p.104 4章2節 プログラミングの実践

**\_\_\_\_\_\_\_\_\_\_年 \_\_\_\_\_\_\_\_\_\_組 \_\_\_\_\_\_\_\_\_\_番**

**名前 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**１ プログラミングの基礎**①

プログラミング言語では，基本的な計算や，基本制御構造をどのように表現しているだろうか。

ダイアグラム, アイコン

自動的に生成された説明 **１ 変数を使用したプログラム**

・ メモリ上のデータ(値)を格納する領域を（ 　 　　 ）といい，

その領域に付けた名前を(　 　　　 　 )という。

・ 変数に値を格納することを( 　　 　)という。

・ Pythonの基本的な命令文

　　　print(値)・・・（　）内の値を（ 　　 　）する。

　　　input(文字列)・・・（　）内の文字列を案内として表示し，ユーザに値を（ 　　 　）させる。

　　　int(変数)・・・（　）内の変数の値を（ 　　 　）型に変換する。

・ Pythonの基本的な演算子

|  |  |  |  |
| --- | --- | --- | --- |
| 加算 | （　　　　） | 余り | （ 　 　） |
| 減算 | （ 　　 ） | 商の整数部分 | （ ） |
| 乗算 | （ 　 　） | べき乗 | （ 　 ） |
| 除算 | （ 　 　） |  |  |

例題１ 　**二つの値を加算するプログラム**

ユーザが入力した二つの整数を加算し，結果を表示するプログラムを作成してみよう。

**アルゴリズム**

① aにユーザが入力した数値を（ 　 ）

② bにユーザが入力した数値を代入

③ cに（　 　　 　）の結果を代入

④ 文字列'a+bは'を表示

⑤ cを表示

**プログラム**

① a = int(input('数字aを入力'))

② b = int(input('数字bを入力'))

③ 　 　　 高さ調整

④ print('a + bは')

⑤ print(c)

確認問題１

減算，乗算，除算の結果も表示されるように，例題1のプログラムを変更してみよう。

1. a = int(input('数字aを入力'))
2. b = int(input('数字bを入力'))
3. c = a + b
4. print(‘a + bは')
5. print(c)

**目標に対する達成度：** よくできた ・ できた ・ ある程度できた ・ できなかった

**学習の振り返り**

プログラミング言語では，基本的な計算をどのように表現しているか，理解できたことを記述しよう。

p.105 4章2節 プログラミングの実践

**\_\_\_\_\_\_\_\_\_\_年 \_\_\_\_\_\_\_\_\_\_組 \_\_\_\_\_\_\_\_\_\_番**

**名前 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**１ プログラミングの基礎**②

プログラミング言語では，基本的な計算や，基本制御構造をどのように表現しているだろうか。

**２ 分岐構造を使用したプログラム**

・ プログラムの流れを変える命令文を( 　 　　　 )といい，

分岐構造で使用する制御文を( 　 　　　 )という。

・ if文の記述方法

　 if 条件１：

　　 処理１

（ 　　 ） 条件２：

　　 処理２

（ ）:

　　 処理３

例題２ 　**偶数か奇数か判定するプログラム**

ユーザが入力した整数が，偶数か奇数，もしくはゼロか判定するプログラムを作成してみよう。

**アルゴリズム**

① ユーザが入力した数値を整数型に変換し，変数aに代入

② もしaが（ 　　 ）と等しければ

③ 'ゼロ'と表示

④ そうでなくもしaを2で割った（ 　 ）が0ならば

⑤ '偶数'と表示

⑥ そうでなければ

⑦ '奇数'と表示

**プログラム**

① a = int(input('数字を入力'))

② 　　　　　　　　　 高さ調整

③ print('ゼロ')

④ 　　　　　　　　　 　さ調整

⑤ print('偶数')

⑥ 高さ調整

⑦ print('奇数')

確認問題２

ユーザが入力した数値が，3の倍数か，5の倍数か，3と5の倍数であるか，それ以外の数であるか判定し，その結果を「3の倍数です」のように表示するプログラムを作成してみよう。

**目標に対する達成度：** よくできた ・ できた ・ ある程度できた ・ できなかった

**学習の振り返り**

このプリントで学習した基本的な制御構造について，どのような表現をしているか，理解できたことを記述しよう。

p.106 4章2節 プログラミングの実践

**\_\_\_\_\_\_\_\_\_\_年 \_\_\_\_\_\_\_\_\_\_組 \_\_\_\_\_\_\_\_\_\_番**

**名前 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**１ プログラミングの基礎③**

プログラミング言語では，基本的な計算や，基本制御構造をどのように表現しているだろうか。

**３ 反復構造を使用したプログラム**

・ 繰り返しの回数があらかじめ定まっている場合に適している反復構造の制御文を ( 　 　　　　 )といい，

回数が定まっていない場合に適している反復構造の制御文を ( 　 　　　　　　 )という。

例題3-1 　**任意の九九の段を表示するプログラム**

入力した数字に対し，1から9までの数をかけた結果を順に表示するプログラムを作成してみよう。

**アルゴリズム**

① ユーザが入力した数値を整数型に変換し変数aに代入

② iを( )から( )まで1ずつ増やしながら繰り返し

③（ 　）を表示

**プログラム**

① a = int(input('数字を入力'))

② 高さ調整

③ print( 高さ調整 )

例題3-2 　**パスワードを確認するプログラム**

正しいパスワード（Joho1とする）をユーザが入力するまで，入力させ続けるプログラムを作成してみよう。

**プログラム**

① a = 'Joho1'

② b = input('パスワードを入力')

③ 高さ調整

④ 　b = input('パスワードを再入力')

⑤ print('正しく入力されました')

**アルゴリズム**

① 正しいパスワードをaに代入

② ユーザが入力したパスワードをbに代入

③ aとbが（ 　　　　　　　）間，④を繰り返し

④ パスワードをユーザに再入力させる

⑤ 正しいパスワードが入力されたことを表示

確認問題３

ユーザが入力した数字を1ずつ0までカウントダウンするプログラムを作成してみよう。例えば，10が入力されたら，9から0まで順に表示されるようにすること。

**目標に対する達成度：** よくできた ・ できた ・ ある程度できた ・ できなかった

**学習の振り返り**

このプリントで学習した基本的な制御構造について，どのような表現をしているか，理解できたことを記述しよう。

p.107 4章2節 プログラミングの実践

**\_\_\_\_\_\_\_\_\_\_年 \_\_\_\_\_\_\_\_\_\_組 \_\_\_\_\_\_\_\_\_\_番**

**名前 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**１ プログラミングの基礎④**

プログラミング言語では，基本的な計算や，基本制御構造をどのように表現しているだろうか。

**４ 配列**

・ 複数のデータをまとめて扱えるようにし，全体に対して一つの名前を付けたものを ( 　 　　 )という。

・ 一つひとつの変数を ( 　 　　 )といい，配列の要素は配列名と（ 　　 　( 　　　　　　　　　))と呼ばれる番号で指定する。

・ Pythonにおける要素の追加方法

　　　 （ 　　　　　　　　　　　　　　）・・・（　）内で指定した値がリストの最後の要素として追加される。

・ 配列の要素の表示方法

　　　 （ 　　　　　　　　　　　　　）・・・配列中の要素がすべて表示される。

例題４ 　**ユーザが指定した九九の段を配列に格納するプログラム**

ユーザが指定した九九の段を配列に格納し，その後格納した値を表示するプログラムを作成してみよう。

**プログラム**

① 　高さ調整

② b = int(input('数字を入力'))

③ 高

④　 高

⑤ print(a)

**アルゴリズム**

① 配列aを（ 　　　）の数を決めずに宣言

② ユーザが入力した数値を整数型に変換し，変数bに代入

③ iを( )から( )まで1ずつ増やしながら④の処理を繰り返し

④ b×( )をaの要素として追加

⑤ 配列aの要素をすべて表示

確認問題４

例題4のプログラムを実行すると，添字0に格納されている値も表示されてしまう。5行目以降を変更し，添字1から9までに格納されている値を表示するプログラムを作成してみよう。

① a = []

② b = int(input('数字を入力'))

③ for i in range(0, 10):

④ 　a.append(b \* i)

⑤ for i in range( 　高さ調整):

⑥ 　 　 高さ調整

**目標に対する達成度：** よくできた ・ できた ・ ある程度できた ・ できなかった

**学習の振り返り**

プログラミング言語では，基本的な計算をどのように表現しているか，理解できたことを記述しよう。

p.108～109 4章2節 プログラミングの実践

**\_\_\_\_\_\_\_\_\_\_年 \_\_\_\_\_\_\_\_\_\_組 \_\_\_\_\_\_\_\_\_\_番**

**名前 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**２ 関数を使用したプログラム**

大規模なプログラムを効率よく作成するためには，どのようにプログラムを記述したらよいだろうか。

**１ 関数とは**

・ プログラムの処理の一部を抜き出して記述した部分を ( 　 　　 )と呼ぶ。

（ 　　 　）を受け取って一連の処理を実行し，（ 　　　　 　）を返す。

例題５ 　**自動販売機を関数と見立てたときの引数と戻り値**

関数とは，受け取った引数をもとに処理を行って結果を返すものなので，清涼飲料水の自動販売機のような機械も関数とみなすことができる。この場合，引数と戻り値はそれぞれ何にあたるか整理してみよう。

引数：（ 　　　　　 ），（ 　　　　　　 ） 戻り値：（ 　　　　　　　）

**２ 関数の定義**

例題６ 　**四捨五入の処理をするユーザ定義関数**

正の小数を小数第1位で四捨五入する，関数myroundを作成してみよう。

**四捨五入は，四捨五入をする値に（**  　**）を加えて小数点以下を（** 　　　　　　**）て求める。**

**引数と戻り値を表のように整理しておく。また，関数を呼び出すプログラムはプログラム１のようになる。**

**プログラム1**

① a = 7.4

② print（ 　　　　　　高さ調整 ）

③ b = 7.5

④ print（ 　　　　　　高さ調整 ）

**表．引数と戻り値**

|  |  |  |
| --- | --- | --- |
|  | **名前** | **意味** |
| **引数** | **arg** | ( 　　　　　　　　　　 　　) |
| **戻り値** |  | ( 　　　　　　　　　　 　　) |

関数myroundのアルゴリズムとプログラムは以下のようになる。

**アルゴリズム**

① argを（ 　）とした関数を定義

② 引数argに0.5を加えた値をaに代入

③ aの小数以下を切り捨てた値をbに代入

④ bの値を（ 　）とする

**プログラム**

① 　 高さ調整

② 　a = arg + 0.5

③ 　b = int(a)

④ 　 　高さ調整

参考 　**変数のスコープ**

（ 　　　　　 ）変数： スコープが関数内である変数

（ 　　　　　　 ）変数： スコープが関数を超え，ほかの関数からもアクセスできる変数

**３ APIとWeb API**

・ ( 　 　)とは，プログラムの機能やデータなどを外部のプログラムから利用できるようにしたものである。

・ インターネット経由でHTTPプロトコルを用いて利用できるAPIを（ ）という。

**目標に対する達成度：** よくできた ・ できた ・ ある程度できた ・ できなかった

**学習の振り返り**

大規模なプログラムを効率よく作成するためのプログラムの記述方法について，理解できたことを記述しよう。

p.110 4章2節 プログラミングの実践

**\_\_\_\_\_\_\_\_\_\_年 \_\_\_\_\_\_\_\_\_\_組 \_\_\_\_\_\_\_\_\_\_番**

**名前 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**３ 探索と整列のプログラム①**

データの探索や並べ替えを行うためには，どのようなアルゴリズムを用いればよいだろうか。また，アルゴリズムによって効率に差はあるだろうか。

**１ 線形探索のアルゴリズム**

・ ( 　　　 )とは，たくさんのデータの中から目的のデータを探し出すことをいう。

・ 最初のデータから順番に一つずつ調べていく探索の方法を（ ）という。

例題７ 　**線形探索で在庫数を調べるプログラム**

右の図のような商品番号，商品名，在庫数のデータを格納した二次元配列（リスト）がある。商品番号を，線形探索のアルゴリズムを用いてリストから探索し，商品名と在庫数を表示するプログラムを作成してみよう。

stock = [[1001, '飲料水A', 87],

[1003, '飲料水B', 89],

[1004, '飲料水C', 52],

[1007, '飲料水D', 120],

[1011, '飲料水E', 119],

[2001, '菓子パンA', 67]]

教科書p.110の解答例のプログラムを実行し，商品番号1007を入力したとき，変数など変化は次のようになる。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **行番号** | **i** | **n** | **stock[i][0]** | **処理** |
| 2 | ― | 6 | ― | ― |
| 3 | ( 　) | 6 | 1001 | iに初期値( 　)を代入する |
| 4 | ( 　) | 6 | 1001 | 商品番号が1007と等しくない |
| 3 | ( 　) | 6 | ( 　 ) | iを1増やす。nより小さいので繰り返しを継続 |
| 4 | ( 　) | 6 | ( 　) | 商品番号が1007と等しくない |
| 3 | ( 　) | 6 | ( 　 ) | iを1増やす。nより小さいので繰り返しを継続 |
| 4 | ( 　) | 6 | ( 　 ) | 商品番号が1007と等しくない |
| 3 | ( 　) | 6 | ( 　 ) | iを1増やす。nより小さいので繰り返しを継続 |
| 4 | ( 　) | 6 | ( 　) | 商品番号が1007と等しい |
| 5 | ( 　) | 6 | ( 　) | 結果を表示：（ 　 　） |
| 6 | 3 | 6 | 1007 | 探索終了 |

**目標に対する達成度：** よくできた ・ できた ・ ある程度できた ・ できなかった

**学習の振り返り**

データの探索を行うためにはどのようなアルゴリズムを用いればよいか，理解できたことを記述しよう。

p.110～111 4章2節 プログラミングの実践

**\_\_\_\_\_\_\_\_\_\_年 \_\_\_\_\_\_\_\_\_\_組 \_\_\_\_\_\_\_\_\_\_番**

**名前 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**３ 探索と整列のプログラム②**

データの探索や並べ替えを行うためには，どのようなアルゴリズムを用いればよいだろうか。また，アルゴリズムによって効率に差はあるだろうか。

**２ 二分探索のアルゴリズム**

・ ( 　　　　　 )とは，データを（ 　　 　）に分けてどちらに探索するデータが含まれているか調べ，含まれているほうのデータをさらに（ 　　 　）に分ける，という手順を繰り返すアルゴリズム。

あらかじめ（ 　　 　）されているデータに対して用いることができる。

例題８ 　**二分探索で在庫数を調べるプログラム**

例題7の商品番号2001を二分探索で探索するプログラムを作成してみよう。

教科書p.111の解答例のプログラムを実行し，商品番号2001を入力したとき，変数などの変化は次のようになる。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **行番号** | **imin** | **imax** | **imid** | **stock[imid,0]** | **処理** |
| 2 | 0 | 5 | ― | ― | ― |
| 3 | 0 | 5 | ― | ― | ― |
| 4 | 0 | 5 |  |  | imin ≦ imaxより繰り返しを継続 |
| 5 | 0 | 5 | ( 　 ) | 1004 | imidに値を代入 |
| 6 | 0 | 5 | ( 　 ) | 1004 | 2001と等しくない |
| 10 | 0 | 5 | ( 　 ) | 1004 | stock[imid,0]は2001より小さい |
| 13 | ( 　 ) | 5 | ( 　 ) | 1004 | iminに値を代入 |
| 4 | ( 　 ) | 5 | ( 　 ) | 1004 | imin ≦ imaxより繰り返しを継続 |
| 5 | ( 　 ) | 5 | ( 　 ) | ( 　) | imidに値を代入 |
| 6 | ( 　 ) | 5 | ( 　 ) | ( 　) | 2001と等しくない |
| 10 | ( 　 ) | 5 | ( 　 ) | ( 　) | stock[imid,0]は2001より小さい |
| 13 | ( 　 ) | 5 | ( 　 ) | ( 　) | iminに値を代入 |
| 4 | ( 　 ) | 5 | ( 　 ) | ( 　) | imin ≦ imaxより繰り返しを継続 |
| 5 | ( 　 ) | 5 | ( 　 ) | ( 　) | imidに値を代入 |
| 6 | ( 　 ) | 5 | ( 　 ) | ( 　) | 2001と等しい |
| 7 | ( 　 ) | 5 | ( 　 ) | ( 　) | 結果を表示：（ 　　） |
| 8 | ( 　 ) | 5 | ( 　 ) | 2001 | 探索終了 |

**３ アルゴリズムの評価**

・ アルゴリズムの実行時間の目安になる量を( 　　　　 　)という。一般的には，計算量が少ないアルゴリズムが優れているといえる。

・ 線形探索は，データが2倍になると，計算量は（ 　　 ）になる。

・ 二分探索は，データが2倍になると，計算量は（ 　　　　　　　　 ）しか増えないことが予想される。

**目標に対する達成度：** よくできた ・ できた ・ ある程度できた ・ できなかった

**学習の振り返り**

データの探索を行うためにはどのようなアルゴリズムを用いればよいか，理解できたことを記述しよう。

また，アルゴリズムによる効率の差について記述しよう。

p.112～113 4章2節 プログラミングの実践

**\_\_\_\_\_\_\_\_\_\_年 \_\_\_\_\_\_\_\_\_\_組 \_\_\_\_\_\_\_\_\_\_番**

**名前 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**３ 探索と整列のプログラム③**

データの探索や並べ替えを行うためには，どのようなアルゴリズムを用いればよいだろうか。また，アルゴリズムによって効率に差はあるだろうか。

**４ 整列のアルゴリズム**

・ データをある値に基づいて並べ替えることを( 　　　 )という。ここでは

・ ここで学習するアルゴリズムは（ 　　　　　　 ）という。

例題９ 　**バブルソートを実行するプログラム**

左の図のようなバラバラの順でリストに格納された在庫データを，バブルソートのアルゴリズムを用いて，商品番号順に整列させるプログラムを作成してみよう。また，計算量を求めてみよう。

stock = [[2006, '菓子パンA', 67],

[2009, '菓子パンD', 82],

[2010, '菓子パンE', 82],

[1003, '飲料水C', 52]]

教科書p.11３のプログラムを実行したとき，変数などの変化は次のようになる。

| **行番号** | **Comp** | **imax** | **i** | **stock[i][0]** | **stock[i+1][0]** | **処理** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | ( ) | ― | ― | ― | ― | compへ代入 |
| 2 | 0 | ― | ― | ― | ― | start\_indへ代入 |
| 3 | 0 | ― | ― | ― | ― | last\_indへ代入 |
| 4 | 0 | ( ) | ― | ― | ― | imaxへ初期値代入 |
| 5 | 0 | 2 | 0 | ( 　) | ( 　 ) | iへ初期値代入 |
| 6 | ( ) | 2 | 0 | ( 　) | ( 　 ) | compに1加算 |
| 7 | 1 | 2 | 0 | ( 　) | ( 　 ) | stock[i][0]がstock[i+1][0]より小さい |
| 5 | 1 | 2 | 1 | ( 　) | ( 　) | iに1加算。i < imax + 1より繰り返し継続 |
| 6 | ( ) | 2 | 1 | ( 　) | ( 　) | compに1加算 |
| 7 | 2 | 2 | 1 | ( 　) | ( 　) | stock[i][0]がstock[i+1][0]より小さい |
| 5 | 2 | 2 | 2 | ( 　) | ( 　) | iに1加算。i < imax + 1より繰り返し継続 |
| 6 | ( ) | 2 | 2 | ( 　) | ( 　) | compに1加算 |
| 7 | 3 | 2 | 2 | ( 　) | ( 　) | stock[i][0]がstock[i+1][0]より大きい |
| 8〜16 | 3 | 2 | 2 | ( 　 ) | ( 　) | 入れ替え処理 |
| 5 | 3 | 2 | 3 | ( 　 ) | ( 　) | iに1加算。i = imax + 1より繰り返し終了 |
| 4 | 3 | ( ) | 3 | ( 　 ) | ( 　) | imaxから1減算。imax > start\_ind-1より繰り返し継続 |
| 5 | 3 | 1 | 0 | ( 　) | ( 　) | iへ初期値代入 |
| 6 | ( ) | 1 | 0 | ( 　) | ( 　) | compに1加算 |
| 7 | 4 | 1 | 0 | ( 　) | ( 　) | stock[i][0]がstock[i+1][0]より小さい |
| 5 | 4 | 1 | 1 | ( 　) | ( 　 ) | iに1加算。i < imax + 1より繰り返し継続 |
| 6 | ( ) | 1 | 1 | ( 　) | ( 　 ) | compに1加算 |
| 7 | 5 | 1 | 1 | ( 　) | ( 　 ) | stock[i][0]がstock[i+1][0]より大きい |
| 8〜16 | 5 | 1 | 1 | ( 　 ) | ( 　) | 入れ替え処理 |
| 5 | 5 | 1 | 2 | ( 　 ) | ( 　) | iに1加算。i = imax + 1より繰り返し終了 |
| 4 | 5 | ( ) | 2 | ( 　 ) | ( 　) | imaxから1減算。imax > start\_ind-1より繰り返し継続 |
| 5 | 5 | 0 | 0 | ( 　) | ( 　) | iへ初期値代入 |
| 6 | ( ) | 0 | 0 | ( 　) | ( 　) | compに1加算 |
| 7 | 6 | 0 | 0 | ( 　) | ( 　) | stock[i][0]がstock[i+1][0]より大きい |
| 8 | 6 | 0 | 0 | ( 　) | ( 　) | 入れ替え処理 |
| 5 | 6 | 0 | 0 | ( 　) | ( 　) | iに+1するとi = imax + 1より繰り返し終了 |
| 4 | 6 | 0 | 0 | ( 　) | ( 　) | imaxから-1するとimax = start\_ind-1より繰り返し終了 |
| 17,18 | 6 | 0 | 3 | ( 　) | ( 　) | 実行結果表示: [1003, '飲料水C', 52]  [2006, '菓子パンA', 67]  [2009, '菓子パンD', 82]  [2010, '菓子パンE', 82] |
| 19 | 6 | 0 | 3 | ( 　) | ( 　) | 計算量を表示: 6 |

確認問題5

（1） 例題9で，データ数を2倍，4倍に増やしたとき，計算量がどのように増加するか調べてみよう。ただし，ここでは商品番号の大小を比較する回数を計算量とする。

|  |  |  |  |
| --- | --- | --- | --- |
| データ数 | 例題９：4件 | ２倍（8件） | ４倍（16件） |
| 計算量 | 6 | （ ） | （ ） |

（2） 例題9のプログラムを修正し，商品番号の降順に並べ替えてみよう。

修正箇所(行番号): 7

修正内容： if stock[i][0] < stock[i + 1][0]:

**目標に対する達成度：** よくできた ・ できた ・ ある程度できた ・ できなかった

**学習の振り返り**

データの整列を行うためにはどのようなアルゴリズムを用いればよいか，理解できたことを記述しよう。