# 内容


０１　変数と演算子 ..... 2
０２　print ..... 5
０３　input ..... 9
０４　int ..... 12
０５　str ..... 13
A　ここまでの総合演習 ..... 15
０６　if ..... 24
B　ここまでの総合演習 ..... 27
０７　リスト ..... 33
０８　タプル ..... 35
０９　in ..... 37
１０　for ..... 38
１１　辞書 ..... 44
C　ここまでの総合演習 ..... 46
１２　while ..... 68
１３　split ..... 77
１４　splitlines ..... 79
１５　open（ファイル入出力） ..... 81
１６　import ..... 103
１７　def（ユーザ定義関数） ..... 105
１８　return ..... 107
D　関数総合演習 ..... 109
１９　モジュール ..... 141
２０　+=,-= ..... 143
２１　slice ..... 145
２２　range ..... 147
２３　enumerate ..... 149
２４　変数のスコープ ..... 151

# 01　変数と演算子

## 演習 1-01

以下のプログラムを動作させると、変数a、b、cはそれぞれ最終的にどのような値になるか答えてください。

［Python コード］

```
a = 100
b = 200
c = 500
a = c
b, c = b, b + 200
```

| | |
|---|---|
| a | |
| b | |
| c | |

## 演習 1-02

次の変数の型を答えてください。

```
a = 1
b = 'True'
c = False
d = None
e = ''
f = 1.
```

| | |
|---|---|
| a | |
| b | |
| c | |
| d | |
| e | |
| f | |

## 演習 1-03

次の演算の結果を答えてください。

```
a = 1 + 2
b = 7 + 7 / 7 + 7
c = 7 + 7 * 7 / 7 + 7
d = 7 + (7 + 7 * 7 / 7 + 7)
e = 1 + 2 // 3 - 4 % 5 ** 2
f = 100 == "100"
g = 4 == 4.
```

ちなみに、`1 == 1.0000000000000001` は `True` となります。

| | |
|---|---|
| a | |
| b | |
| c | |
| d | |
| e | |
| f | |
| g | |

# 02 print

## 演習 2-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
a bc
d e
```

［Python コード］

```
print(___ア___, ___イ___)
print(___ウ___, ___エ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 2-02

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
1|2|3,4,5
```

［Python コード］

```
___ア___ (1,2, ___イ___,___ウ___)
___エ___ (3,4,5, ___オ___)
```

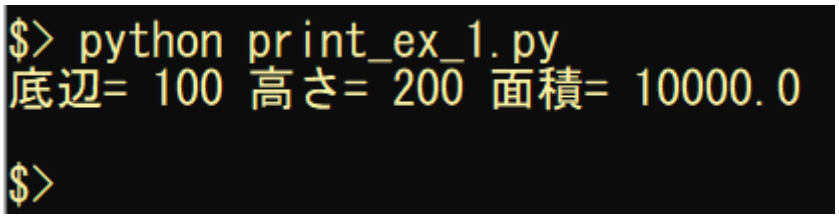
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

## 演習 2-03

下記のプログラムは三角形の面積を求めて結果を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

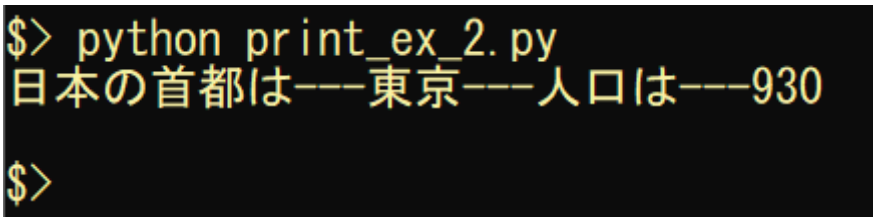
```
# 三角形の面積を求める

# 底辺を代入
base = 100
# 高さを代入
height = 200
# 三角形の面積を計算
area = base * height / 2
# 結果の表示
print(___ア___, ___イ___, ___ウ___, ___エ___, ___オ___, ___カ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |

## 演習 2-04

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 文字列を表示する

# 首都を代入
capital = '東京'
# 人口を代入
population = 930
# 表示する
print(___ア___, ___イ___, ___ウ___, ___エ___, ___オ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

# 03 input

## 演習 3-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果例］ キーボードから"田中"を入力した場合

お名前を入力して下さい ＞ 田中

こんにちは田中さん

［Python コード］

```
#ユーザによる名前の入力
s = ___ア___
#入力された値を利用して、あいさつ
___ア___
```

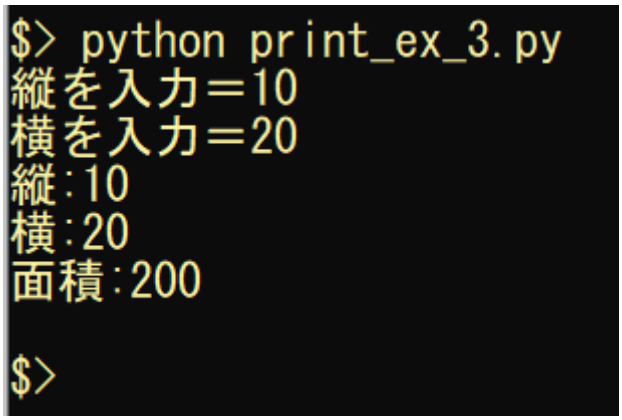
| | |
|---|---|
| ア | |
| イ | |

## 演習 3-02

下記のプログラムは三角形の面積を求めて結果を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 長方形の面積を求める

# 縦を入力
vertical = input('縦を入力＝')
# 横を入力
side = input('横を入力＝')
# 面積を求める
area = int(vertical) * int(side)
# 面積を表示
print(___ア___,___イ___,___ウ___)
print(___エ___,___オ___,___カ___)
print(___キ___,___ク___,___ケ__)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |
| ク | |
| ケ | |

# 04 int

## 演習 4-01

以下の選択肢の内、整数値（int 型）の 10 に変換する選択肢をすべて答えてください。

ア int(10)
イ int("10")
ウ int("十")
エ int("50/5")
オ int(50/5)
カ int(r"  10   ")

# 05 str

## 演習 5-01

以下の記述を**10**進数の**10**に変換する記述にするために、空欄を埋めてください。なお、**int** 関数は**(  )**内の「**,**(カンマ)」に続けて基数を指定することができます。例えば、**int("101", 2)**は2進数 **101** を表し、**10** 進表記にすると **5** となります。

```
1 int("___ア___", 2)
2 int("___イ___", 8)
3 int("___ウ___", 10)
4 int("___エ___", 16)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 5-02

以下の選択肢の内、**str** 型の**"100"**に変換する選択肢をすべて答えてください。

ア **str(100)**

イ **str(100.)**

ウ **str(10*10)**

エ **str(r"100")**

# A ここまでの総合演習

## 演習 A-01

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python mondai1_ans.py
100  +  200  =  300

$>
```

［Python コード］

```
# 足し算の問題

# 変数 number1 に 100 を代入する
___ア___
# 変数 number2 に 200 を代入する
___イ___
# 変数 sum に number1 と number2 の足し算の結果を代入する
___ウ___
# print 関数を使って、足し算の結果をディスプレイに表示する
___エ___
```

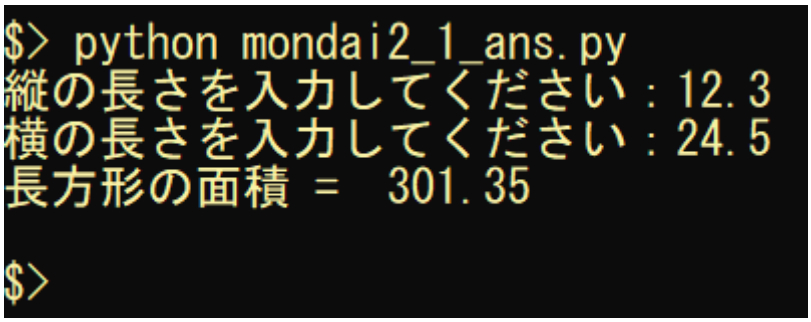
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 A-02

下記のプログラムはキーボードから長方形の縦の長さと横の長さを入力し、長方形の面積を求めて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 長方形の面積を求める

# キーボードから縦の長さを入力する
___ア___
# キーボードから横の長さを入力する
___イ___
# 入力された縦と横の長さを浮動小数点に変換して長方形の面積を求める
___ウ___
# 結果を表示する
print('長方形の面積 = ', area)
```

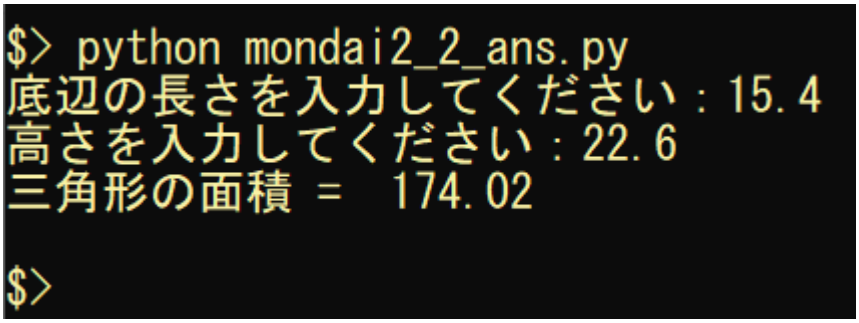
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 A-03

下記のプログラムはキーボードから三角形の底辺の長さと高さを入力し、三角形の面積を求めて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 三角形の面積を求める

# キーボードから底辺の長さを入力する
___ア___
# キーボードから高さを入力する
___イ___
# 入力された底辺の長さと高さを浮動小数点に変換して三角形の面積を求める
___ウ___

# 結果を表示する
print('三角形の面積 = ', area)
```

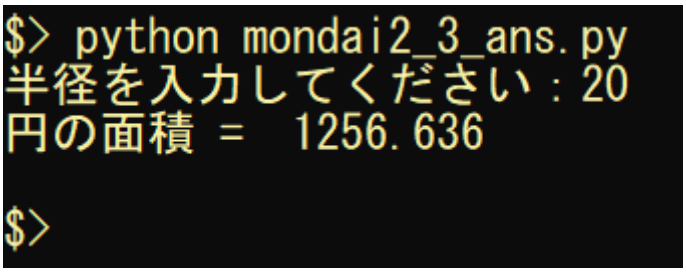
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 A-04

下記のプログラムはキーボードから半径を入力し、円の面積を求めて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 円の面積を求める

# 円周率
PAI = 3.14159
# キーボードから半径を入力する
___ア___
# 入力された半径を浮動小数点に変換して、円の面積を求める
___イ___
# 円の面積を表示する
print('円の面積 = ', area)
```

| | |
|---|---|
| ア | |
| イ | |

## 演習 A-05

下記のプログラムはキーボードから台形の上底の長さ、下底の長さ、高さを入力し、台形の面積を求めて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python mondai2_4_ans.py
上底の長さを入力してください：15.5
下底の長さを入力してください：35.5
高さを入力してください：22.8
台形の面積 =  581.4

$>
```

［Python コード］

```
# 台形の面積を求める

# キーボードから上底の長さを入力する
___ア___
# キーボードから下底の長さを入力する
___イ___
# 入力された上底と下底の長さを浮動小数点に変換して台形の面積を求める
___ウ___
# 結果を表示する
print('台形の面積 = ', area)
```

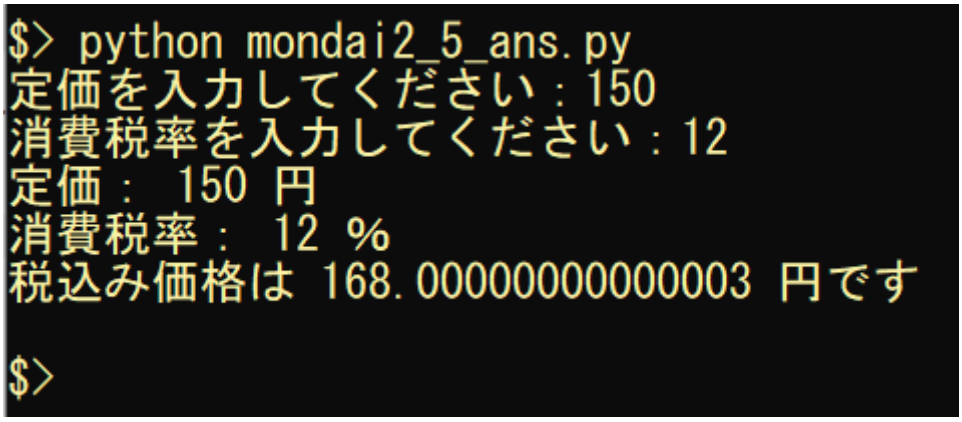
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 A-06

下記のプログラムはキーボードから定価と消費税率を入力し、税込み価格を求めて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

[実行結果]




[Python コード]

```
# 税込み価格の計算

# キーボードから定価を入力し、整数型に変換
list_price = ___ア___
# 消費税率を入力し、整数型に変換
tax_rate = ___イ___
# 税込み価格を計算
tax_price = list_price * (1 + tax_rate / 100)
# 入力した定価の表示
print(___ウ___)
# 入力した税率の表示
print(___エ___)
# 税込み価格の表示
print(___オ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

## 演習 A-07

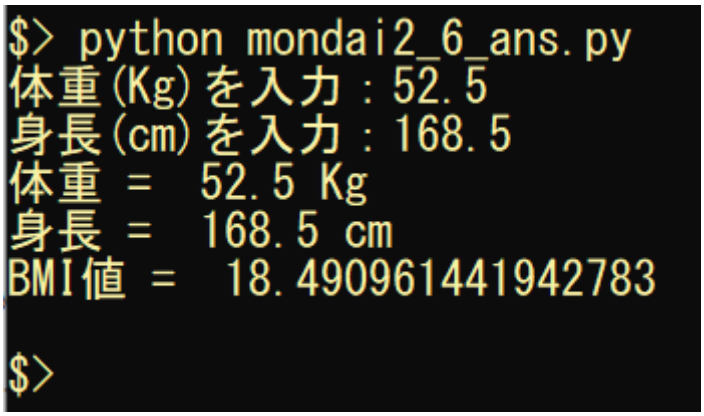
下記のプログラムはキーボードから体重(Kg 単位)と身長(cm 単位)を入力し、BMI 値を求めて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

ただし、BMI 値は以下の計算式によります。

BMI 値 ＝ 体重 kg ÷ (身長 m)$^2$

[実行結果]




[Python コード]

```
# BMI 値を求める

# キーボードから体重(Kg 単位)を入力し浮動小数点に変換
weight = ___ア___
# キーボードから身長(cm 単位)を入力し浮動小数点に変換
height_cm = ___イ___
# 身長をメートルに換算
height = ___ウ___
# BMI 値を計算
bmi = ___エ___
# 入力した体重を表示
print(___オ___)
# 入力した身長を表示
```

```
print(___カ___)
# BMI 値を表示
print(___キ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |

# 06 if

## 演習 6-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

3 は奇数です。

［Python コード］

```
___ア___

___イ___:
    print(num, "は偶数です。")
___ウ___:
    print(num, "は奇数です。")
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 6-02

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。なお、if 文に複数の条件を指定するためには、or や and で条件を繋いで記述します。例えば、「if a or b」とした場合、条件 a と b のどちらかが真となれば条件式として真の結果となります。

［実行結果］

補欠合格

［Python コード］

```
score1 = 11
score2 = ___ア___
sum = score1 + score2

# 合計が 160 以上または score1 か score2 が 100
if sum >= 160 or score1 == 100 or score2 == 100:
    print('特待生')
# 合計が 120 以上かつ score1 と score2 が 40 以上
elif ___イ___:
    print('合格')
# 合計が 110 以上
elif ___ウ___:
    print('補欠合格')
# 条件外
else:
    print('不合格')
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

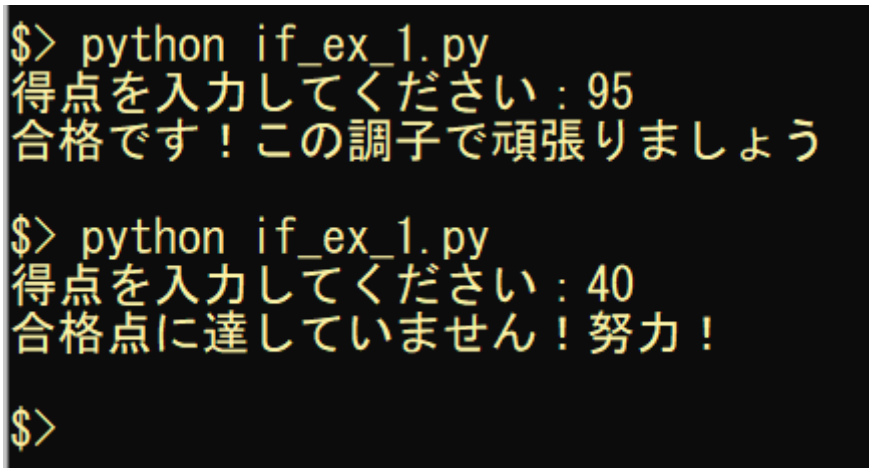
# B ここまでの総合演習

## 演習 B-01

下記のプログラムはキーボードから得点を入力し、70点以上であれば合格、70点未満であれば合格に達していないことを判断して表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 成績を評価する

# 得点を入力
score = int(___ア___)
# 成績を評価
if  ___イ___:
  print('合格です！この調子で頑張りましょう')
else:
  print('合格点に達していません！努力！')
```

| | |
|---|---|
| ア | |
| イ | |

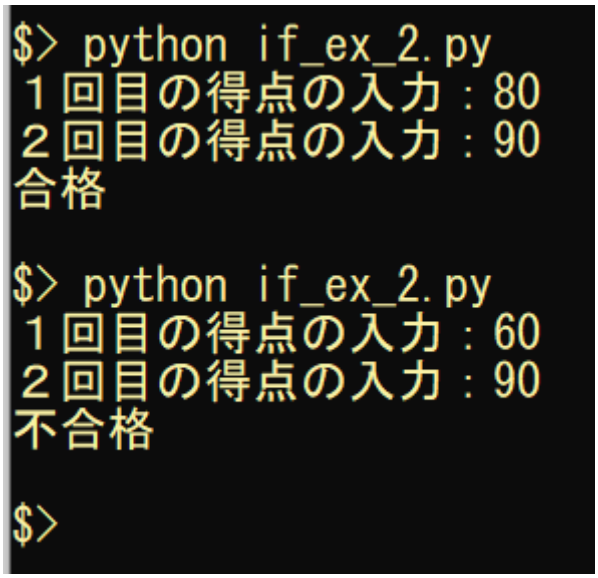
## 演習 B-02

下記のプログラムは 2 回実施した試験の結果に基づいて、合否を判定します。

合否は 2 回の試験がともに 70 点以上であれば“合格”と表示し、そうでなければ“不合格”と表示します。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。(プログラム1とプログラム2は同じ結果を表示します。)

[実行結果]




[Python コード]

プログラム1

```
# 成績を評価する

# 得点を入力(1回目)
score1 = int(input('1回目の得点の入力:'))
# 得点を入力(2回目)
score2 = int(input('2回目の得点の入力:'))

# 成績を評価
if score1 < 70:
```

```
    print(___ア___)
  else:
    if score2 >= 70:
      print(___イ___)
    else:
      print(___ウ___)
```

プログラム2

```
# 成績を評価する

# 得点を入力(1回目)
score1 = int(input('1回目の得点の入力:'))
# 得点を入力(2回目)
score2 = int(input('2回目の得点の入力:'))

# 成績を評価
if ___エ___:
  print('合格')
else:
  print('不合格')
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 B-03

下記のプログラムは西暦をキーボードから入力し、閏年か閏年でないかを判定し、表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

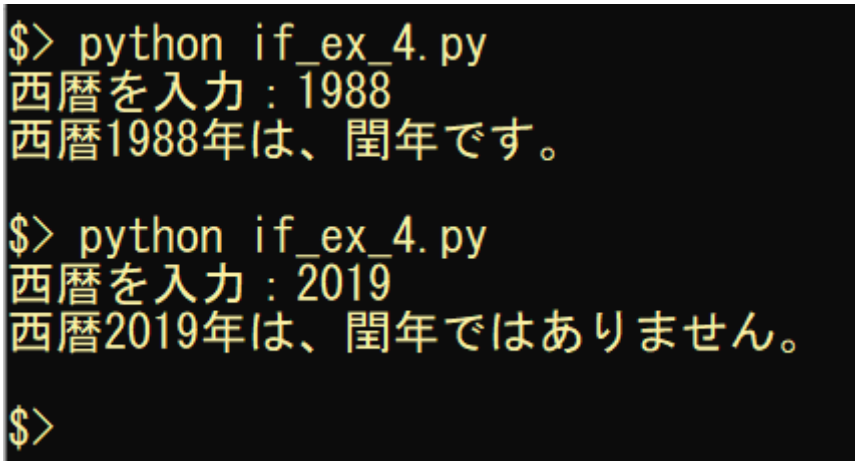
ただし閏年の判定は以下のように行います。

・西暦年が 4 で割り切れる年は(原則として)閏年

・ただし、西暦年が 100 で割り切れる年は(原則として)平年

・ただし、西暦年が 400 で割り切れる年は必ず閏年

［実行結果］




［Python コード］

```
# 閏年の判断

# 西暦を入力
year = int(input('西暦を入力:'))
# 閏年を判断
if ___ア___:
  print('西暦{}年は、閏年です。'.format(year))
else:
  print('西暦{}年は、閏年ではありません。'.format(year))
```

| | |
|---|---|
| ア | |

# 07 リスト

## 演習 7-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
['orange', 'apple', 'banana', 'apple']
2
['orange', 'apple', 'banana', 'apple', 'banana']
1
['banana', 'apple', 'banana', 'apple', 'orange']
```

［Python コード］

```
fruits = ___ア___
print(fruits)
print(fruits.count(___イ___))
fruits.append(___ウ___)
print(fruits)
print(fruits.index(___エ___))
fruits.___オ___ ()
print(fruits)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

## 演習 7-02

ア,イには以下のプログラムを実行すると、どのような実行結果が出力されるか空欄を埋めてください。またウ,エには以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
[1, 2, 3, 4, 5, 6, 7, 8]
3
___ア___
___イ___
```

［Python コード］

```
num_lists = ___ウ___
print(num_lists)
print(num_lists[___エ___])
print(num_lists[-3])
print(num_lists[-0])
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

# 08 タプル

## 演習 8-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
1
2
3
```

［Python コード］

```
a = ___ア___
b, c = a
d, e = c
print(b)
print(d)
print(e)
```

| | |
|---|---|
| ア | |

## 演習 8-02

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。なお、**print** 関数の**()**内の文字列中の**"%s"**は、後述する**%**の後ろに書く値に置き換わります。例えば、「**print("**吾輩は**%s** である**" % "**猫**")**」を実行すると、画面には**"**吾輩は猫である**"**と表示されます。

［実行結果］

**empty** の長さは **0**

**singleton** の長さは **1**

**singleton** の中身は **Hello**

［**Python** コード］

```
empty = ___ア___
singleton = ___イ___
print("empty の長さは%s" % len(empty))
print("singleton の長さは%s" % len(singleton))
print("singleton の中身は%s" % singleton[0])
```

| | |
|---|---|
| ア | |
| イ | |

# 09 in

## 演習 9-01

以下の記述の内、**True** となる記述はどれかすべて選んでください。

ア `2 in [1, 2, 3]`

イ `"2" in [1, 2, 3]`

ウ `2 in "123"`

エ `2 in ((1, 2), 3)`

オ `2 in (((1, 2, 3)))`

カ `2 in [((1, 2, 3))]`

# 10 for

## 演習 10-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
2
4
6
14
42
```

［Python コード］

```
for i in [1, 2, 3, ___ア___]:
    print(___イ___)
```

| | |
|---|---|
| ア | |
| イ | |

## 演習 10-02

以下のように、**11** が素数かどうかを判定するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
11 は素数です。
```

［**Python** コード］

```
x = 11
# 11 に、1 と 11 以外の約数があるかを調べる
for n in___ア___:
    # n で 11 を割り切れるかを調べる
    if ___イ___:
        print(x, 'は', n, 'で割り切れるので、素数ではありません。')
        # n で割り切れたので、ループを終える
            ___ウ___
#ループを全て回せたので、素数と判定する
___エ___:
    print(x, 'は素数です。')
```

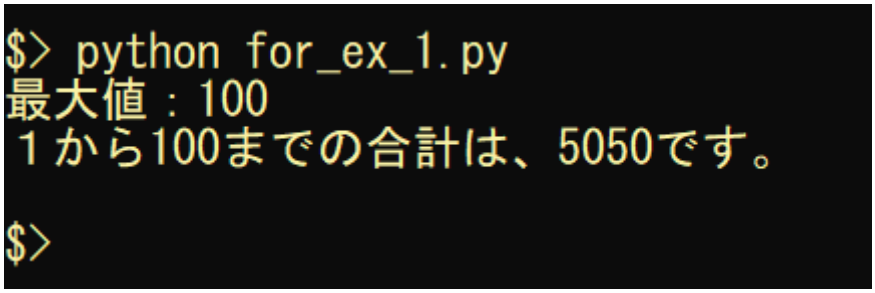
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 10-03

下記のプログラムは1から、キーボードから入力された正の整数値までの合計を求めて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# キーボードから正の数値 n を入力し
# 1～n までの総和を求める。

# 最大値を入力
count = int(input('最大値:'))

# 1から最大値の合計を求める
total = 0 # 合計値を初期化
for value in ___ア___:
  ___イ___

# 合計結果を表示
# テンプレートを作成
tpl = '1から{}までの合計は、{}です。'
print(___ウ___)
```

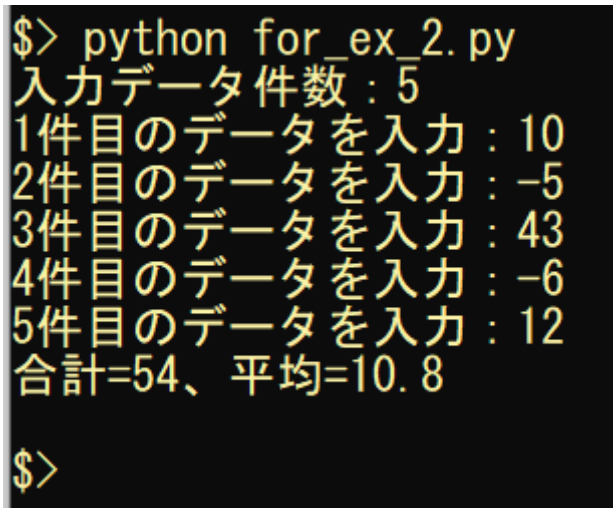
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 10-04

下記のプログラムはキーボードから入力した入力件数分の整数値を入力し、合計と平均を求めてそれぞれ表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 指定した数のデータを入力し
# 入力したデータの合計、平均を求めて画面に表示する。

# データ入力件数を入力
number = int(input('入力データ件数:'))

# データ入力件数分の数値の合計を求める
total = 0 # 合計値の初期化
average = 0 # 平均値の初期化
for value in ___ア___:
  # 入力促進用の文字列を作成
  prompt = ___イ___
  # データを入力
```

```
    data = ___ウ___
    # 合計を計算
    ___エ___
# 平均値を計算
average = ___オ___
# 合計と平均を表示
print('合計={}、平均={}'.___カ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |

# 11　辞書

**演習 11-01**

以下の選択肢の内、辞書型の記述として正しいものを全て選んでください。

ア `{"apple":"りんご", "banana":"バナナ"}`

イ `{"","",}`

ウ `{1:3, 4:3}`

エ `{(1,2):3, 3:2}`

オ `{}`

カ `{[1]:1, [2]:2}`

## 演習 11-02

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
{'国語': 75, '算数': 80}
{'国語': 75, '数学': 80}
{'国語': 75, '数学': 80, '理科': 65, '社会': 90, '英語': 70}
```

［Python コード］

```
result_of_exam = ___ア___
print(result_of_exam)
___イ___(result_of_exam["算数"])
___ウ___
print(result_of_exam)
result_of_exam.___エ___ ({"理科":65, "社会":90, "英語":70})
print(result_of_exam)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

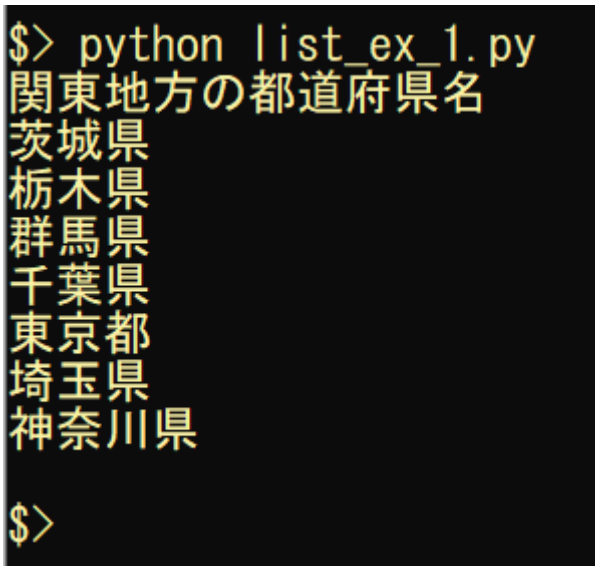
# C ここまでの総合演習

## 演習 C-01

下記のプログラムはリストに保存された要素をすべて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

[実行結果]




[Python コード]

```
# 文字列リストの要素を表示する

# 文字列リストの初期化
kanto = [
  '茨城県',
  '栃木県',
  '群馬県',
  '千葉県',
  '東京都',
  '埼玉県',
```

```
  '神奈川県'
]

# 文字列リストを表示
print('関東地方の都道府県名')
for ___ア___:
  print(ken_name)
```

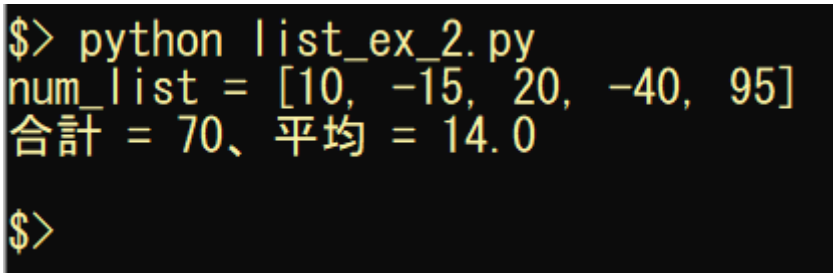
| | |
|---|---|
| ア | |

## 演習 C-02

下記のプログラムはリストに保存された要素の合計値と平均値を求めて、それぞれ表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 整数リストの合計値と平均値を求める

# リストの初期化
num_list = [10, -15, 20, -40, 95]
# 合計と平均の初期化
total = 0
average = 0
# 合計を計算
for value in ___ア___:
  ___イ___
# 平均を計算
average = ___ウ___
# リスト全体を表示
print('num_list =', num_list)
# 合計と平均を表示
print('合計 = {}、平均 = {}'.___エ___)
```

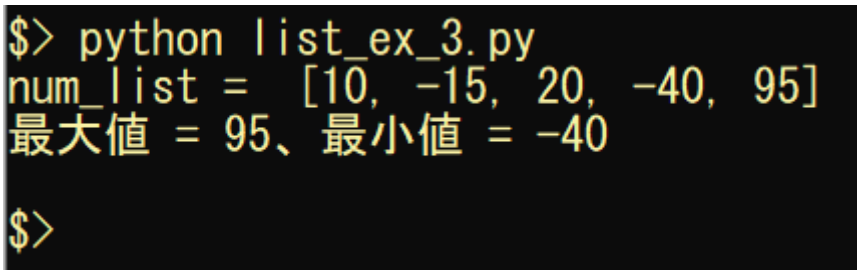
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 C-03

下記のプログラムはリストに保存された要素の最大値と最小値を求めて、それぞれ表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 整数リストの最大値と最小値を求める

# リストの初期化
num_list = [10, -15, 20, -40, 95]

max = num_list[0] # 最大値の初期化
min = num_list[0] # 最小値の初期化
# 最大値と最小値を求める
for ___ア___:
  # 最大値を求める
  if ___イ___:
    max = value
  # 最小値を求める
  if ___ウ___:
    min = value
# リスト全体を表示
print('num_list = ', num_list)
# 最大値と最小値を表示
```

```
print('最大値 = {}、最小値 = {}'.___エ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 C-04

下記のプログラムはキーボードから10件の整数値を入力し、偶数値の要素のリストと奇数値の要素のリストを生成して、それぞれ表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python list_ex_4.py
1件目：整数を入力 = 10
2件目：整数を入力 = 20
3件目：整数を入力 = -3
4件目：整数を入力 = 15
5件目：整数を入力 = 22
6件目：整数を入力 = -8
7件目：整数を入力 = 33
8件目：整数を入力 = 58
9件目：整数を入力 = -9
10件目：整数を入力 = 18
偶数値リスト =  [10, 20, 22, -8, 58, 18]
奇数値リスト =  [-3, 15, 33, -9]

$>
```

［Python コード］

```
# キーボードから10件の整数を入力し
# 偶数値のリストと奇数値のリストを生成して表示する

# 偶数値のリストと奇数値のリストを初期化
even_list = []
odd_list = []
# 10件の整数を入力
for number in ___ア___:
```

```
  # 入力促進文字列の生成
  prompt = '{}件目:整数を入力 = '.___イ___
  # 入力
  data = ___ウ___
  # 偶数か否かを判断
  if ___エ___:
    # 偶数値のリストに偶数を追加
    ___オ___
  else:
    # 奇数値のリストに奇数を追加
   ___カ___
# 偶数値リストと奇数値リストを表示
print('偶数値リスト = ', even_list)
print('奇数値リスト = ', odd_list)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |

## 演習 C-05

下記のプログラムはリストの要素から最大値を求め、その最大値の要素を削除し、削除後のリストを表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python list_ex_5.py
最初のリスト =  [10, 20, 33, 55, 60, 79, 14]
最大値削除後のリスト =  [10, 20, 33, 55, 60, 14]

$>
```

［Python コード］

```
# 整数値のリストから最大値を削除

# 整数値リストを初期化
num_list = [10, 20, 33, 55, 60, 79, 14]
# 初期化後のリストの表示
print('最初のリスト = ', num_list)
# 最大値の初期化
max = num_list[0]
# 整数値リストの要素数分繰り返す
for ___ア___:
  # 最大値を求める
  if ___イ___:
    max = value
# リストから最大値を削除
___ウ___
# 最大値を削除したリストの表示
print('最大値削除後のリスト = ', num_list)
```

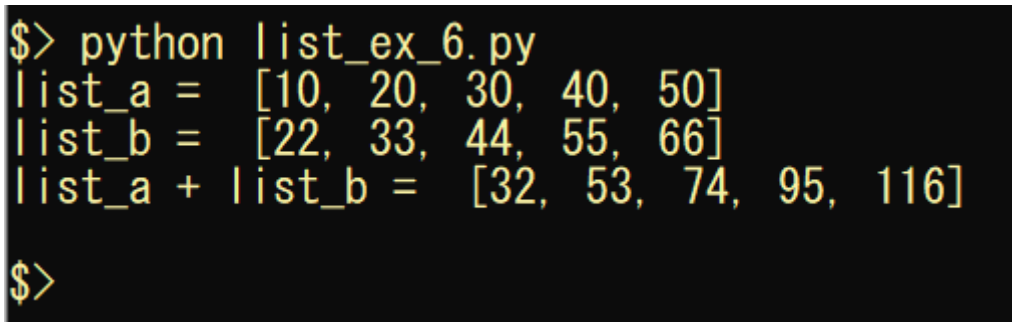
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 C-06

下記のプログラムは2つのリストの要素同士の足し算をして結果を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 整数リストの要素同士の足し算

# 整数リストの初期化
list_a = [10, 20, 30, 40, 50]
list_b = [22, 33, 44, 55, 66]
list_add = []
# リストの要素数分繰り返す
for index in ___ア___:
  ___イ___
# リストの表示
print('list_a = ', list_a)
print('list_b = ', list_b)
# リストの要素同士の足し算
print('list_a + list_b = ', list_add)
```

| | |
|---|---|
| ア | |
| イ | |

## 演習 C-07

下記のプログラムは整数を要素とするリストにおいて要素を降順にソートし、結果を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python list_ex_7.py
ソート前 =  [10, -10, 40, -15, 90, 15, -80]
降順ソート後 =  [90, 40, 15, 10, -10, -15, -80]

$>
```

［Python コード］

```
# 整数値のリストを降順にソートする

# 整数値のリストを初期化
num_list = [10, -10, 40, -15, 90, 15, -80]
# ソート前のリストを表示
print('ソート前 = ', num_list)
for index1 in ___ア___:
  for index2 in ___イ___:
    # 後のリストの要素との比較
    if ___ウ___:
      # 後のリスト要素が大きければ前のリスト要素と入れ替え
      temp = num_list[index1]
      ___エ___
      num_list[index2] = temp
# 降順ソート後のリストを表示
print('降順ソート後 = ', num_list)
```

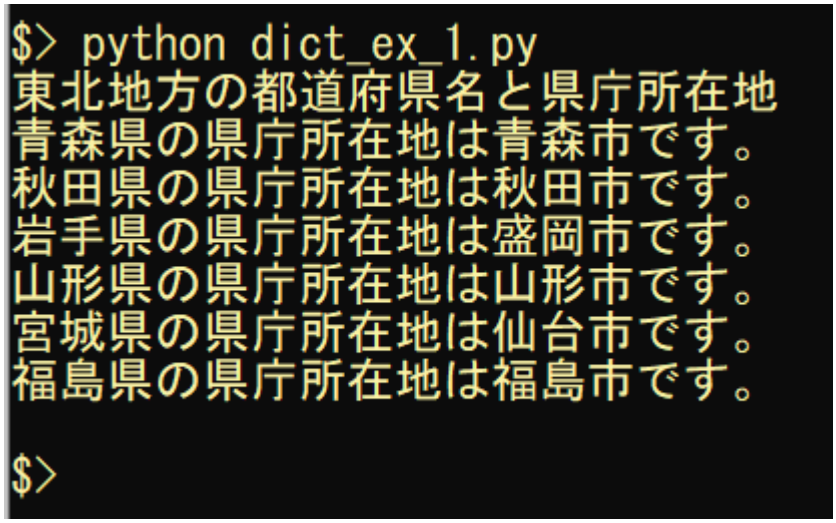
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 C-08

下記のプログラムは都道府県名をキーとし、県庁所在地名をバリューとする辞書のキーとバリューを表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 辞書

# 辞書の初期化
tohoku = {
  '青森県' : '青森市',
  '秋田県' : '秋田市',
  '岩手県' : '盛岡市',
  '山形県' : '山形市',
  '宮城県' : '仙台市',
  '福島県' : '福島市'
}
# 都道府県名と県庁所在地を表示
print('東北地方の都道府県名と県庁所在地')
```

```
for ___ア___:
  print('{}の県庁所在地は{}です。'.format(key, ___イ___))
```

| | |
|---|---|
| ア | |
| イ | |

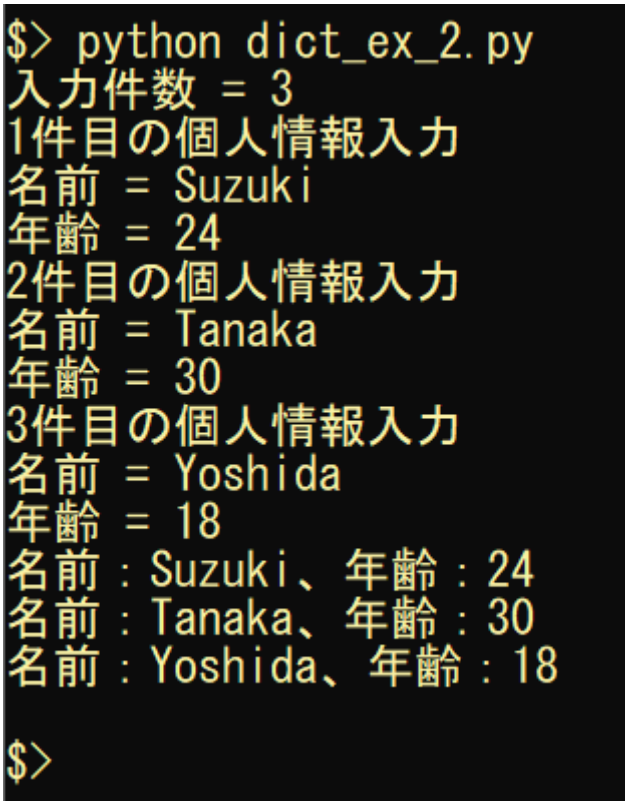
## 演習 C-09

下記のプログラムはキーボードから入力件数を入力し、その件数分の個人情報（名前と年齢）を入力して、辞書に保存します。

その後、その辞書に保存された個人情報を表示します。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 辞書
# 個人情報の入力と表示

# 個人情報辞書の初期化
person = {}
# 入力件数を入力
```

```
number = int(input('入力件数 = '))
# 入力件数分繰り返す
for ___ア___:
  print('{}件目の個人情報入力'.format(count))
  # 名前の入力
  name = input('名前 = ')
  # 年齢の入力
  age = int(input('年齢 = '))
  # 個人情報辞書に名前と年齢を追加
  ___イ___
# 入力した個人情報を表示
for key in person:
  print('名前:{}、年齢:{}'.format(___ウ___))
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 C-１０

下記のプログラムはキーボードから入力件数を入力し、その件数分の個人情報（名前と年齢）を入力して、辞書に保存します。

その後、その辞書に保存された個人情報を表示し、平均年齢の表示も行います。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python dict_ex_3.py
入力件数 = 5
1件目の個人情報入力
名前 = Suzuki
年齢 = 24
2件目の個人情報入力
名前 = Tanaka
年齢 = 18
3件目の個人情報入力
名前 = Yoshida
年齢 = 20
4件目の個人情報入力
名前 = Sato
年齢 = 38
5件目の個人情報入力
名前 = Kato
年齢 = 16
名前：Suzuki、年齢：24
名前：Tanaka、年齢：18
名前：Yoshida、年齢：20
名前：Sato、年齢：38
名前：Kato、年齢：16
平均年齢は23.2歳です。

$>
```

［Python コード］

```
# 辞書
# 個人情報の入力と表示および平均年齢の計算と表示

# 個人情報辞書の初期化
person = {}
# 入力件数を入力
number = int(input('入力件数 = '))
# 入力件数分繰り返す
for ___ア___:
  print('{}件目の個人情報入力'.format(count))
  # 名前の入力
  name = input('名前 = ')
  # 年齢の入力
  age = int(input('年齢 = '))
  # 個人情報辞書に名前と年齢を追加
  ___イ___

# 合計と平均を初期化
total = 0
average = 0
# 入力した個人情報と平均年齢を表示
for ___ウ___:
  # 合計を計算
  ___エ___
  print('名前:{}、年齢:{}'.format(key, ___オ___))
# 平均年齢の計算
average = ___カ___
# 平均年齢を表示
print('平均年齢は{}歳です。'.format(average))
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |

## 演習 C-１１

下記のプログラムは辞書に保存された各都市の北緯と東経を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python dict_ex_4.py
主要都市情報
東京都の位置 :  (北緯35.6896342, 東経139.6899121)
横浜市の位置 :  (北緯35.4440992, 東経139.6358831)
札幌市の位置 :  (北緯43.0620803, 東経141.3521727)
大阪市の位置 :  (北緯34.6937381, 東経135.4999759)
神戸市の位置 :  (北緯34.6894859, 東経135.1935503)
京都市の位置 :  (北緯35.0116574, 東経135.7659363)
金沢市の位置 :  (北緯36.5610485, 東経136.6543849)

$>
```

［Python コード］

```
# 辞書
# 主要都市情報
'''
都市名(文字列)    北緯と東経(タプル)
東京都            (35.6896342,139.6899121)
横浜市            (35.4440992,139.6358831)
札幌市            (43.0620803,141.3521727)
大阪市            (34.6937381,135.4999759)
神戸市            (34.6894859,135.1935503)
京都市            (35.0116574,135.7659363)
金沢市            (36.5610485,136.6543849)
'''

# 都市情報辞書の初期化
```

```
city = {
  '東京都' : (35.6896342, 139.6899121),
  '横浜市' : (35.4440992, 139.6358831),
  '札幌市' : (43.0620803, 141.3521727),
  '大阪市' : (34.6937381, 135.4999759),
  '神戸市' : (34.6894859, 135.1935503),
  '京都市' : (35.0116574, 135.7659363),
  '金沢市' : (36.5610485, 136.6543849)
}

# 都市情報を表示
print('主要都市情報')
for key in city:
  print('{}の位置:(北緯{}, 東経{})'¥
    .format(___ア___, ___イ___, ___ウ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

# 12 while

## 演習 12-01

以下のプログラムは **437** の約数の内、1以外で最小の値を出力するプログラムです。

実行結果のような出力をするために、プログラム中の空欄を埋め、プログラムを完成せてください。なお、**print** 関数の**()**内の文字列中の**"%s"**は、後述する**%**の後ろに書く値に置き換わります。例えば、「**print("**吾輩は**%s** である**" % "**猫**")**」を実行すると、画面には**"**吾輩は猫である**"**と表示されます。

［実行結果］

**19** は **437** の約数です。

［**Python** コード］

```
number = 437
divisor = 2
# number を divisor で割り切れるまで、ループを繰り返す。
while ___ア___:
    # 割り切れなかった場合は、divisor を 1 増やす
    ___イ___

print("%s は%s の約数です。" % (___ウ___, ___エ___))
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 １２-02

以下のプログラムは累乗で **1,000,000** 未満の最大の数を求めるプログラムです。

実行結果のような出力をするために、プログラム中の空欄を埋め、プログラムを完成せてください。なお、プログラム中の「＼（バックスラッシュ）」は1行が長くなる際に用いられる記号で、次行に継続していることを示すことを表します。

［実行結果］

**2**の累乗で**1,000,000**未満の最大の数は**2**の**19**乗である。

［**Python** コード］

```
number = 2
count = 1
# numberが1,000,000以上になるまで、繰り返す
while ___ア___:
    number ___イ___
    count ___ウ___

print("2の累乗で1,000,000未満の最大の数は2の{}乗である。" ＼
    .format(count-1))
```

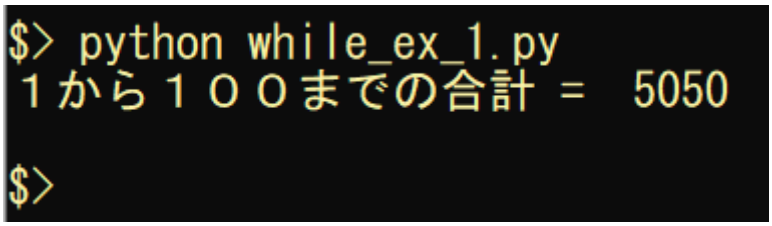
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 12-03

下記のプログラムは1から100までの整数値を合計して表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 1から100までの合計を計算

# 合計値を初期化
total = 0
# 1から100まで繰り返す
number = 1
while ___ア___:
  # 合計を求める
  ___イ___
  # インクリメント
  ___ウ___
# 合計値を表示
print('1から100までの合計 = ', total)
```

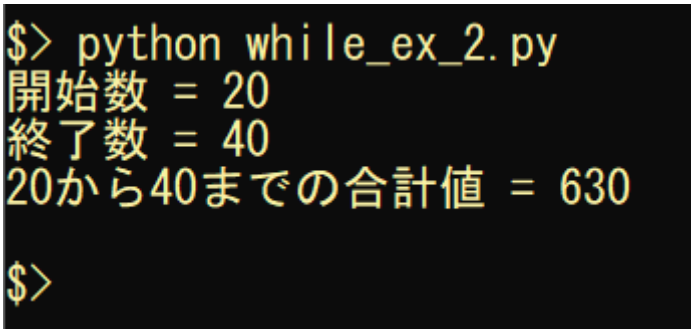
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 12-04

下記のプログラムはキーボードから開始数と終了数を入力し、開始数から終了数までの整数値を合計し表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 開始数と終了数間の整数の合計値を求める

# 開始数を入力
start_num = int(input('開始数 = '))
# 終了数を入力
end_num = int(input('終了数 = '))
# 合計値を初期化
total = 0
# カウンタを初期化
count = start_num
# 開始数から終了数まで繰り返す
while ___ア___:
  # 合計値を求める
  ___イ___
  # カウンタをインクリメント
  ___ウ___
# テンプレート文字列の作成
```

```
tpl = '{}から{}までの合計値 = {}'¥
    .___エ___
# 合計を表示
print(tpl)
```

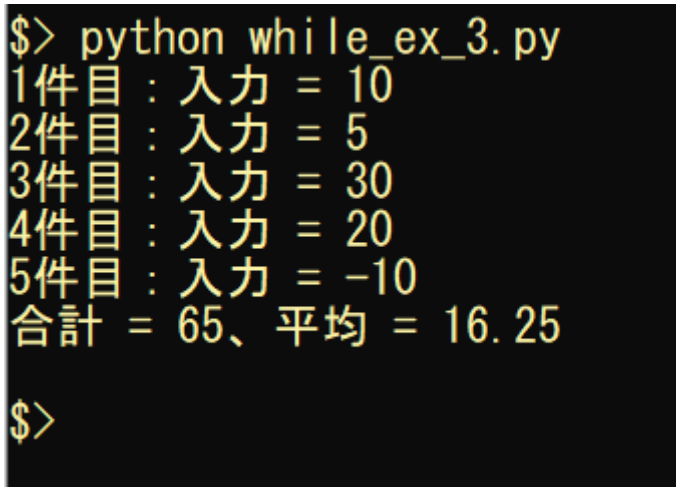
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 12-05

下記のプログラムはキーボードから負の整数が入力されるまで整数を入力し、その合計値と平均値を求めて表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 負の整数が入力されるまで
# 整数値を入力し、合計と平均を表示

# 合計と平均の初期化
total = 0
average = 0
# 無限ループ
# カウンターの初期化
number = 0
while ___ア___:
  # 入力促進文字列の生成
  prompt = '{}件目:入力 = '.format(number+1)
  # 整数値を入力
```

```
    input_data = int(input(prompt))
    # 負の整数値かどうかを判定
    if ___イ___:
        # 無限ループから抜ける
        ___ウ___
    # 合計を計算
    ___エ___
    # カウンタをインクリメント
    ___オ___
# 平均値を求める
average = ___カ___
# 合計と平均を表示
print('合計 = {}、平均 = {}'.format(total, average))
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |

## 演習 12-06

下記のプログラムは整数値を要素とするリストから最大値を削除し、その削除後のリストを表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python while_ex_4.py
最初のリスト =  [10, 20, 33, 55, 60, 79, 79, 14]
最大値削除後のリスト =  [10, 20, 33, 55, 60, 79, 14]

$>
```

［Python コード］

```
# 整数値のリストから最大値を削除

# 整数値リストを初期化
num_list = [10, 20, 33, 55, 60, 79, 79, 14]
# 初期化後のリストの表示
print('最初のリスト = ', num_list)
# 最大値の初期化
max = num_list[0]
# リストのインデックスの初期化
index = 1
# 整数値リストの要素数分繰り返す
while ___ア___:
  # 最大値を求める
  if ___イ___:
    max = num_list[index]
  # インデックスのインクリメント
  ___ウ___
# リストから最大値を削除
___エ___
```

```
# 最大値を削除したリストの表示
print('最大値削除後のリスト = ', num_list)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

# 13 split

**演習 13-01**

以下の選択肢の記述によって生成されるリストをそれぞれ答えてください。

ア `'1,2,3'.split(',')`

イ `'1,2,3'.split(',', maxsplit=1)`

ウ `'1,2,,3,'.split(',')`

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 13-02

`['a', 'b', 'c']`というリストを生成する記述として、正しいものを以下の選択肢から選んでください。

ア `'a,b,c'.split()`
イ `'a b c'.split()`
ウ `'a.b.c'.split()`

# 14　splitlines

## 演習 14-01

以下の選択肢の中から、**splitlines** 関数で行境界と判断される文字を全て選んでください。

ア **¥a**
イ **¥n**
ウ **¥s**
エ **¥t**
オ **¥r¥n**
カ **¥g¥f**
キ **¥x0a**
ク **¥x0b**

## 演習 14-02

以下の選択肢の記述によって生成されるリストをそれぞれ答えてください。

ア `"a¥nb¥rc¥x0bd".splitlines()`

イ `"a¥nb¥rc¥x0bd".splitlines(True)`

| | |
|---|---|
| ア | |
| イ | |

# 15　open（ファイル入出力）

## 演習 15-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
1
2
3
4
5
```

［Python コード］

```
path = 'number.txt'
___ア___ ___イ___(path) as ___ウ___:
    s = f.___エ___
    print(s)
```

[number.txt の中身]

```
1
2
3
4
5
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 15-02

空欄を埋め、以下のように **sample.txt** の中身に書き込みをするプログラムを完成させてください。

［プログラム実行後の **sample.txt** の中身］

```
Hello
```

［**Python** コード］

```
path = 'sample.txt'

___ア___ ___イ___(path, ___ウ___) as ___エ___:
        f.___オ___
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

## 演習 15-03

下記のプログラムはキーボードから「end」が入力されるまでデータを入力し、ファイル data.txt に書き込むものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python file_ex_1.py
入力：Hello
入力：world.
入力：Python is programing language.
入力：end
書き込み終了

$> type data.txt
Hello
world.
Python is programing language.

$>
```

［Python コード］

```
# ファイルに書き込む

# ファイル data.txt を書き込みモードでオープン
write_file = open(___ア___, ___イ___)

# キーボードから文字列「end」が入力されるまで繰り返す
while ___ウ___:
  str_data = input('入力:')
  # end が入力されたら無限ループより抜ける
  if ___エ___:
    break
```

```
  write_file.___オ___(str_data + '¥n')
write_file.___カ___
print('書き込み終了')
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |

## 演習 15-04

下記のプログラムはキーボードから「end」が入力されるまでデータを入力し、ファイル data.txt に行番号付きで書き込むものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］


［Python コード］

```
# ファイルに書き込む

# ファイル data.txt を書き込みモードでオープン
write_file = open(___ア___, ___イ___)

# 行番号を初期化
line_number = 1
# キーボードから文字列「end」が入力されるまで繰り返す
while ___ウ___:
```

```
  str_data = input('入力:')
  # end が入力されたら無限ループより抜ける
  if ___エ___:
    ___オ___
  # 行番号付きで書き込む
  write_file.___カ___(___キ___+ ' : ' + str_data + '¥n')
  # 行番号をインクリメント
  ___ク___
# ファイルをクローズ
write_file.___ケ___
print('書き込み終了')
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |
| ク | |
| ケ | |

## 演習 15-05

下記のプログラムはキーボードから「**end**」が入力されるまでデータを入力し、またキーボードから入力されたファイル名のファイルに追加モードで行番号付きで書き込むものです。

下記の **Python** コードのコメント部分に従って **Python** コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python file_ex_3.py
ファイル名 : append.txt
入力 : aaa
入力 : bbbb
入力 : cc
入力 : end
書き込み終了

$> python file_ex_3.py
ファイル名 : append.txt
入力 : ddddd
入力 : eee
入力 : end
書き込み終了

$> type append.txt
1 : aaa
2 : bbbb
3 : cc
1 : ddddd
2 : eee

$>
```

［Python コード］

```
# ファイル名を入力し、そのファイルへの追加書き込み

# 書き込むファイル名を入力
file_name = input('ファイル名:')
# ファイルを追加モードでオープン
write_file = open(___ア___, ___イ___)
# 行番号を初期化
line_number = 1
# 文字列「end」が入力されるまで繰り返す
while ___ウ___:
  # キーボードから文字列を入力
  input_str = input('入力:')
  # 「end」が入力されたかを判断
  if ___エ___:
    # 無限ループより抜ける
    ___オ___
  # オープンしたファイルに行番号付きで書き込む
  write_file.___カ___(___キ___+ ' : ' + input_str + '¥n')
  # 行番号をインクリメント
  ___ク___
# ファイルをクローズ
write_file.___ケ___
print('書き込み終了')
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |

| | |
|---|---|
| キ | |
| ク | |
| ケ | |

## 演習 15-06

下記のプログラムはキーボードから入力する個人情報の件数と、データを保存するファイル名を入力し、個人情報を入力後、入力されたファイル名のファイルに追加モードで書き込み、保存するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python file_ex_4.py
ファイル名 : person.txt
入力件数 = 3
1件目の個人情報入力
名前 = Tanaka
年齢 = 20
2件目の個人情報入力
名前 = Sato
年齢 = 18
3件目の個人情報入力
名前 = Suzuki
年齢 = 32

$> type person.txt
名前 : Takeda、年齢 : 38
名前 : Sato、年齢 : 20
名前 : Tanaka、年齢 : 40
名前 : Tanaka、年齢 : 20
名前 : Sato、年齢 : 18
名前 : Suzuki、年齢 : 32

$>
```

［Python コード］

```
# 辞書
# 個人情報の入力とファイルへの出力

# 出力するファイルのファイル名を入力
file_name = input('ファイル名:')
# 指定したファイルを追加モードでオープン
write_file = open(___ア___, ___イ___)

# 個人情報辞書の初期化
person = {}
# 入力件数を入力
number = int(input('入力件数 = '))
# 入力件数分繰り返す
for ___ウ___:
  print('{}件目の個人情報入力'.format(count))
  # 名前の入力
  name = input('名前 = ')
  # 年齢の入力
  age = int(input('年齢 = '))
  # 個人情報辞書に名前と年齢を追加
  ___エ___

# 入力した個人情報を指定されたファイルに書き出す
for ___オ___:
  # 辞書の個人情報を整形
  person_data = '名前:{}、年齢:{}'.format(key, ___カ___)
  # ファイルに書き出す
  write_file.___キ___ (person_data + '¥n')

# ファイルをクローズ
write_file.___ク___
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |
| ク | |

## 演習 15-07

下記のプログラムは **person.txt** を読み込みモードでオープンし、保存されているデータを表示するプログラムです。

下記の **Python** コードのコメント部分に従って **Python** コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python file_ex_5.py
名前：Takeda、年齢：38
名前：Sato、年齢：20
名前：Tanaka、年齢：40
名前：Tanaka、年齢：20
名前：Sato、年齢：18
名前：Suzuki、年齢：32


$>
```

［**person.txt** の内容］

```
$> type person.txt
名前：Takeda、年齢：38
名前：Sato、年齢：20
名前：Tanaka、年齢：40
名前：Tanaka、年齢：20
名前：Sato、年齢：18
名前：Suzuki、年齢：32

$>
```

［**Python** コード］

```
# ファイルからのデータの読み込み
```

```
# ファイル person.txt を読み込みモードでオープン
read_file = open(___ア___, ___イ___)
# ファイルからデータを読み込む
raw_data = read_file.___ウ___
# ファイルをクローズする
read_file.___エ___
# 読み込んだデータを表示する
print(raw_data)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 15-08

下記のプログラムは **person.txt** を読み込みモードでオープンし、保存されているデータを「、」で区切って読み込んで表示するものです。

下記の **Python** コードのコメント部分に従って **Python** コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> type person.txt
名前：Takeda、年齢：38
名前：Sato、年齢：20
名前：Tanaka、年齢：40
名前：Tanaka、年齢：20
名前：Sato、年齢：18
名前：Suzuki、年齢：32

$> python file_ex_6.py
['名前：Takeda', '年齢：38¥n名前：Sato', '年齢：20¥n
名前：Tanaka', '年齢：40¥n名前：Tanaka', '年齢：20¥n
名前：Sato', '年齢：18¥n名前：Suzuki', '年齢：32¥n']


$>
```

［**Python** コード］

```
# ファイルからデータを読み込む
# split メソッドを使用して読み込んだデータを分割

# ファイル parson.txt を読み込みモードでオープン
read_file = open(___ア___, ___イ___)
# ファイルからデータを読み込む
raw_data = read_file.___ウ___
# ファイルをクローズする
read_file.___エ___
```

```
# 読み込んだデータを「、」で区切る
str_data = raw_data.___オ___
# 表示する
print(str_data)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

## 演習 15-09

下記のプログラムは **person.txt** を読み込みモードでオープンし、保存されているデータを改行で区切って読み込んで表示するものです。

下記の **Python** コードのコメント部分に従って **Python** コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> type person.txt
名前：Takeda、年齢：38
名前：Sato、年齢：20
名前：Tanaka、年齢：40
名前：Tanaka、年齢：20
名前：Sato、年齢：18
名前：Suzuki、年齢：32

$> python file_ex_7.py
['名前：Takeda、年齢：38', '名前：Sato、年齢：20', '名前：Tanaka、年齢：40',
'名前：Tanaka、年齢：20', '名前：Sato、年齢：18', '名前：Suzuki、年齢：32']

$>
```

［**Python** コード］

```
# ファイルからデータを読み込む
# splitlines メソッドを使用して読み込んだデータを改行で分割

# ファイル person.txt を読み込みモードでオープン
read_file = open(___ア___, ___イ___)
# ファイルからデータを読み込む
raw_data = read_file.___ウ___
# ファイルをクローズする
read_file.___エ___
# 読み込んだデータを改行で区切る
str_data = raw_data.___オ___
print(str_data)
```

| | |
|---|---|
| ア | |
| イ | |

| | |
|---|---|
| ウ | |
| エ | |
| オ | |

## 演習 15-10

下記のプログラムはキーボードから入力されたファイル名のファイルを読み込みモードでオープンして、読み込んだデータを行番号付きで表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］


［Python コード］

```
# 指定したファイルを読み込み
# 行番号付きで表示する

# 読み込むファイルを入力
file_name = input('ファイル名 = ')
```

```
# 指定したファイルを読み込みモードでオープン
read_file = open(___ア___, ___イ___)
# ファイルからデータを読み込む
raw_data = read_file.___ウ___
# ファイルをクローズ
read_file.___エ___
# 読み込んだデータを改行で区切る
read_line = raw_data.___オ___
# 行番号を初期化する
line_number = 1
# 行数分繰り返す
for ___カ___:
  # 読み込んだ行に行番号を付ける
  disp_line = ___キ___
  # 表示
  print(disp_line)
  # 行番号をインクリメント
  line_number += 1
```

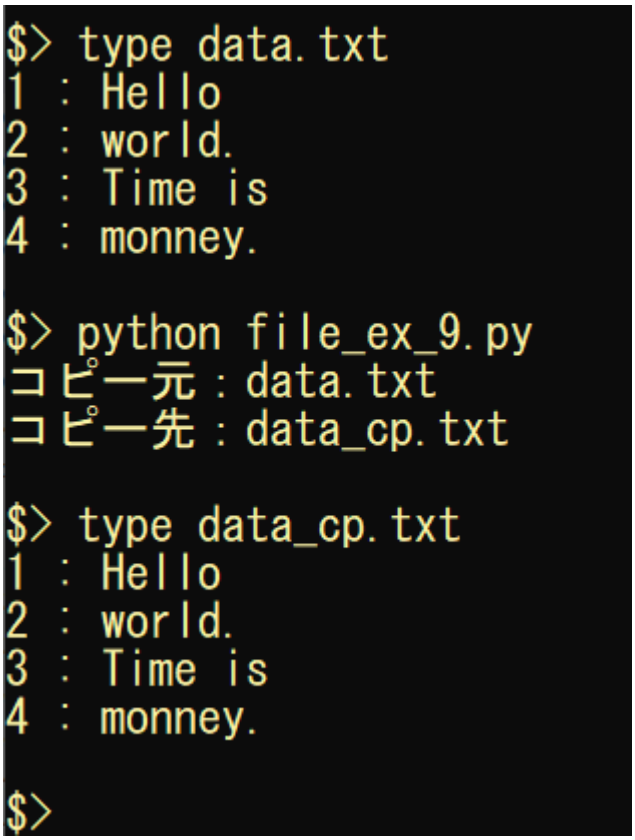
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |

## 演習 15-11

下記のプログラムはキーボードから入力されたファイル名のファイルから、キーボードから入力されたファイル名のファイルへコピーするものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# ファイルをコピーする

# コピー元のファイル名を入力
read_file_name = input('コピー元:')
# コピー先のファイル名を入力
write_file_name = input('コピー先:')
```

```
# コピー元のファイルを読み込みモードでオープン
read_file = open(___ア___, ___イ___, ___ウ___)
# コピー先ファイルを書き込みモードでオープン
write_file = open(___エ___, ___オ___, ___カ___)

# コピー元からデータを読み込む
raw_data = read_file.___キ___
# コピー先へデータを書き込む
write_file.___ク___

# ファイルをクローズ
read_file.___ケ___
write_file.___ケ___
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |
| ク | |
| ケ | |

# 16　import

## 演習 16-01

あるプログラム内で、数学関数を集めた **math** モジュール内の **sqrt** 関数を使うことを考えます。**sqrt** 関数は、1つの引数を受け取り、その引数の平方根を返す関数です。

**sqrt** 関数を呼び出すときの命令として次のア～ウのように書いた場合、それぞれ **import** 文はどのように記述しておくべきか答えてください。

なお、ウは **import** 文によって **sqrt** 関数に別名「**root**」と付け替えたことを前提としています。

ア **math.sqrt(4)**
イ **sqrt(4)**
ウ **root(4)**

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 16-02

「**from some_module import ***」のように、*を用いて **module** の関数を呼び出すと問題がある場合があります。

どのような問題が生じるか可能性があるか答えてください。

# 17　def(ユーザ定義関数)

## 演習 17-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

ひとつめの関数です。

ふたつめの関数です。

［Python コード］

```
___ア___  my_func2():
    ___イ___

___ア___  my_func1():
    ___ウ___

my_func1()
print()
my_func2()
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 17-02

以下のプログラムにはある問題がある。実行結果例のようにするには、どのように修正すればよいか答えてください。

また、その理由を答えてください。

［実行結果例］

関数を呼び出しました。

［Python コード］

```
my_func()
def my_func():
    print("関数を呼び出しました。")
```

# 18 return

## 演習 18-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
2
これは 5 です
```

［Python コード］

```
___ア___  my_func1():
    ___イ___

___ア___  my_func2():
    ___ウ___

print(my_func1() % 5)
print(my_func2() % 5)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 18-02

下記のプログラムを実行すると結果がどうなるか空欄を埋めてください。

［実行結果］

＿＿ア＿＿

＿＿イ＿＿

＿＿ウ＿＿

＿＿エ＿＿

［Python コード］

```
def my_func1():
    1
def my_func2():
    print(2)
def my_func3():
    return 3

print(my_func1())
print(my_func2())
print(my_func3())
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

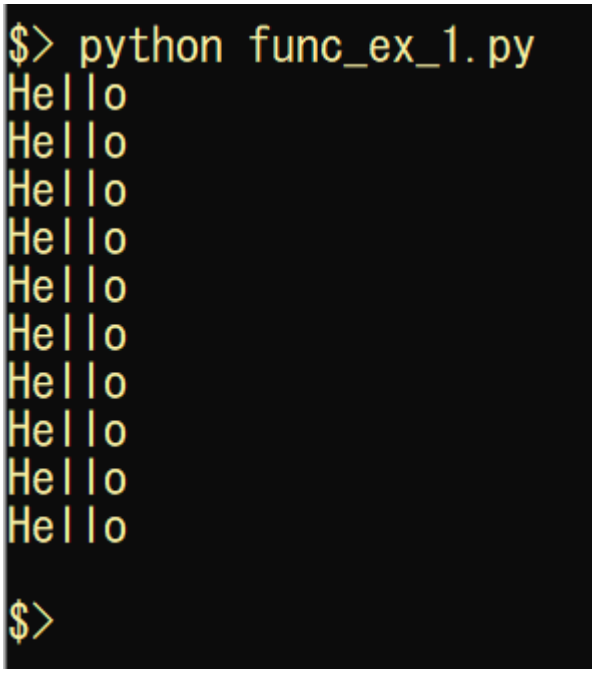
# D ユーザ定義関数総合演習

## 演習 D-01

下記のプログラムは「Hello」を10回表示する関数を定義し、実行するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 関数
# 引数なし、戻り値無し関数

# Hello を10回表示する関数の定義
___ア___ hello_10times():
  for count in ___イ___:
    print('Hello')
```

**# hello_10times** を呼び出す

___ウ___

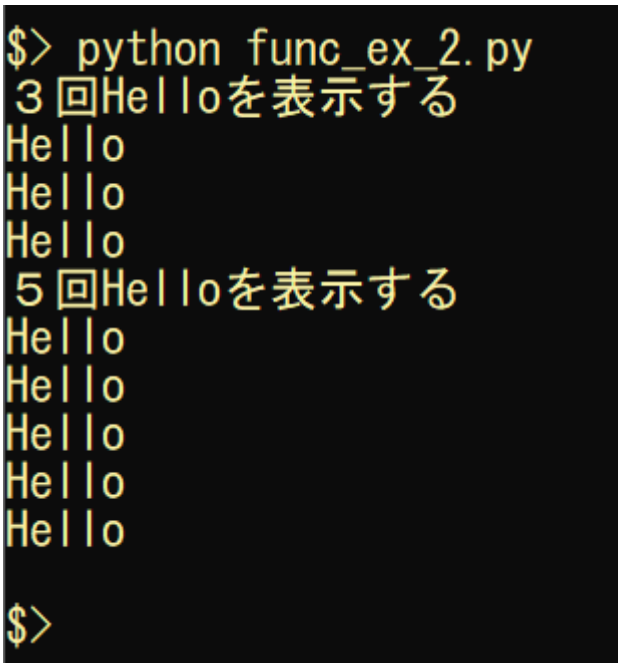
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 D-02

下記のプログラムは「Hello」を指定された回数表示する関数を定義し、実行するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# Hello を指定された回数表示する関数

# 関数 hello_times(times) の定義
def hello_times(times):
  # 指定された回数 Hello を表示する
  for count in ___ア___:
    print('Hello')

print('3回 Hello を表示する')
# hello_times を呼び出す（引数 3）
___イ___
```

```
print('5回 Hello を表示する')
```
___ウ___

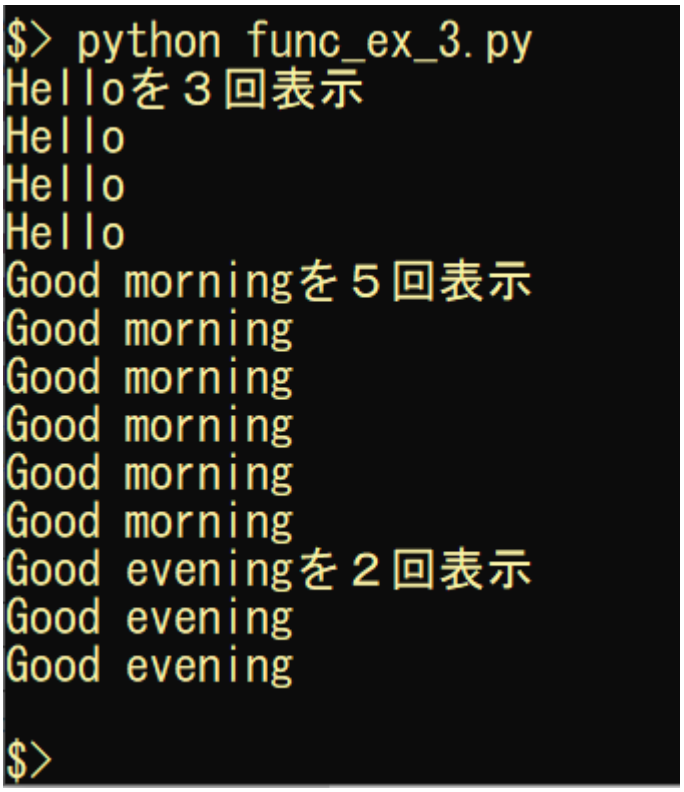
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 D-03

下記のプログラムは指定された文字列を、指定された回数表示する関数を定義し、実行するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 指定された文字列を指定された回数表示する関数

# 関数 disp_times(str, times) の定義
def disp_times(str, times):
  for count in ___ア___:
    ___イ___
```

```
# Helloを3回表示
print('Helloを3回表示')
___ウ___
# Good moringを5回表示
print('Good morningを5回表示')
___エ___
# Good eveningを2回表示
print('Good eveningを2回表示')
___オ___
```

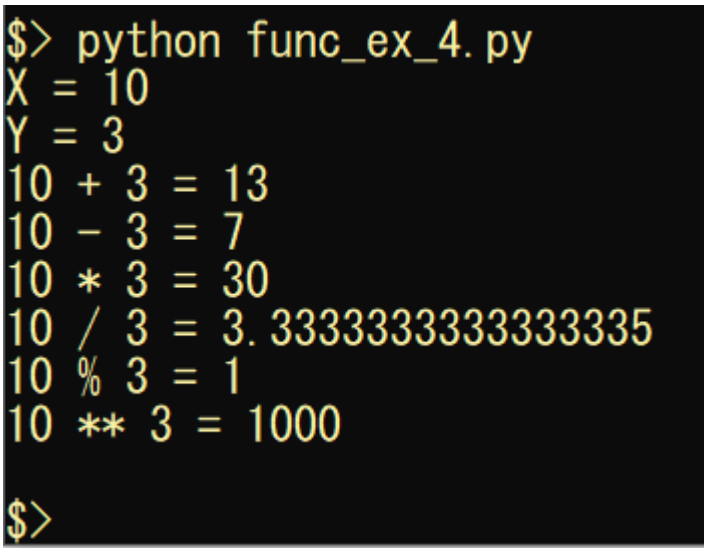
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

## 演習 D-04

下記のプログラムはキーボードから2つの整数を入力し、四則演算（足し算、引き算、掛け算、割り算、余り、べき乗）を行う関数を使用して計算し、計算結果を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 四則演算をする関数の定義

# 足し算 add(x, y)
def add(x, y):
  ___ア___

# 引き算 sub(x, y)
def sub(x ,y):
  ___イ___

# 掛け算 mul(x, y)
def mul(x, y):
  ___ウ___
```

```
# 割り算 div(x, y)
def div(x, y):
  ___エ___

# 余り   mod(x, y)
def mod(x, y):
  ___オ___

# べき乗 pow(x, y)
def pow(x ,y):
  ___カ___

# 計算結果を表示する関数
# calc_disp(x, y, op) の定義
def calc_disp(x, y, ans, op):
  # 演算子 op の判断
  if ___キ___: # 足し算 + のとき
    print('{} + {} = {}'.format(x, y, ans))
  elif ___ク___: # 引き算 - のとき
    print('{} - {} = {}'.format(x, y, ans))
  elif ___ケ___: # 掛け算 * のとき
    print('{} * {} = {}'.format(x, y, ans))
  elif ___コ___: # 割り算 / のとき
    print('{} / {} = {}'.format(x, y, ans))
  elif ___サ___: # 余り % のとき
    print('{} % {} = {}'.format(x, y, ans))
  elif ___シ___: # べき乗 ** のとき
    print('{} ** {} = {}'.format(x, y, ans))

# キーボードから2つの整数を入力
x = int(input('X = '))
y = int(input('Y = '))
```

```
# 四則演算と結果を表示
calc_disp(___ス___, '+')  # 足し算
calc_disp(___セ___, '-')  # 引き算
calc_disp(___ソ___, '*')  # 掛け算
calc_disp(___タ___, '/')  # 割り算
calc_disp(___チ___, '%')  # 余り
calc_disp(___ツ___, '**') # べき乗
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |
| ク | |
| ケ | |
| コ | |
| サ | |
| シ | |
| ス | |
| セ | |
| ソ | |
| タ | |
| チ | |
| ツ | |

## 演習 D-05

下記のプログラムはキーボードから身長と体重を入力し、BMI 値と適正体重を計算して表示します。

ただし、BMI 値を計算する関数と、適正体重を計算する関数を定義して、それらの関数を使用して実行します。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

注）

- BMI ＝ 体重 kg ÷ (身長 m)²
- 適正体重 ＝ (身長 m)² ×22

［実行結果］

```
$> python func_ex_5.py
体重(kg) = 54.5
身長(cm) = 168.6
身長：54.5、体重：1.686 のBMI値は 19.172615454324145 です。
適正体重は 62.537112kg です。

$>
```

［Python コード］

```
# BMI 値と適正体重を計算する関数

# BMI 値を計算する関数
# calc_bmi(weight, height) の定義
def calc_bmi(weight, height):
  bmi = ___ア___
  ___イ___

# 適正体重を計算する関数
# calc_proper_weight(height) の定義
def calc_proper_weight(height):
  proper_weight = ___ウ___
  ___エ___
```

```
# 身長と体重を入力
weight = ___オ___(input('体重(kg) = '))
height = ___オ___(input('身長(cm) = '))
# 身長(cm)をメートルに換算
height /= 100
# 関数を呼び出し、BMI 値を計算
bmi = ___カ___
# 関数を呼び出し、適正体重を計算
proper_weight = ___キ___
# 表示用テンプレートの作成
result_bmi = '身長:{}、体重:{} の BMI 値は {} です。
'.format(weight, height, bmi)
result_proper_weight = '適正体重は {}kg です。'.format(proper_weight)
# 表示
print(result_bmi)
print(result_proper_weight)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |

## 演習 D-06

下記のプログラムは「演習 D-05」のプログラムの機能に加えて、肥満度を表示します。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

ただし、前問のプログラムにおける関数定義部分を obesity.py として別ファイルとして保存してインポートしてください。

注)

- BMI ＝ 体重 kg ÷ (身長 m)²
- 適正体重 ＝ (身長 m)² ×22
- 肥満度

世界保健機関(WHO)の基準

| BMI 値 | 判定 |
|---|---|
| 16 未満 | 痩せすぎ |
| 16.00～16.99 以下 | 痩せ |
| 17.00～18.49 以下 | 痩せぎみ |
| 18.50～24.99 以下 | 普通体重 |
| 25.00～29.99 以下 | 前肥満 |
| 30.00～34.99 以下 | 肥満(1 度) |
| 35.00～39.99 以下 | 肥満(2 度) |
| 40.00 以上 | 肥満(3 度) |

［実行結果］

```
$> python func_ex_6.py
体重(kg) = 98.6
身長(cm) = 165.4
身長：98.6、体重：1.6540000000000001 のBMI値は 36.04175287200864 です。
適正体重は 60.18575200000001kg です。
あなたは、肥満（２度） です。

$>
```

［Python コード］

```
# 肥満度を判断する

# BMI 値と適正体重を計算する関数をインポート
from obesity import ___ア___

# 身長と体重を入力
weight = ___イ___(input('体重(kg) = '))
height = ___イ___(input('身長(cm) = '))
# 身長(cm)をメートルに換算
height /= 100
# 関数を呼び出し、BMI 値を計算
bmi = ___ウ___
# 関数を呼び出し、適正体重を計算
proper_weight = ___エ___

# 肥満度判定結果を初期化
degree_of_obesity = ''
# 肥満度を判定
if ___オ___:
  degree_of_obesity = '肥満(3度)'
elif ___カ___:
  degree_of_obesity = '肥満(2度)'
elif ___キ___:
  degree_of_obesity = '肥満(1度)'
elif ___ク___:
  degree_of_obesity = '前肥満'
elif ___ケ___:
  degree_of_obesity = '普通体重'
elif ___コ___:
  degree_of_obesity = '痩せぎみ'
elif ___サ___:
  degree_of_obesity = '痩せ'
else:
```

```
  degree_of_obesity = '痩せすぎ'

# 結果の表示
# 表示用テンプレートの作成
result_bmi = '身長：{}、体重：{} の BMI 値は {} です。
'.format(weight, height, bmi)
result_proper_weight = '適正体重は {}kg です。'.format(proper_weight)
result_obesity = 'あなたは、{} です。'.format(degree_of_obesity)
# 表示
print(result_bmi)
print(result_proper_weight)
print(result_obesity)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |
| ク | |
| ケ | |
| コ | |
| サ | |

## 演習 D-07

下記のプログラムはリストの長さ(要素の数)を求めて返す関数を定義し、それを使用して各リストの長さ(要素数)を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

[実行結果]

```
$> python func_ex_7.py
[1, 2, 3, 4, 5] の長さは、 5
[] の長さは、 0
['apple', 'orange', 'grape'] の長さは、 3
[1, 'apple', 3.14, 'PAI'] の長さは、 4

$>
```

[Python コード]

```
# リストの要素の長さを返す関数

# my_len(list) の定義
def my_len(list):
  # 長さの初期化
  length = 0
  for element in ___ア___:
    # length をインクリメント
    ___イ___
  ___ウ___

# リストの初期化
list1 = [1, 2, 3, 4, 5]
list2 = []
list3 = ['apple', 'orange', 'grape']
list4 = [1, 'apple', 3.14, 'PAI']
# リストの長さを表示
```

```
print(list1, 'の長さは、', ___エ___)
print(list2, 'の長さは、', ___オ___)
print(list3, 'の長さは、', ___カ___)
print(list4, 'の長さは、', ___キ___)
```

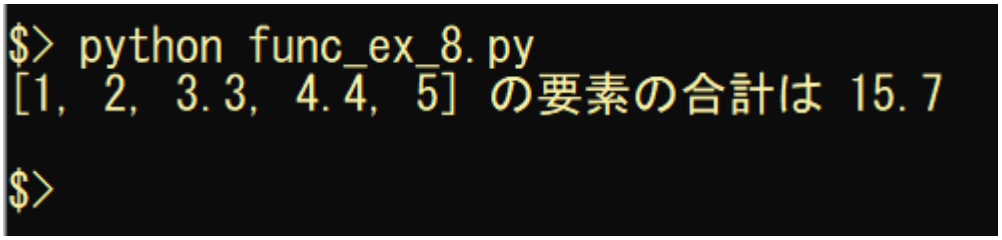
| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |

## 演習 D-08

下記のプログラムはリストの要素の合計値を求めて返す関数を定義し、それを使用してリストの要素の合計値を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# 数値リストの合計を計算する関数

# list_sum(list) の定義
def list_sum(list):
  # 合計の初期化
  total = 0
  # リストの要素数分繰り返す
  for element in ___ア___:
    # 合計を求める
    ___イ___
  # 合計を返す
  ___ウ___

# リストの初期化
list = [1, 2, 3.3, 4.4, 5]
# 合計を計算し表示
print(list, 'の要素の合計は', ___エ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

## 演習 D-09

下記のプログラムはリストの要素に指定された要素が存在するか否かを判断する関数を定義し、それを使用してリストに指定された要素が存在するか否かを表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python func_ex_9.py
整数値を入力：7
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] に 7 が存在する。

$> python func_ex_9.py
整数値を入力：100
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] に 100 は存在しない。

$>
```

［Python コード］

```
# リストの要素を検索する関数

# list_search(list, element) の定義
def list_search(list, element):
  # リストの要素数分繰り返す
  for value in ___ア___:
    # リストに要素が存在するか？
    if ___イ___:
      # 存在する
      return True
  # 存在しない
  return False

# リストの初期化
list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
# 検索する要素を入力
element = int(input('整数値を入力:'))
# リストに入力した整数値があるか否かを判断
if ___ウ___:
  print(list, 'に', element, 'が存在する。')
else:
  print(list, 'に', element, 'は存在しない。')
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 D-10

下記のプログラムはリスト内の奇数値を取り出して奇数値のみのリストを生成して返す関数を定義し、その関数を使用して指定されたリストから奇数値を要素とするリストを表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python func_ex_10.py
元のリストは [10, 11, 12, 30, 33, -55, 14, 22, 90, 89]
奇数値のリストは [11, 33, -55, 89]

$>
```

［Python コード］

```
# 奇数値を要素とするリストを返す関数

# create_odd_list(list) の定義
def create_odd_list(list):
  # 奇数値リストの初期化
  odd_list = []
  # リストの要素数分繰り返す
  for element in ___ア___:
    # 奇数か否かを判定
    if ___イ___:
      # 奇数なら奇数値リストに要素を追加
      ___ウ___
  # 奇数値リストを返す
 ___エ___

# リストを初期化
list = [10, 11, 12, 30, 33, -55, 14, 22, 90, 89]
# リスト全体の表示
print('元のリストは', list)
```

```
# 奇数値のリストを表示
print('奇数値のリストは', ___オ___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

## 演習 D-11

下記のプログラムは指定されたリストから偶数値のみのリストと奇数値のみのリストを生成して返す関数を定義し、その関数を使用して偶数値のみのリスト、奇数値のみのリストを表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python func_ex_11.py
元のリストは [10, 11, 12, 30, 33, -55, 14, 22, 90, 89]
偶数値リストは [10, 12, 30, 14, 22, 90]
奇数値リストは [11, 33, -55, 89]

$>
```

［Python コード］

```
# 偶数値のリストと奇数値のリストをタプルにして返す関数

# create_even_odd_list(list) の定義
def create_even_odd_list(list):
  # 偶数値リストと奇数値リストの初期化
  even_list = []
  odd_list = []
  # リストの要素数分繰り返す
  for element in ___ア___:
    # 偶数か奇数かを判断
    if ___イ___:
      # 要素が偶数
      ___ウ___
    else:
      # 要素が奇数
      ___エ___
  # 偶数値リストと奇数値リストのタプルを返す
  ___オ___
```

```
# リストを初期化
list = [10, 11, 12, 30, 33, -55, 14, 22, 90, 89]
# リスト全体の表示
print('元のリストは', list)
# 偶数値リストと奇数値リストを求める
result = ___カ___
# 偶数値リストを表示
print('偶数値リストは', ___キ___)
# 奇数値リストを表示
print('奇数値リストは', ___ク___)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |
| ク | |

## 演習 D-12

下記のプログラムは辞書のキーとバリューを表示する関数を定義し、それを利用して辞書の内容を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python func_ex_12.py
key[青森県] : value[1249000]
key[秋田県] : value[969500]
key[岩手県] : value[1229000]
key[山形県] : value[1080000]
key[宮城県] : value[2306000]
key[福島県] : value[1848000]

$>
```

［Python コード］

```
# 辞書のキーとバリューを表示する関数

# disp_dict(dict) の定義
def disp_dict(dict):
  # 辞書の要素数分繰り返す
  for ___ア___:
    # 表示用テンプレートの作成
    tpl = 'key[{}]:value[{}]'.format(key, ___イ___)
    # 辞書のキーとバリューを表示
    print(tpl)

# 辞書の初期化
tohoku = {
  '青森県' : 1249000,
  '秋田県' : 969500,
```

```
  '岩手県' : 1229000,
  '山形県' : 1080000,
  '宮城県' : 2306000,
  '福島県' : 1848000
}

# 辞書のキーとバリューを表示
___ウ___
```

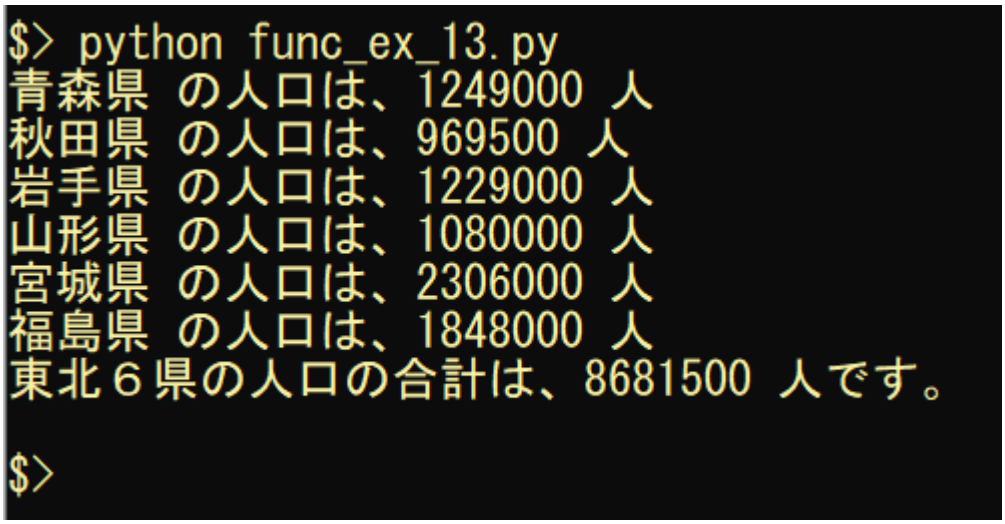
| | |
|---|---|
| ア | |
| イ | |
| ウ | |

## 演習 D-１３

下記のプログラムは東北の情報（辞書）の内容を表示する関数と、県の人口の合計を求める関数を定義して、それらを利用し東北地方の県の情報と人口の合計を表示するものです。

下記の Python コードのコメント部分に従って Python コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］




［Python コード］

```
# バリューの合計を求める関数

# 県の人口の合計を求める関数の定義
def prefecture_total(prefecture):
  # 辞書の要素数分繰り返す
  # 合計を初期化
  total = 0
  for ___ア___:
    # 人口を合計する
    ___イ___
  ___ウ___

# 県名と人口を表示する関数
def disp_prefecture(prefecture):
```

```
  # 辞書の要素数分繰り返す
  for ___エ___:
    # 表示用テンプレートの作成
    tpl = '{} の人口は、{} 人'.format(name, ___オ___)
    # 県名と人口を表示
    print(tpl)

# 辞書の初期化
tohoku = {
  '青森県' : 1249000,
  '秋田県' : 969500,
  '岩手県' : 1229000,
  '山形県' : 1080000,
  '宮城県' : 2306000,
  '福島県' : 1848000
}

# 県名と人口を表示
___カ___
# 東北の人口の合計を求める
total_population = ___キ___
# 表示用テンプレートの作成
tpl = '東北6県の人口の合計は、{} 人です。'.___ク___
# 人口の合計を表示
print(tpl)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |

| | |
|---|---|
| カ | |
| キ | |
| ク | |

## 演習 D-１４

下記のプログラムは個人の成績を入力する関数と、個人の成績を表示する関数を定義して、成績を表示するものです。

ただし、名前として「**end**」が入力された時に入力を終えるものとします。

下記の **Python** コードのコメント部分に従って **Python** コードを入力し、下記の実行結果となるようにプログラムを完成させてください。

［実行結果］

```
$> python func_ex_14.py
名前：Yoshida
国語の得点：80
算数の得点：60
理科の得点：90
名前：Yamada
国語の得点：80
算数の得点：90
理科の得点：75
名前：Nakano
国語の得点：100
算数の得点：50
理科の得点：40
名前：end
名前 = Yoshida：国語の得点 = 80：算数の得点 = 60：理科の得点 = 90
名前 = Yamada：国語の得点 = 80：算数の得点 = 90：理科の得点 = 75
名前 = Nakano：国語の得点 = 100：算数の得点 = 50：理科の得点 = 40

$>
```

［**Python** コード］

```
# 成績管理

# 成績を入力する関数を定義
def input_score():
  # 名前として end が入力されるまで繰り返す
  # 個人成績辞書を初期化
  personal_grade = {}
  while ___ア___:
    # 名前を入力
    name = input('名前:')
```

```
    # 名前として end が入力されたか否かを判断
    if ___イ___:
      # 無限ループを抜ける
      break
    # 国語の得点を入力
    kokugo = int(input('国語の得点:'))
    # 算数の得点を入力
    sansu = int(input('算数の得点:'))
    # 理科の得点を入力
    rika = int(input('理科の得点:'))
    # 3科目の得点をタプルとして保存
    score = ___ウ___
    # 個人成績を辞書に追加
    ___エ___
  # 個人成績辞書を返す
  ___オ___

# 個人成績を表示する関数を定義
def disp_score(score):
  # 辞書の要素数分繰り返す
  for ___カ___:
    # 表示用テンプレートを生成
    tpl = '名前 = {}:国語の得点 = {}:算数の得点 = {}:理科の得点 = {}'¥
        .format(name, ___キ___, ___ク___, ___ケ___)
    # 表示
    print(tpl)

# 個人成績の初期化
pesonal_info = {}
# 個人成績を入力
personal_info = ___コ___
# 入力した成績を表示
___サ___
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |
| オ | |
| カ | |
| キ | |
| ク | |
| ケ | |
| コ | |
| サ | |

# 19　モジュール

## 演習 19-01

以下のプログラムのように、モジュールを読み込むためには、どのようなファイル名をつけて、その中にどのような関数を宣言する必要があるか答えてください。

［Python コード］

```
from my_module import my_func as mf
```

| ファイル名： |
| --- |
| 関数名： |

## 演習 19-02

以下のようなプログラムを実行すると、どのような実行結果になるか答えてください。

プログラムは3つのファイル **main.py**、**my_module1.py**、**my_module2.py** に書いているコードを記したものです。ただし、**main.py** というプログラムを実行した結果とします。

また、同じフォルダの中に、**my_module1.py** と **my_module2.py** というファイルがあるものとします。

［Python コード main.py］

```
from my_module1 import *
from my_module2 import *
my_func()
```

［Python コード my_module1.py］

```
def my_func():
    print("これは1の関数です")
```

［Python コード my_module2.py］

```
def my_func():
    print("これは2の関数です")
```

# 20　+=,-=

## 演習 20-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。ただし、解答には演算子のみを用いることとします。

［実行結果］

```
1
3
0
```

［Python コード］

```
a = 1
print(a)
a ___ア___  2
print(a)
a ___イ___  3
print(a)
```

| | |
|---|---|
| ア | |
| イ | |

## 演習 20-02

以下のプログラムを実行すると、どのような実行結果が出力されるか、空欄を埋めてください。

［Python コード］

```
a = '文字'
a += '1'
print(a)

b = [1,2]
b += [3,4]
print(b)
```

［実行結果］

＿＿ア＿＿
＿＿イ＿＿

| | |
|---|---|
| ア | |
| イ | |

# 21　slice

## 演習 21-01

以下の問題文の空欄を埋めてください。

**some_list**の値が**[1, 2, 3, 4, 5]**であると仮定します。
このとき、**some_list[___ア___]**という記述で、**[2, 3]**という部分配列を取得できます。
この部分配列の呼び出し方は、**some_list[slice(1,3)]**と同等になります。

| | |
|---|---|
| ア | |

## 演習 21-02

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
[1, 2, 3, 4, 5, 6, 7, 8]
[5, 6]
[1, 3, 5, 7]
[8, 7, 6, 5, 4, 3, 2, 1]
```

［Python コード］

```
num_lists = ___ア___
print(num_lists)
print(num_lists[___イ___])
print(num_lists[___ウ___])
print(num_lists[___エ___])
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

# 22　range

## 演習 22-01

以下のプログラムを実行すると、どのような実行結果が出力されるか、空欄を埋めてください。

［Python コード］

```
r = range(5)
print(r)
```

［実行結果］

___ア___

| | |
|---|---|
| ア | |

## 演習 22-02

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
10
8
6
4
2
```

［Python コード］

```
for ___ア___  in range(___イ___,___ウ___,___エ___):
    print(i)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |

# 23　enumerate

## 演習 23-01

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
(0, '太郎')
(1, '花子')
(2, '一郎')
```

［Python コード］

```
names = ["太郎", "花子", "一郎"]
for ___ア___ in ___イ___ (names):
    print(t)
```

| | |
|---|---|
| ア | |
| イ | |

## 演習 23-02

以下のような結果を出力するプログラム中の空欄を埋め、プログラムを完成させてください。

［実行結果］

```
1 リンゴ
2 バナナ
3 メロン
```

［Python コード］

```
fruits = ["リンゴ", "バナナ", "メロン"]
for ___ア___, ___イ___ in enumerate(___ウ___):
    print(index, name)
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |

# 24　変数のスコープ

## 演習 24-01

以下のプログラムを実行すると、どのような実行結果が出力されるか、空欄を埋めてください。

［Python コード］

```
def scope_test(some_value):
    some_value = "test"
    print(some_value)

some_value = "the first value"
scope_test(some_value)
print(some_value)
```

［実行結果］

___ア___
___イ___

| | |
|---|---|
| ア | |
| イ | |

## 演習 24-02

以下のプログラムを実行すると、どのような実行結果が出力されるか、空欄を埋めてください。

［Python コード］

```
def scope_test():
    def do_local():
        spam = "local"

    def do_nonlocal():
        nonlocal spam
        spam = "nonlocal"

    def do_global():
        global spam
        spam = "global"

    spam = "the first value"
    do_local()
    print(spam)
    do_nonlocal()
    print(spam)
    do_global()
    print(spam)

scope_test()
print(spam)
```

［実行結果］

```
___ア___
___イ___
___ウ___
___エ___
```

| | |
|---|---|
| ア | |
| イ | |
| ウ | |
| エ | |