0
r_service = 0
p_toctime= 0
p_end = 0
p_start = 0
s_waitsum = 0
i = 0

i <= ninzu-1 の間くりかえす：
  | 表示する("--客番号",i,"--")
  | r_kankaku = 整数 (乱数()*10)
  | 表示する("次の客の到着間隔",r_kankaku)
  | 表示する("到着時刻",p_toctime)
  | r_service = 1 + 整数 (乱数()*5)
  | 表示する("サービス時間",r_service)
  | もし p_toctime > p_end ならば：
  |   | p_start = p_toctime
  | そうでなければ：
  |   L p_start = p_end
  | 表示する("サービス開始時刻",p_start)
  | p_end = p_start + r_service
  | 表示する("サービス終了時刻",p_end)
```

別紙6−9

アンケートの回答（5択）

no	性別	学年	問1	問2
1	男子	1年	3	2
2	男子	2年	4	5
3	男子	3年	1	2
4	男子	3年	5	1
5	女子	3年	3	4
6	男子	3年	4	5
7	男子	1年	3	2
8	女子	2年	5	4
9	女子	3年	2	4
10	男子	3年	2	3
11	女子	2年	1	3
12	女子	3年	1	1
13	男子	3年	4	4
14	女子	3年	2	2
15	女子	1年	1	2
16	男子	3年	4	5
17	男子	3年	5	2
18	女子	2年	2	5
19	男子	1年	3	4
20	男子	1年	4	2
21	男子	1年	1	4
22	男子	1年	2	1
23	男子	1年	4	4
24	男子	1年	1	5
25	男子	2年	3	2

別紙6－10



別紙6－11

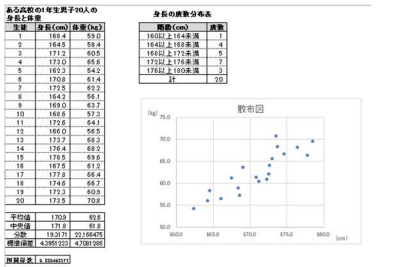
数学	英語	組性
18	13	1組
18	10	1組
17	10	1組
17	12	1組
11	10	1組
15	13	1組
14	11	1組
14	10	1組
11	8	1組
13	9	2組
17	9	2組
15	10	2組
17	10	2組
12	14	2組
12	11	2組
14	9	2組
11	12	2組
15	11	2組
11	13	2組
14	12	2組

別紙6－12

ある高校の1年生男子20人の身長と体重

生徒	身長(cm)	体重(kg)
1	168.4	59.0
2	164.5	58.4
3	171.2	60.5
4	173.0	65.6
5	162.2	54.2
6	170.8	61.4
7	172.5	62.2
8	164.2	56.1
9	169.0	63.7
10	168.6	57.3
11	172.6	64.1
12	166.0	56.5
13	173.7	68.3
14	176.4	68.2
15	178.5	69.6
16	167.5	61.2
17	177.8	66.4
18	174.6	66.7
19	172.3	60.9
20	173.5	70.8

別紙6－13



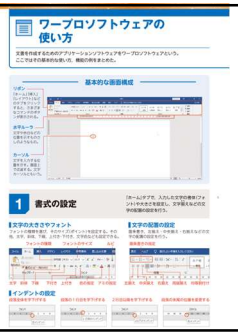
ある高校の1年生男子20人の身長と体重

生徒	身長(cm)	体重(kg)
1	168.4	59.0
2	164.5	58.4
3	171.2	60.5
4	173.0	65.6
5	162.2	54.2
6	170.8	61.4
7	172.5	62.2
8	164.2	56.1
9	169.0	63.7
10	168.6	57.3
11	172.6	64.1
12	166.0	56.5
13	173.7	68.3
14	176.4	68.2
15	178.5	69.6
16	167.5	61.2
17	177.8	66.4
18	174.6	66.7
19	172.3	60.9
20	173.5	70.8
平均値	170.9	62.6
中央値	171.8	63.2
分散	19.3171	22.166475
標準偏差	4.3951222	4.7081085

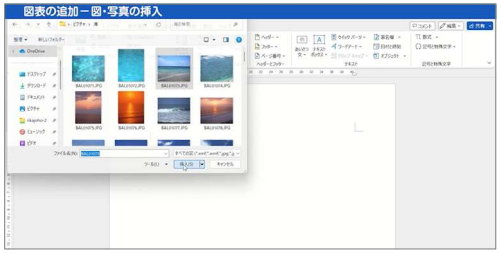
身長の度数分布表

階級(cm)	度数
160以上164未満	1
164以上168未満	4
168以上172未満	5
172以上176未満	7
176以上180未満	3
計	20

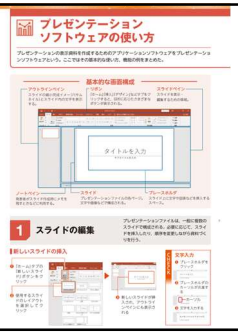
別紙6－14



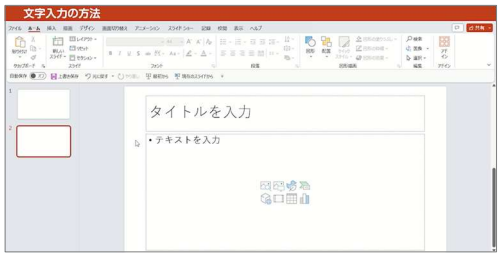
別紙6－15



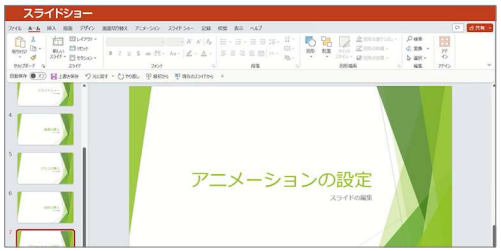
別紙6－16



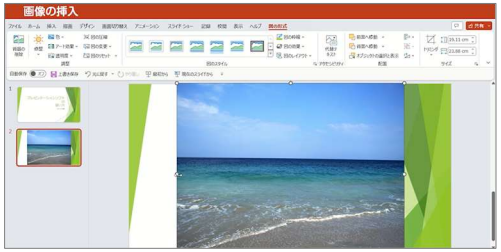
別紙6－17



別紙6－18



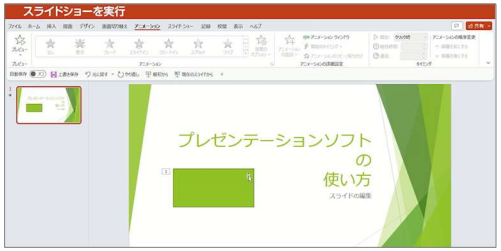
別紙6－19



別紙6－20

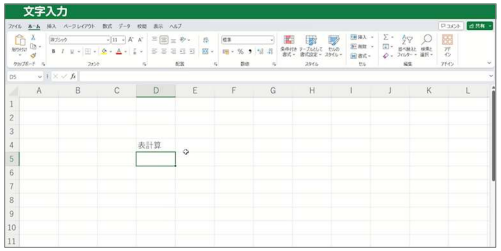
別紙6－21

別紙6－22

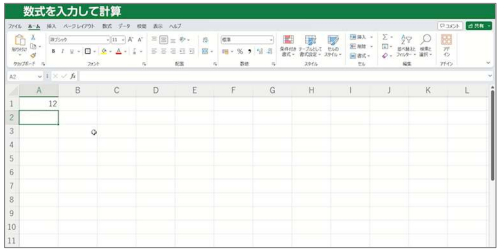


別紙6－23

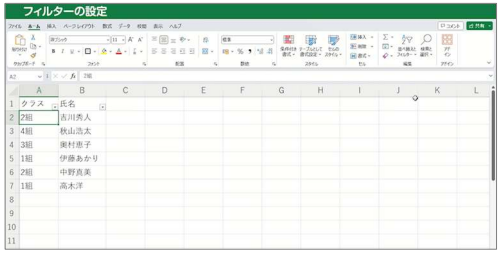
別紙6－24



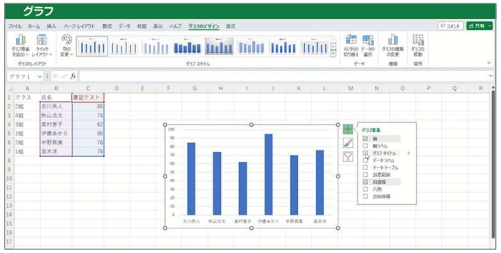
別紙6－25



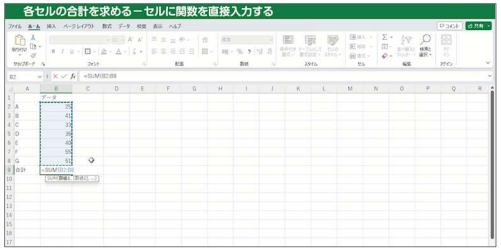
別紙6－26



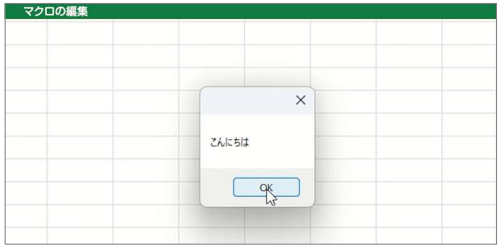
別紙6－27



別紙6－28



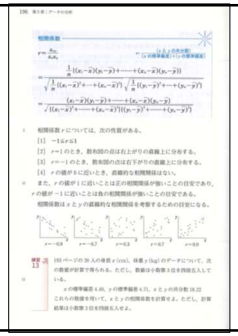
別紙6－29



別紙6－30

```
<html>
<head>
<title>〇〇のウェブページ</title>
</head>
<body>
<h1>私の好きなもの</h1>
ここでは，私の好きなものを紹介します。<br>これから3つ紹介します。
<p>まず，6月の花であるアジサイについてです。</p>
<img src="アジサイ.jpeg">
<p>〇〇についての詳細は，<a href="http://www.example.or.jp">
こちら</a>をご覧ください。</p>
</body>
</html>
```

別紙6－31



別紙6－32

別紙6－33

「分岐構造」のプログラムのデータ（Python）

```
n = 70
if n >= 80:
    print("最高")
elif n >= 60:
    print("よい")
else:
    print("普通")
```

別紙6－34

「分岐構造」のプログラムのデータ（疑似言語）

```
n = 70
もし n >= 80 ならば：
  | 表示する("最高")
elif n >= 60 ならば：
  | 表示する("よい")
else:
  L 表示する("普通")
```

別紙6－35

「反復構造」のプログラムのデータ（Python）

```
for i in range(1, 6, 2):
  print(i)

------

for i in range(3, 6, 1):
  print(i)

------

for i in range(3):
  print(i)

------

for i in range(3):
  print("A")
```

別紙6－36

「反復構造」のプログラムのデータ（疑似言語）

```
iを1から5まで2ずつ増やしながらくりかえす：
  L 表示する (i)

------

iを3から5まで1ずつ増やしながらくりかえす：
  L 表示する (i)

------

iを0から2まで1ずつ増やしながらくりかえす：
  L 表示する(i)

------

iを0から2まで1ずつ増やしながらくりかえす：
  L 表示する("A")
```