Python によるプログラミング入門
解答と解説

※各問のファイル名は、「ex＋問題番号」とすること。
例：問 1-1　→　ファイル名は「ex1_1」
（ファイル名から拡張子の表示は省略している）

第 1 章 Python の特徴　実行方法

問 1-1

自分の名前をコマンドプロンプトに表示するプログラムを作成せよ。(以下、コマンドプロンプトに表示する、は表示すると記述する)

解答例（ソースコード）

```
print("ウチダ タロウ")
```

解説

print 関数を使うとコマンドプロンプトに文字を表示できます。日本語を書くときは全角モードに切り替えますが、半角モードに戻すのを忘れないように気を付けてください。

実行するときは、コマンドプロンプトで python ex1_1.py と入力すれば実行できます。コマンドプロンプトを開くフォルダと、プログラムを作成するフォルダが同じになるように気を付けてください。

問 1-2

実行例のように、おはようございます、こんにちは、こんばんは、と表示するプログラムを作成せよ。

解答例（ソースコード）

```
print("おはようございます")
print("こんにちは")
print("こんばんは")
```

解説

Python は基本的に一行に書く命令は一つです。改行を挟んで print 関数を 3 つ書きましょう。print 関数は、指定された文字列を表示した後に文末で自動的に改行してくれます。

第 2 章 データ型と変数

問 2-1

name という変数に自分の名前を代入し、その変数を表示するプログラムを作成せよ。

解答例（ソースコード）

```
name = "ウチダタロウ"
print(name)
```

解説

変数名 = 右辺、と書くことで右辺の内容を変数に代入（定義）できます。定義した変数は、変数名を書くことで、そのプログラム中で利用できます。利用できない変数名もあるので、よく覚えていない場合は動画をチェックしてください。

問 2-2

pi という変数に 3 を代入して pi を表示し、その後、変数 pi を 3.14 で上書きし、pi を表示するプログラムを作成せよ。

解答例（ソースコード）

```
pi = 3
print(pi)
pi = 3.14
print(pi)
```

解説

一度定義した変数名に値を代入すると、もともと代入されていた値を上書きすることができます。異なる型のオブジェクトで上書きすることもできます。

問 2-3

変数 a に 5、変数 b に 3 を代入し、a と b の積を表示するプログラムを作成せよ。

解答例（ソースコード）

```
a = 5
b = 3
print(a * b)
```

解説

一度定義した変数名に値を代入すると、もともと代入されていた値を上書きすることができます。
異なる型のオブジェクトで上書きすることもできます。

問 2-4

1 つ 100 円のリンゴを 3 つ、1 つ 80 円のみかんを 5 つ買ったときの合計金額を表示するプログラムを作成せよ。ただし、ソースコードは 1 行のみにすること。

解答例（ソースコード）

```
print(100*3 + 80*5)
```

解説

( )を使えば計算の優先順位を変えられます。ソースコードの行数に制限がなければ以下のよう解答でも構いません。

```
apple = 100 * 3
orange = 80 * 5
sum = apple + orange
print(sum)
```

問 2-5

name という変数に自分の名前を代入し、その変数を表示するプログラムを作成せよ。ただし、name への代入は input 関数を用いて与え、ソースコードに直接自分の名前は書かないこと。

解答例（ソースコード）

```
name = input("名前を入力してください：")
print(name)
```

解説

input を書くと、プログラムはキーボードからの入力待ちになります。括弧の中身は、入力前に表示されるメッセージです。これがなくても動作しますが、プログラムの利用者にとって分かりにくいので何かメッセージを書くようにしましょう。

問 2-6

input 関数を用いて名前を入力すると以下のようなメッセージを表示するプログラムを作成せよ。

解答例（ソースコード）

```
name = input("名前を入力してください：")
print("こんにちは　" + name + "さん" )
```

解説

print 関数を二つ書くと改行が入ってしまいます。＋演算子を用いて文字列を結合すれば、改行することなく一行で表示できます。こんにちは、と名前の間に空白いれるためには、ダブルクオーテーションの中に空白を入れればOK です。

問 2-7

input 関数を用いて、変数 a と変数 b に任意の文字列を代入し、その後 a と b の値を交換して、a と b の表示するプログラムを作成せよ。

解答例（ソースコード）

```
a = input("変数 a に代入する文字列を入力してください：")
b = input("変数 b に代入する文字列を入力してください：")
tmp = a
a = b
b = tmp
print("変数 a: " + a)
print("変数 b: " + b)
```

解説

変数の中身を交換する方法を考える問題です。a=b, b=a と書いてしまうと、一回目の代入で元々の a の値が消えてしまいます。そのため、tmp = a のように別の変数に退避させておくことがポイントです。

また、動画中では扱っていない方法ですが、以下のように書いても変数の中身を交換できます。

```
a = input("変数 a に代入する文字列を入力してください：")
b = input("変数 b に代入する文字列を入力してください：")
a, b = b, a
print("変数 a: " + a)
print("変数 b: " + b)
```

第 3 章 コレクション

問 3-1

"バナナ"、"リンゴ"、"ミカン"、"イチゴ"を要素として持つリストを作成し、全ての要素を表示させた後、リストの 2 つ目の要素（この例だと”リンゴ”）を表示するプログラムを作成せよ。
注意：インデックスは 0 番から振られる。

解答例（ソースコード）

```
fruits = ["バナナ", "リンゴ", "ミカン", "イチゴ"]
print(fruits)
print(fruits[1])
```

解説

リストの作成方法は、変数名 = [要素, 要素,・・・]です。print(変数名)と書くと、リストの中の全ての要素を表示できます。特定の番号の要素にアクセスするには、インデックスを指定します。この問題の例だと、fruits[1]と書くことで、1 番目のインデックスの要素にアクセスすることができます。インデックスは 0 番から振られるので、先頭から数えて 2 つ目の要素にアクセスできます。

問 3-2

"バナナ"、"リンゴ"、"ミカン"、"イチゴ"を要素として持つリストを作成し、全ての要素を表示させた後、1 番目のインデックスの要素をリストから削除し、新たに”ブドウ”という要素を追加し、再びすべての要素を表示するプログラムを作成せよ。

解答例（ソースコード）

```
fruits = ["バナナ", "リンゴ", "ミカン", "イチゴ"]
print(fruits)
fruits.pop(1)
fruits.append("ブドウ")
print(fruits)
```

解説

fruits.pop(1)と書くと、fruits というリストから 1 番目の要素であるリンゴを削除できます。
fruits.append(“ブドウ”)と書くと、fruits というリストの末尾にブドウという要素を追加できます。

問 3-3

key と value のペアが、"バナナ"：100、"リンゴ"：200、"ミカン"：80 の要素を持つ辞書を作成し、全ての要素を表示させた後、key として”ミカン”を指定して value を表示するプログラムを作成せよ。

解答例（ソースコード）

```
fruits = {"バナナ":100, "リンゴ":200, "ミカン":80}
print(fruits)
print(fruits["ミカン"])
```

解説

辞書の作成方法は、変数名 ={key 名:value,・・・}です。print(変数名)と書くと、辞書の中の全ての要素を表示できます。また、変数名[key]とすることで、key を指定して value を表示することができます。には、インデックスを指定します。

問 3-4

key と value のペアが、"バナナ"：100、"リンゴ"：200、"ミカン"：80 の要素を持つ辞書を作成し、全ての要素を表示させた後、key として”ミカン”を持つ要素を辞書から削除し、新たに”ブドウ”：300 を要素として辞書に追加し、再びすべての要素を表示するプログラムを作成せよ。

解答例（ソースコード）

```
fruits = {"バナナ":100, "リンゴ":200, "ミカン":80}
print(fruits)

fruits.pop("ミカン")
fruits["ブドウ"] = 300
print(fruits)
```

解説

fruits.pop("ミカン")と書くと、fruits という辞書からミカンとその value である 80 を削除できます。

fruits["ブドウ"] = 300 と書くと、fruits という辞書の末尾にブドウという要素を追加できます。

問3-5

"バナナ"、"リンゴ"、"ミカン"、"イチゴ"を要素として持つリストを作成し、そのリストの長さを表示するプログラムを作成せよ。

解答例（ソースコード）

```
fruits = ["バナナ", "リンゴ", "ミカン", "イチゴ"]
print(len(fruits))
```

解説

len 関数を使うとリストや辞書の長さを得られます。また、len()の括弧の中に文字列を入れるとその文字列の長さを得られます。

第 4 章 条件分岐

問 4-1

input 関数で整数(年齢)を入力し、年齢が 18 歳未満であれば、「未成年です」と表示するプログラムを作成せよ。この問題以降、input 関数で整数を入力させる指示があった場合、整数以外が入力されることを想定する必要はない。(整数以外を入力したときにエラーとなっても構わない)

解答例（ソースコード)

```
age_str = input("年齢を入力してください:")
age = int(age_str)
if age < 18:
        print("未成年です")
```

解説

if 文の演習問題です。コロンの書き忘れやインデントのズレに注意してください。18 歳未満が条件なので、条件式を <= としてはいけません。補足にも記載しましたが、input 関数で入力した整数は文字列となってしまうので、そのまま数値の比較ができません。int 関数で int 型に変換してから条件式で使いましょう。

問4-2

input 関数で文字列を入力し、その文字列が"おはよう"に等しければ、「午前です」を表示するプログラムを作成せよ。

解答例（ソースコード）

```
greeting = input("あいさつしてください:")
if greeting == "おはよう":
        print("午前です")
```

解説

if 文の演習問題です。条件式では、文字列を比較することもできます。挨拶の表現はこれ以外にもありますが、これ以外の表現に対応したプログラムを書くのも良い練習になると思います。

問 4-3

input 関数で整数を入力し、その値が 1 ならば「天気は晴れです」、2 ならば「天気は曇りです」、それ以外ならば「天気は雨です」と表示するプログラムを作成せよ。

解答例（ソースコード）

```
weather = input("整数を入力してください:")
if weather == "1":
        print("天気は晴れです")
elif weather == "2":
        print("天気は曇りです")
else:
        print("天気は雨です")
```

解説

elif 節と else 節の演習問題です。elif 節は if 文とセットで記述します。elif 節の条件式の書き方は if 文と同様ですが、if 文やその elif 節より上の行に書かれている他の elif 文の条件式を満たさなかったときのみ、判定が行われます。else 節も if 文とセットで記述されます。else 節は他の if 文や elif 節の条件式を満たさなかったときのみ、処理が実行されます。

今回の演習では入力を整数としていますが、4-1 と異なり int 関数で整数に変換していません。それは、入力された値の大小を条件で比較していないためです。入力された値が特定の値であるかどうかは、文字列のままでも==を使えば判定できるので、整数に変換していません。ただし、文字列の 1 と等しいかどうか判定するので、判定式は、

```
if weather == "1":
```

のように 1 にダブルクオーテーションが付いていることに注意してください。

問 4-4

input 関数で国語と数学のテストの点数を整数で入力し、数学の点が 90 点以上かつ国語の点が 80 点以上ならば「合格です」、それ以外ならば「不合格です」と表示するプログラムを作成せよ。なお、数学 3000 点と国語 1000 点など 100 点より大きい整数を入力しても合格と表示しても構わない。(100 点より大きい整数について条件を分ける必要はない)

解答例（ソースコード）

```
math_str = input("数学の点数を入力してください:")
math = int(math_str)
japanese_str = input("国語の点数を入力してください:")
japanese = int(japanese_str)

if math >= 90:
        if japanese >= 80:
                print("合格です")
        else:
                print("不合格です")
else:
        print("不合格です")
```

解説

ネストの演習問題です。if 文の中にさらに if 文を書くことで、より詳細な条件分岐ができます。数学が 90 点以上かつ国語が 80 点以上という条件は、学習済みの方法ではネストを使わないと実現できません。動画では取り扱っていない内容ですが、一般的にはこのような条件式は以下のように and 演算子を使って実現することが多く、見た目にも分かりやすいです。

```
math_str = input("数学の点数を入力してください:")
math = int(math_str)
japanese_str = input("国語の点数を入力してください:")
japanese = int(japanese_str)

if math>=90 and japanese>=80:
        print("合格です")
else:
        print("不合格です")
```

問 4-5

input 関数で西暦（整数）を入力し、それがうるう年である場合は「うるう年です」と表示し、そうでない場合は「うるう年ではありません」と表示するプログラムを作成せよ。

解答例（ソースコード）

```
year_str = input("西暦を入力してください：")
year = int(year_str)

if year % 400 == 0:
        print("うるう年です")
elif year % 100 == 0:
        print("うるう年ではありません")
elif year % 4 == 0:
        print("うるう年です")
else:
        print("うるう年ではありません")
```

解説

複雑な条件の if 文の演習問題です。実現方法はいろいろあるので、様々な値を入力して自分のプログラムが正しく動作しているか確認してください。400 で割り切れるかどうかという厳しい条件式を先に書くことで、ネストを使わずに書くことができます。もちろん、問題文に条件指定はありませんので、ネストを使って書いても正解です。ある数で割り切れるかは、%演算子を使ってあまりを計算し、それが 0 かどうかを判定すれば OK です。

第5章 繰り返し

問5-1

0から9までの整数を順に表示するプログラムを while 文を利用して作成せよ。

解答例（ソースコード）

```
i = 0
while i < 10:
        print(i)
        i = i + 1
```

解説

while 文の演習問題です。while 文は条件式を満たしている間は、処理を繰り返します。条件式の書き方は if 文と同じです。for 文と異なり、条件式を満たさなくする処理を加えないと無限ループに陥るので気を付けましょう。

問 5-2

"バナナ"、"リンゴ"、"ミカン"、"イチゴ"を要素として持つリストを作成した後、以下の実行例のように表示するプログラムを for 文を利用して作成せよ。

解答例（ソースコード）

```
fruits = ["バナナ", "リンゴ", "ミカン", "イチゴ"]
for fruit in fruits:
        print(fruit + "を食べます")
```

解説

for 文の演習問題です。for 文はシーケンスが持つオブジェクトに対して、繰り返し同じ処理を行えます。ここでは、“を食べます”という文字列と結合して画面に表示しています。if 文や while 文と異なり、for の隣に書くのは、for 文の中で使う変数名：シーケンス名、となることに注意しましょう。

問 5-3

0 から 10 までの整数の内、偶数のみを順に表示するプログラムを for 文と range 関数を利用して作成せよ。

解答例（ソースコード）

```
for i in range(0,11,2):
        print(i)
```

解説

range 関数を用いた for 文の演習問題です。range 関数を使うことで等差数列を作れます。開始の数、終了の数、ステップを指定してオブジェクトを作成します。開始の数とステップは省略可能で、どのような動作になるか認識があいまいな人は再度動画で確認してください。ここでは、開始の値 0、終了の値 11、ステップ 2 を指定することで 0,2,4,6,8,10 という数列を作成しています。終了の値を 10 にすると数列に 10 が含まれなくなることに気を付けましょう。少し複雑になってしまいますが、if 文を使って以下のように書いても同様の表示が得られます。

解答例（ソースコード）

```
for i in range(11):
        if i%2 == 0:
                print(i)
```

問 5-4

input 関数を用いて正の整数を 5 回入力し、その後、その中で最も大きい値のみを表示するプログラムを作成せよ。

解答例（ソースコード）

```
max = 0
for i in range(5):
        tmp_str = input("正の整数を入力してください:")
        tmp = int(tmp_str)
        if max < tmp:
                max = tmp
print(max)
```

解説

繰り返しと条件分岐の組み合わせの演習です。5 つの整数の内、最大となるものを探すアルゴリズムを考える問題です。今回は、最大となるものだけを最後に表示すればいいので、max というここまでの最大値と入力された値を比較し、より大きければ代入するという処理を、for 文の中で繰り返し実行しています。最初の値は比較するものがないので、あらかじめ max には 0 を代入しておきます。

問 5-5

0 が入力されるまで input 関数を用いて整数を繰り返し入力し、最後に入力された数値の合計値を表示するプログラムを作成せよ。

解答例（ソースコード）

```
total = 0
while True:
        num_str = input("整数を入力してください：")
        num = int(num_str)
        if num != 0:
                total = num + total
        elif num == 0:
                break
print(total)
```

解説

繰り返しと条件分岐の組み合わせの演習です。合計値を保存する変数を while 文の外側で定義しておき、while 文の内側で足していきます。while 文の条件式に True とだけ書いてありますが、こう書くと while 文は無限ループとなります。そのため、elif 節で break を書いてループを中断できるようにします。以下のように書けば、while True や break を使わなくても実現できます。

```
total = 0
num = 1
while num != 0:
        num_str = input("整数を入力してください：")
        num = int(num_str)
        if num != 0:
                total = num + total
print(total)
```

第 6 章 関数

問 6-1

以下の要件を満たすプログラムを作成せよ。

- 三角形の底辺と高さを引数とし、面積を戻り値とする get_triangle_area 関数を定義する。
- input 関数を用いて底辺と高さを入力し、get_triangle_area 関数を用いて面積を計算し、計算結果を画面に表示する。

解答例（ソースコード）

```
def get_triangle_area(base, height):
        return base * height / 2

base_str = input("底辺を入力してください：")
teihen = int(base_str)
height_str = input("高さを入力してください：")
takasa = int(height_str)

area = get_triangle_area(teihen, takasa)
print(area)
```

解説

関数を定義してそれを利用する演習です。関数を定義するときに、関数名、引数、戻り値をどのように書くのかを動画で確認してください。引数に用いられている base と height はローカル変数なので、関数の外側では別の変数として同じ名前を利用できます。今回は違いを分かりやすくするため、あえて 5 行目で teihen、7 行目で takasa という変数名を付けていますが、これは base と height でも問題なく動作します。

問 6-2

以下の要件を満たすプログラムを作成せよ。

- 三角形の底辺と高さを引数とし、その面積を計算して画面に表示する print_triangle_area 関数を定義する。なお、戻り値はなしとする。
- input 関数を用いて底辺と高さを入力し、print_triangle_area 関数を用いて面積を画面に表示する。

解答例（ソースコード）

```
def print_triangle_area(base, height):
        area = base * height / 2
        print(area)

base_str = input("底辺を入力してください：")
base = int(base_str)
height_str = input("高さを入力してください：")
height = int(height_str)

print_triangle_area(base, height)
```

解説

関数を定義してそれを利用する演習です。6-1 と異なり関数の戻り値がありません。今回のように戻り値が不要な場合は定義する必要はありません。

問 6-3

以下の要件を満たすプログラムを作成せよ。

- 自然数を引数とし、それが素数であれば True、素数でなければ False を戻り値とする check_prime 関数を定義する。なお、この関数への引数として 1 は想定しなくてもよい。
- input 関数を用いて自然数を入力し、check_prime 関数を用いて素数の判定をし、素数であれば「素数です」、素数でなければ「素数ではありません」と画面に出力する。

解答例（ソースコード）

```
def check_prime(num):
        for i in range(2, num):
                if num%i == 0:
                        return False
        return True

num_str = input("2 以上の整数を入力してください：")
num = int(num_str)

if check_prime(num):
        print("素数です")
else:
        print("素数ではありません")
```

解説

総合的な演習です。素数判定の方法がポイントですが、ここでは 2 から与えられた引数まで全ての値で引数を割り、割り切れる値があるかをチェックしています。未学習の方法ですが、if 文の条件式にそのまま関数の呼び出しを書いています。これは、check_prime の戻り値が bool 型のオブジェクトなので可能な方法です。

問 6-4

以下の要件を満たすプログラムを作成せよ。

- 2 つの整数を引数とし、それらの数が小さい方を戻り値とする get_min 関数を定義する。
- 2 つの整数を引数とし、それらの最大公約数を戻り値とする get_gcd 関数を定義する。なお、この関数への引数として 1 以下の値は想定しなくてもよい。また、この関数の内部で get_min 関数を利用すること。
- input 関数を用いて 2 つの整数を入力し、get_gcd 関数を用いて最大公約数を計算し、計算結果を画面に出力する。

解答例（ソースコード）

```
def get_min(num1, num2):
        min = num2
        if num1 < num2:
                min = num1
        return min

def get_gcd(num1, num2):
        min = get_min(num1, num2)
        gcd = 1
        for i in range(2, min+1):
                if (num1%i == 0):
                        if(num2%i == 0):
                                gcd = i
        return gcd

num_str = input("正の整数を入力してください：")
num1 = int(num_str)
num_str = input("正の整数を入力してください：")
num2 = int(num_str)
gcd = get_gcd(num1, num2)
print(gcd)
```

解説

総合的な演習です。問題文の条件にあるように、自分で定義した関数は他の関数の中でも呼び出せます。最大公約数の計算方法がポイントですが、ここでは入力した 2 つの整数について割り切れる数を 1 から順番に探しています。最大公約数はどんなに大きくても 2 つの整数の内の小さい方の値にしかなりません。そのため、1 からその小さい方の値までの整数すべてについて、割り切れる数を探していけば最大公約数は見つかります。