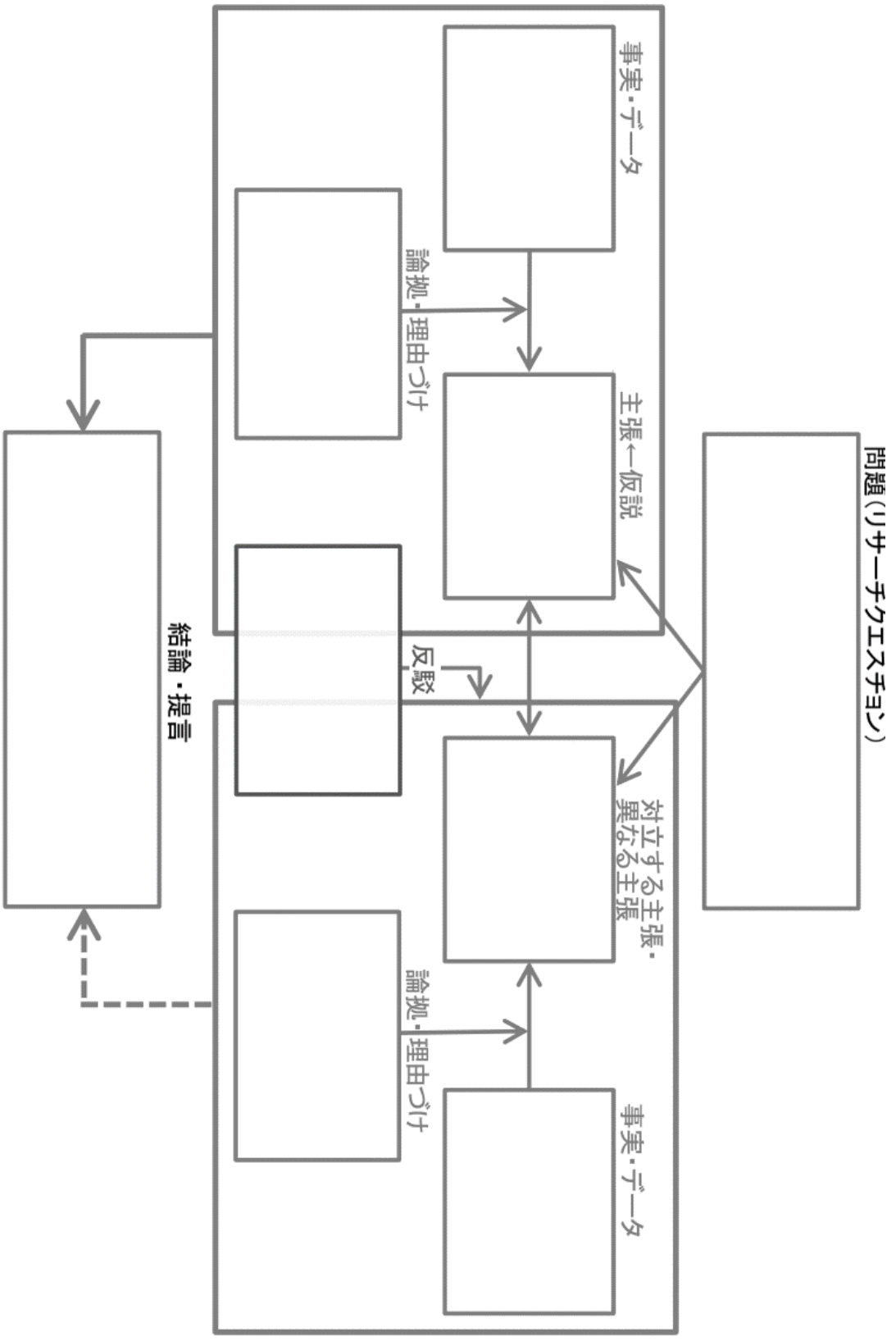
4⑸レポートにまとめよう

以下の論証モデルを使用し，研究内容が論理的なものになるように整理しよう。その上で，次ページの「論証モデルの使い方およびそれを活かした論文作成方法例」を参考にして，レポートを作成しよう。

<u>レポートの様式</u>

日本語，Google ドキュメント（２ページ程度）のデータで提出。ペアと連名で作成すること。

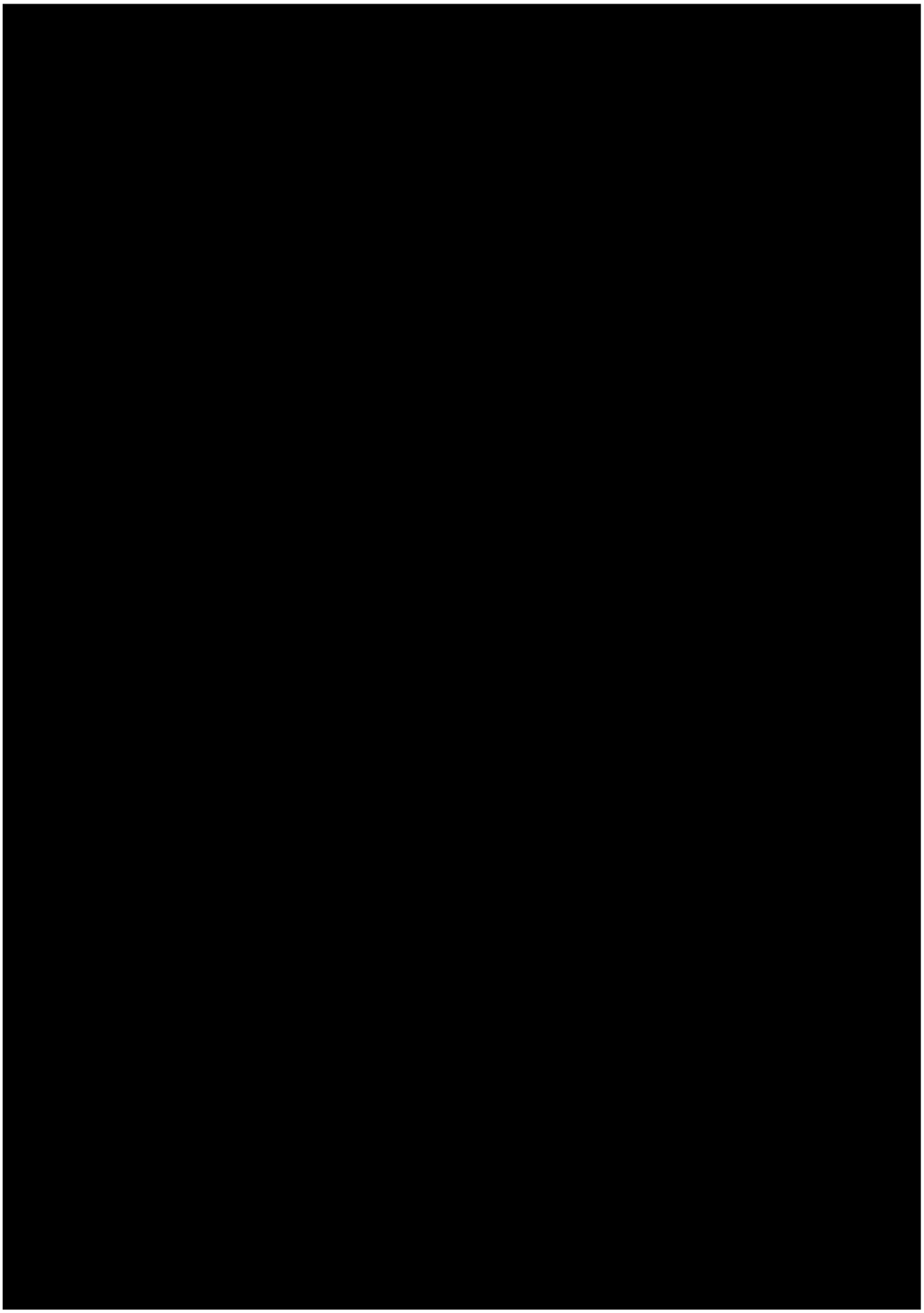


各世代が安心して運転するにはどうすればいいのか？
―「条件付き運転免許返納」の提案―

作成するレポートの例
文章の構成や表現方法などを参考にして作成すること。ただし、実際にレポートを作成する際は，論証モデルは掲載しなくてよい。

1年●組●●番
●● ●●

[redacted]



6　Science English

実施日　　月　　日（　　）

6⑴スライドを作成しよう

情報Ⅰとの関連

Q：受け手に情報をわかりやすく効率的に伝えるためにはどうすればよいか？

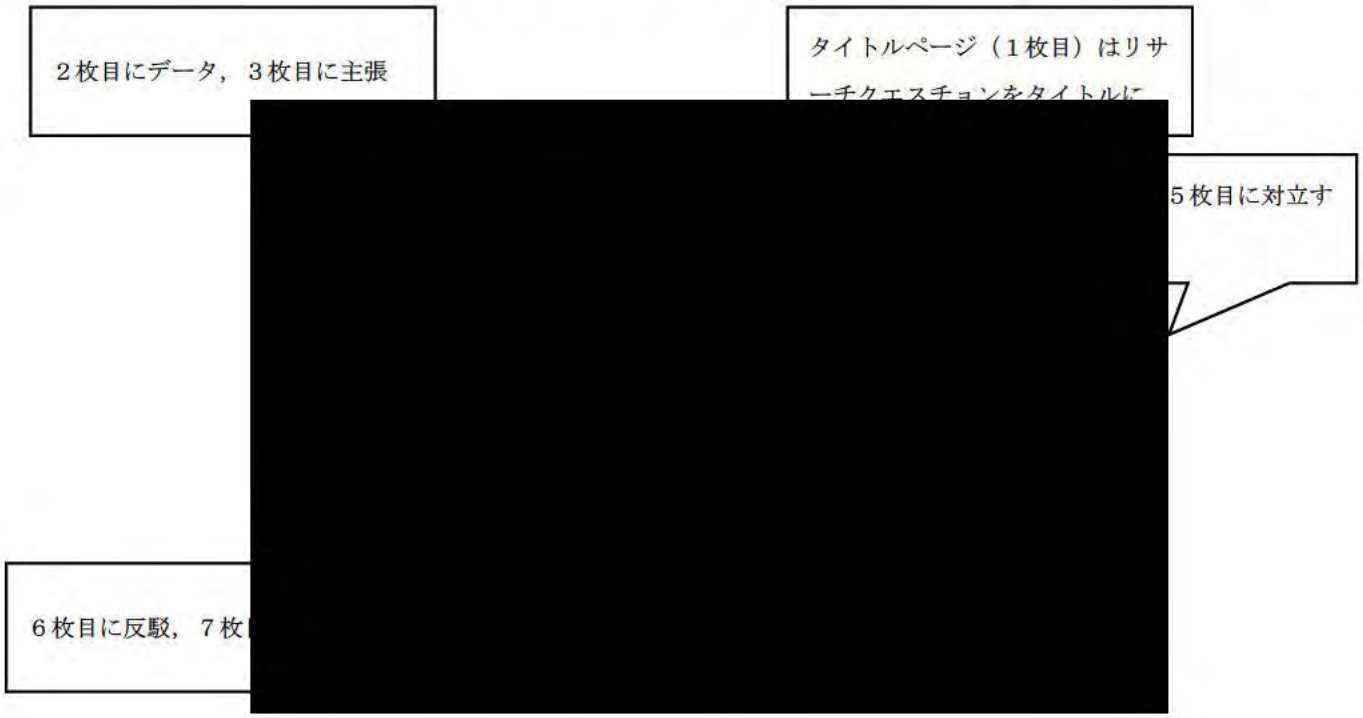
Q：よりよいスライド（情報デザイン）を作成するにはどのような工夫をすればよいか？

GOAL→英語でのプレゼンテーション（1月）

英語でスライドを作成し，英語でプレゼンテーションをします。聴衆にその発表を聞いてもらい，質問をしたり，感想を伝えたり…など，英語でやりとりをします。過去3年間は大阪大学に在学中の留学生（中国・インド・フィリピン・バングラデシュ・アルゼンチン・アルジェリア・インドネシア・フランスなど）に参加してもらいました。各クラスに8～10名の留学生が参加しました。

【作成編】

対話型論証モデルに沿って，説明を展開していきましょう。基本的にスライドは7枚です。



・グラフが小さくて見えない場合は，1枚に詰め込まず複数ページになっても構いません。グラフが見えないと主張の根拠として成り立ちません。

⑵プレゼンテーション練習

【基本編】

□ オーディエンスの方を向いて話す　□ 大きな声で話す　□ 正しい発音で話す

□ ハキハキと明るいトーンで話す　□ 読み方に自信がない単語がないようにしておく

□ 文法の大きなミスなく話す　　□ 内容が頭に入っている状態で話す

□ 難しい単語を使いすぎないようにする

□ 落ち着いて話す（声の高さ，スピード，動き）ジェスチャーは量より質が大事。

□ スピード感を大事にする。聞き手の集中は思っているよりも持たない。

□ 挨拶で始まり，感謝で終える。（Hello や Hi / Thank you for listening.など）

★さらに良くするには何ができる？

伝えたいところを強調するためにその部分を

□ 大きな声で言う　□ ゆっくり言う　□ 繰り返し伝える

聴衆を巻き込むために

□ 質問をする　　□ アクティビティに参加してもらう

聴衆の注目を集めるために

□ 写真を使う　　□ 動きを大きくする　　□ アイコンタクトをとる

□ 語と語の間に間をとる

言葉に関してできることは

□ 一文目を工夫する　　□ 比較表現を使う　□ 印象に残る言葉を使う

【実践編】

作成したポスターに沿って，英語でプレゼンテーションをする練習をしましょう。

① Introduction

- 簡単な自己紹介もできるとよい。
- 聴衆に事前に説明しておくべきことはないか？（日本特有の文化など）

② Data

- データの出典はどこなのか説明する。
- グラフや表などの英語例　表（table），棒グラフ（bar graph），円グラフ（pie graph）折れ線グラフ（line graph），　散布図（scatter plot）
- データの大小のみでなく，差を表す表現を使おう。
  - 倍数表現　　twice[four times / one-third / half] as ～ as …
  - 差の表現　　比較級 ＋by 差
  - 差の強調表現　　much, far （最上級の強調は by far）

③ Conclusion

- Data を根拠に，自分の意見・考えを述べる。（I think …）
- 結論に不備がある場合は，その理由も考えてみましょう。

④ 感謝で終わる。（Thank you for listening.）　質疑応答へ移る。

さぁ，練習してみましょう！　意識すべきは，次の５か所！

eyes ・ mouth ・ shoulders ・ hands ・ feet

を開いて話をしよう！

⑶質疑応答（Q＆A session）の練習

質問は何のために行うのでしょうか?

⇒ ① 発表者とのコミュニケーション - 話を聞いていたよ！面白い話をありがとう！

② 研究のお手伝い - 研究を次につなげていくための素材提供

★質疑応答の表現集：練習しておこう！

○ **質問を受け付ける**

Does anyone have any questions or comments? （質問やコメントはありますか。）

If anyone has any questions, I'd be pleased to answer them.

（質問があれば，喜んでお答えします。）

○ **質問をする**

I have a question. / Let me ask you a question. （質問があります。）

My question is about ～. （私の質問は～についてです。）

○ **繰り返しを求める**

Excuse me, could you repeat that please? （もう一度繰り返していただけますか。）

Could you say that again? （もう一度言っていただけますか。）

○ **質問内容を明確にする**

Do you mean ～? （～ということですか。）

What do you mean by ～? （～はどういう意味ですか。）

Are you asking whether ～? （～かどうかという質問でしょうか。）

Could you explain it in more detail? （もう少し詳しく説明していただけますか。）

○ **答えを確認する**

Did I answer your question? （答えになったでしょうか。）

I hope that answered your question. （これでお答えできたかと思いますが。）

○ **答えがすぐに出ない場合に返答を保留する**

Let me think about it for a minute. （少し考えさせてください。）

That's a good question. I'll think it over. （いい質問ですね。考えておきます。）

★質問をメモしておこう！

研究の弱点や，グラフの触れなかった部分について聞かれることが多い。ある程度想定もできるはずです。練習で聞かれた質問をペアと共有し，メモしておきましょう。

質問①

→答え

質問②

→答え

質問③

→答え

5⑷発表会当日用メモ　１月上旬@会議室

７章　プログラミング

天王寺高校の情報Ⅰでは，最終成果物としてアプリを開発する。ここではそのために基本的なプログラミングの文法と，アプリの画面開発のための HTML と CSS を学習しよう。

１．プログラミングの基本

1-1　出力

print( )：画面に( )の中身を出力する関数

プログラムは、基本的に
上から順番に実行される！

例題１　次のプログラムを実行し、動作を確認せよ。

```
print( 'Hello' )
print( 'World!' )
```

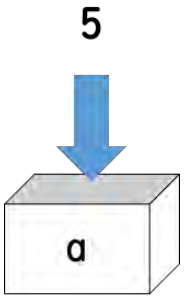
【実行結果】　Hello
　　World!

print( )の( )の中身は、文字列なら「' '」（シングルクォーテーション）か「" "」（ダブルクオーテーション）で挟む。

練習１　print()を３回使って、「おはよう、〇〇さん」「こんにちは、〇〇さん」「こんばんは、〇〇さん」を３行で表示するプログラムを作成せよ。ただし、〇〇には自分の名前を入れること。

1-2　変数

変数とは　・値や文字列を保管する箱のようなもの
・a＝５　で**a という変数に５を代入する**という意味で
　a に５が保存される。
・変数は、値を書き込んだり、更新することもできる。
・a の部分(変数の名前)はある程度制限があるが、
　好きな（わかりやすい）名前をつけることができる。



例題２　次のプログラムを実行し、動作を確認せよ。

```
a = 'Hello'
print(a)
```

【実行結果】　Hello

上のプログラムのように、print(a)という書き方で、
a の中に保存されている（代入されている）数値や文字列を表示することができる。

練習２　変数 aisatu に「おはよう」、変数 namae にあなたの名前を代入したうえで、「おはよう〇〇」と表示させよ。

【ヒント】print(aisatu, namae)のように、print では( )の中を「,」でつなぐと、２つを連結させて出力することができる。

1-3　四則演算

プログラム中では、四則演算を行うことができる。演算子を次の表に示す。

	数学の表記	演算子
足し算	+	+
引き算	−	−
掛け算	×	*
割り算	÷	/

	演算子	例	結果
商	//	9//4	2
余り	%	9%4	1
累乗	**	3**4	81

例題 3　次のプログラムを実行し、動作を確認せよ。

```
a = 2
b = 3
c = a + b
print(c)
print(a + b)
print(2 + 3)
```

【実行結果】　5
5
5

どの形で計算しても、答えは同じになる。しかし、2 と 3 を使って、複数の計算をするときは、変数を用いるほうが好ましい。練習 3 で check!

練習 3　8-3、8÷3 の答え、8 ÷3 の余りをそれぞれ計算し、出力せよ。その後、8 を 15 に変えて再度プログラムを実行せよ。その際、変更箇所ができるだけ少なくなるように組み立てること。

1-4　入力

input( )：ユーザーからの入力を受け付ける関数。( )の中に文字列を入れることで、入力の際に表示させる文字列を指定することができる。

例題 4　次のプログラムを実行し、動作を確認せよ。

```
a = input('名前は?→')
print(a, 'です')
```

【実行結果】　名前は？→天高太郎
天高太郎 です

練習 4　input( )を用いて、任意の名前を入力し、出力として「こんにちは　○○さん」と表示するプログラムを作成せよ。

1-5　数値への変換

int( )：（)の中の文字列を、整数値に変換する関数

例題５　次の２つのプログラムを実行し、動作を確認せよ。

```
a = input(‘１つめは？→‘)
b = input(‘２つめは？→‘)
c = a + b
print(c)
```

【実行結果】１つめは？→１
２つめは？→２
１２

```
a = int(input(‘１つめは？→‘))
b = int(input(‘２つめは？→‘))
c = a + b
print(c)
```

【実行結果】１つめは？→１
２つめは？→２
３

どちらのプログラムも、上の例では a に１を、 b に２を代入しているが、左のプログラムでは１と２がそれぞれ文字列として認識されており、計算が出来ずそのまま１と２を繋げて出力している。
右のプログラムでは、１と２が int によって数値に変換されているので、１＋２で３の答えを出力している。

練習５　input( )、int( )、print( )を使って、整数値を入力することで、立方体の体積を出力するプログラムを作成せよ。入力時の文章は「１辺の長さを入力してください」、出力は「体積は～cm^3 である」と出力せよ。

1-6　データ型

値にはそれぞれデータ型というものがある。例題５では int( )を用いることで文字列を数値（正確には整数）に変換し、変換させることで計算ができるようになっている。同じ「１」という値でも計算ができる数値として扱うのか,文字列として扱うのかによって結果が変わる。

このようにデータの種類や扱い方を決めるのがデータ型と呼ばれるものである。例題５において input( )によって入力されたデータは文字列型といわれるもので、int( )で整数型に変換させている。他にも小数型に変換する float( )や、文字列型に変換する str( )なども存在する。基本的なデータ型は、どのプログラミング言語でも使われているので、これが意識できるようになれば、様々なバグやエラーにも対処できるようになるかもしれません。

練習６　次のプログラムを実行すればエラーとなる。原因を考察せよ。
【hint】「＋」の記号の役割は？

```
1 genzai = int(input('年齢を入力してください'))
2 mirai = genzai + 10
3 print('私の10年後の年齢は',mirai,'歳です')
4 print('私の10年後の年齢は' + mirai + '歳です')
```

2.条件分岐

2-1　基本的な if 文の使い方

```
if 論理式:
    論理式が真のときに実行
else:
    論理式が偽のときに実行
```

論理式では比較演算子を用いる。
if 文では、文頭を半角スペース４つ分さげる。

例題１ 次のプログラムを実行し、動作を確認せよ。

```
a = int(input('0 か 1 を入力→' ))
if a == 0:
    print('0 が入力されました')
else:
    print('1 が入力されました')
```

【実行結果①】
0 か 1 を入力→ 0
0 が入力されました

【実行結果②】
0 か 1 を入力→ 1
1 が入力されました

比較演算子	意味
A ＜ B	A は B より小さい
A ＜＝ B	A は B 以下
A ＞ B	A は B より大きい
A ＞＝ B	A は B 以上
A ＝＝ B	A と B は等しい
A ！＝ B	A と B は等しくない

if の後ろの a == 0 が論理式である。この式が満たされているときに、次の行が実行されている。論理式に用いる比較演算子は上の表を参考にすること。else の部分は省略することも可能である。このプログラムでは、0 か 1 以外の入力をした場合も“1 が入力されました”と出力されてしまうので、次の例題 2 で拡張する。

練習１　入力した整数が 50 以上の場合は「50 以上」、50 未満の場合は「50 未満」と表示するプログラムを作成せよ。

2-2　if 文の拡張

```
if 論理式①:
    論理式①が真のときに実行
elif 論理式②:
    論理式②が真のときに実行
else:
    全ての論理式が偽のときに実行
```

1 つめの条件を満たさないときに、更に条件を追加したい場合は elif を使う。

elif は 1 つだけの制限は無いので、条件を増やす際には elif を増やすこともできる。

例題 2　例題 2-1 を改良した次のプログラムを実行し、動作を確認せよ。

```
a = int(input('0 か 1 を入力してください→'))
if a == 0:
    print('0 が入力されました')
elif a == 1:
    print('1 が入力されました')
else:
    print('0 か 1 以外が入力されました')
```

【実行結果①】

0 か 1 を入力してください→ 1
1 が入力されました

【実行結果②】

0 か 1 を入力してください→ 2
0 か 1 以外が入力されました

練習 2　入力した整数が正の数であるとき「Positive」、0 であるとき「Zero」、負の数であるとき「Negative」と出力せよ。

練習 3　3 つの整数を入力し、最大の数を出力するプログラムを作成せよ。
（ヒント　if 文の中に更に if 文を組み込むことができる）

2-3　python における表記の仕方について

本書の例題においては、例えば a = 2 というコードを入れるとき、a=2 のようにスペースを入れて表記している。実際に打ち込むときはスペースがなくても、プログラムは実行されるが、=の前後には半角スペースを入れることを推奨する。プログラムは他者が読む可能性もあるので、書き方の一般的なルールのようなものが設定されている（python では PEP8 というものがある）。

また、if 文では、論理式が真や偽であった場合に実行するプログラムは、**文頭から半角スペース 4 つ分さげて書き始める**が、python ではこの文頭下げには重要な意味がある。文頭を下げることによって、その下がったコードをブロック、ひとかたまりとしてみなしてくれ、条件によって複数の処理を行うことができる。

```
a = int(input('0 か1を代入'))
if a == 0:
    b = 3
    print('b に3を代入しました')
else:
    b = 4
    print('b に4を代入しました')
print(b)
```

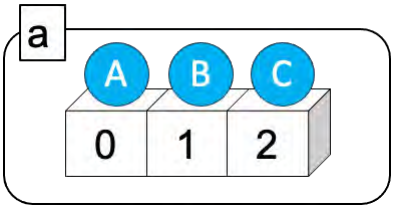
a が 0 なら、文頭を下げたこのブロックの処理を行う

3. 繰り返し

3-1　リスト

リストとは　・一つの変数に複数の値を代入（保存することができる）
　　・イメージは変数の拡張版

〈書き方〉　配列名 ＝ [要素 1, 要素 2, …]
（例）　a ＝ [‘A’, ‘B’, ‘C’]
　　b ＝ [1, 5, 6, 8]



リストの中身（要素）を指定するときは、a[番号]で取り出すことができる。
1 番めが 0、２番めが 1 …という風に番号（添え字、インデックス）が指定されている。

例題 1　次のプログラムを実行し、リストの使い方の基本を確認せよ。

```
a = [‘A’, ‘B’, ‘C’ ]
print(a)
print(a[1])
a[1] = ‘D’
print(a)
```

【実行結果】[‘A’, ‘B’, ‘C’]
　　B
　　[‘A’, ‘D’, ‘C’]

print(a)でリスト全体の出力、
print(a[1])でリスト a の 1 に指定されている要素を出力することができる。また、変数と同じように、再代入することも可能。

3-2　リストの扱い方

リストに入っている要素を操作する機能は複数ある。

①配列名.append(要素)　　配列の要素の最後に、(　)内の要素を付け足す。
　例：a ＝ [1, 2, 3]のとき、 a.append(4)　→　a ＝ [1, 2, 3, 4]

②配列名.insert(インデックス, 要素)　インデックスで指定した番号の前に、要素を追加。
　例：a ＝ [1, 2, 3]のとき、a.insert(2, 0)　→ a ＝ [1, 2, 0 ,3]

③配列名.pop(インデックス)　　インデックスで指定した番号の要素を取り出し、配列から取り除く。
　例：a ＝ [1, 2, 3]のとき、a.pop(1)　→ a ＝ [1, 3]

その他にも操作する機能があるので、必要に応じて調べると良い。

3-3　決まった回数の繰り返し

for 変数 in リスト: 　処理１ 　処理２ 　… 処理Ａ	リストから値を１つずつ取り出し、変数に代入。処理１、２…の実行後、リストの次の値を取り出し、再度変数に代入、実行。すべての値について処理が終われば処理Ａへ進み、繰り返しが終了する。

例題２　次のプログラムを実行し、動作を確認せよ。

```
for i in range(5):
    print(i)
print('finish')
```

※文頭が下がっているところは
半角スペース４つ分(インデント)

【実行結果】
0
1
2
3
4
finish

range(5)は[0, 1, 2, 3, 4]のリストのことである。
変数 i に対して、０から順に代入していき、４まで代入した後ループが終わり、finish が出力される。

range(数値)：数値に合わせて、
　　リストと同じように使えるデータを作成する関数
range(△)　　→0 から△未満までの整数の範囲
range(☆, △)　　→☆から△未満までの整数の範囲
range(☆, △, □)　→☆から△未満までの整数で□ずつ増える範囲

例題３　次のプログラムを実行し、range の動作を確認せよ。

```
for i in range(1, 5):
    print(i)
```

【実行結果】

```
for i in range(1, 10, 2):
    print(i)
```

【実行結果】

練習１　for 文を使って、11 から 20 までの整数に対して、それぞれ２倍した値を出力するプログラムを作成せよ。

3-4　条件を満たすまでの繰り返し

```
while 式:
    処理１
    処理２
    …
処理Ａ
```

式を実行・評価して、真(true)なら処理１、２…に進む。その後再び式を実行・評価し、真なら再び処理１…へ、偽(false)なら処理Ａへ進み、繰り返し終了。

例題４　次のプログラムを実行し、動作を確認せよ。

```
a＝0
while a＜5:
    a＝a＋１
    print(a)
print('finish')
```

【実行結果】
１
２
３
４
５
finish

while のところで a＜５ならインデントのところの処理が実行される。最初は a ＝ 0 で a ＜ 5 に当てはまっているので処理が実行されている。
a ＝ a ＋ １は元々保存されている a の値に １を足した値を新たに a に代入するという式である。最初は a ＝ 0 で 0 に １を足した答えの １を a に代入しているので、その後の print(a)で １が出力されている。
最後 a＝5 になれば、a＜5 に当てはまらないので、finish が出力されている。

＊累乗代入演算子

プログラミングでは、例題４の **a＝a＋１** のような計算はよく用いる。数学的にこの式を見るとおかしく見えるが、例題を実行してもらったように、元の a に入っている数値に対して、１を加えたものを新しく a に代入しなおすコードである。この計算はよく使うので、別の表記の仕方が準備されている。 **a +=１** という式でも同じように実行ができるので、試してみてほしい。足し算以外もあるので、検索してみてほしい。

練習２　２の１乗、２の２乗…と順に出力し、1000 を超えると終了するプログラムを作成せよ。

練習３　１から 10 までの整数を足して、その合計を表示するプログラムを作成せよ。ただし、range()と for を用いること。

練習４　Ａさんのテストの点数は、70 点、50 点、60 点、75 点、45 点であった。これらの点数を用いて score という名前のリストを作成し、合計と平均を出力せよ。

（ヒント）for 文を用いると、score[i]という形でリストの中の要素を取り出すことができる。

（＋α）python で合計や平均を出力できる関数（range( )のようなもの）はないだろうか。

3-5　繰り返しと分岐などの組み合わせ

例題５　次のプログラムを実行し、動作を確認せよ。

```
for i in range(1, 6):
    if 1 % 2 == 0:
        print(i, 'は偶数です')
    else:
        print(i, 'は奇数です')
```

【実行結果】1 は奇数です
2 は偶数です
3 は奇数です
4 は偶数です
5 は奇数です

このプログラムのように、繰り返しの中に、if を入れて組み合わせて使うこともできる。また繰り返しのなかに別の繰り返しをさせることも可能である。

練習５　1 から 30 までの間の、３の倍数だけ表示するプログラムを作成せよ。

練習６　1 から任意の整数（読み込ませる）において、1 から順に出力し、3 の倍数なら「Fizz」、５の倍数なら「Buzz」、３と５の両方の倍数なら「FizzBuzz」とするプログラムを作成せよ。
（例）1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz ...　（実際は縦に出力させる。）

練習７　九九の答えを 1×1 から９×９まで出力するプログラムを作成せよ。

3-6　繰り返しを止めるプログラム

任意のタイミングで繰り返しを止めたいときには、break や continue といったコードを入れると制御することができる。それぞれ制御の仕方が異なるので、例題５で動作を見ながら確認してほしい。

例題５　次のプログラムを実行し、動作を確認せよ。

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

【実行結果】
0
1
2

```
for i in range(5):
    if i == 3:
        continue
    print(i)
```

【実行結果】
0
1
2
4
5

○ break　　→ その時点で繰り返しを終了（ループから抜け出す）
○ continue → その時点で繰り返しの次のステップへ進む

練習８　任意の整数が素数であるかどうかを判別するプログラムを作成せよ。

4. 関数

4-1　組み込み関数

関数とは　・定められた一連の処理を実行する仕組み

例　int( )　→　( )の中身を整数に変換する処理を行う

・プログラミング言語などに予め用意された関数　→　**組み込み関数**

例　print( ), input( )

・関数に読み込ませる値(かっこの中の値)を**引数（ひきすう）**、関数で処理をした結果の値を**戻り値**という。

・関数は組み込み関数以外にも、モジュールやライブラリと呼ばれる別のファイルにある関数や、自作で作る関数も存在している。

4-2　モジュールやライブラリの関数

import モジュール名：モジュールを読み込んで、関数を使えるようにする
random.randint(△, ☆)：標準モジュールの random の中に含まれる関数。 △、☆には整数を入れ、△から☆の範囲でランダムな整数を出力する。

例題１　次の乱数を生成するプログラムを実行し、動作を確認せよ。

```
import random
a = random.randint(1,6)
print(a)
```

【実行結果】 3

random.randint(1, 6)は
１から６の範囲でランダムな整数を出力してくれる。

練習１　１から１００のランダムに整数を生成し、それが偶数なら「(ランダムな整数）は偶数です」、奇数なら「(ランダムな整数）は奇数です」と出力せよ。例えば、ランダムな整数が３２なら、出力は「32 は偶数です」となるようにコーディングすること。
(ヒント　偶数･奇数は２で割ったときの余りで判断できる。余りを求める演算子は%。)

python には標準モジュールがいくつかあり、数学に関する math、日付や時間を扱う datetime などがある。また、モジュールは誰でも作成することができるので、世界中で様々な分野に特化した専用のモジュールが開発され公開されている。例えば、python は機械学習などでも使われているので、高度な数値計算ができる Numpy、グラフ作成することができる Matplotlib などがある。実際に次のプログラムを実行してみよう。

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-5, 5, 0.1)
y = np.sin(x)
plt.plot(x, y)
plt.show()
```

4-3　ユーザー定義関数

関数は同じ処理や計算を、複数の箇所で行うときなどに便利で、プログラムもわかりやすく書くことが出来る。また同じ処理を複数回記述すれば、修正をするときにも複数回修正する必要があるため面倒である。これらのことから、自分で必要な関数を作って（関数を作ることを定義するという）、利用する（関数を実行することを呼び出すという）ことができる。

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

関数を定義するときは、def から始める。
関数の中に引数を設定することで、メインプログラムから値を引き継ぐことができる。
return は必ずしもなくてもよい。

例題 2　円の面積を出力するプログラムについて、次のプログラムを実行し、確認せよ。

```
def area(r):
    s = 3.14 * r * r
    return s

y = area(10)
print('面積は', y)
```

【実行結果】
面積は 314

def で定義された関数は、実際に呼び出されるまでは実行されない。このプログラムは 5 行目の y=…から始まる。area(10)で 10 という値を、関数の r に代入し、計算を進めていく。計算した結果 s を元のプログラムに戻し出てきた値が y に代入されている。

練習 2　1 から任意の整数までの和を表示するプログラムを作成せよ。ただし、和を計算するプログラムを関数 goukei として作成すること。

練習 3　右のプログラムを関数 goukei2 を作成して簡略化せよ。

```
a = [ 1, 2, 3, 4, 5 ]
b = [ 100, 200, 300 ]
c = [ 5, 10, 20, 40 ]

a_goukei = 0
for i in a:
    a_goukei = a_goukei + i

b_goukei = 0
for i in b:
    b_goukei = b_goukei + i

c_goukei = 0
for i in c:
    c_goukei = c_goukei + i

print(a_goukei)
print(b_goukei)
print(c_goukei)
```

4-4　ローカル変数とグローバル変数

変数は、メインプログラムの中と関数の中では少し挙動が異なっている。メインプログラム中で定義した変数はグローバル変数と呼ばれ、関数の中でも変数の値を用いることができる。関数中で定義した変数はローカル変数と呼ばれ、メインプログラムの中では変数の値を用いることはできない。具体例として、下のプログラム１ではメインプログラムで定義した hensu1 は関数中でも同じ値を用いることができている。プログラム２では、関数中で定義した kotae2 をメインプログラムで出力しようとしているがエラーとなる。この場合は、プログラム３のように global 変数　とコーディングしておくことで、メインプログラムでも同じ値を使えるようになる。ただ、同じ値を使えることが必ずしも良いことではないので、どのプログラムも一長一短である。

```
def kansu1(a):
    kotae1 = a *
hensu1
    return kotae1

hensu1 = 10
```

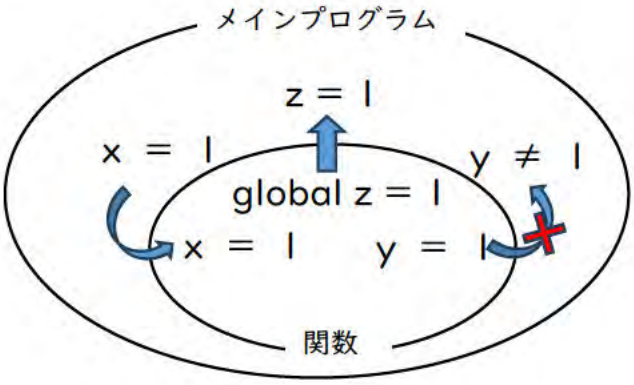
プログラム１

```
def kansu2(a):
    kotae2 = a * 10

kansu2(5)
print(kotae2)
```

プログラム２

```
def kansu3(a):
    global kotae3
    kotae3 = a * 10

kansu3(5)
print(kotae3)
```

プログラム３



5. アルゴリズム

5-1　アルゴリズムの基本

アルゴリズムとは…

問題などを解決するための方法や手順のこと

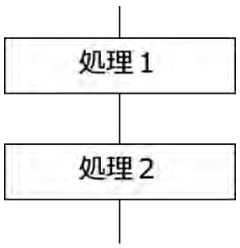
→　視覚的にわかりやすく表現した図

…フローチャート

名称	記号	内容
端子		開始と終了
処理		演算などの処理
判断		条件による分岐
ループ端		繰り返しの開始と終了
データ		データの入力や出力
表示		画面上に表示されるデータなど
線		処理の流れ

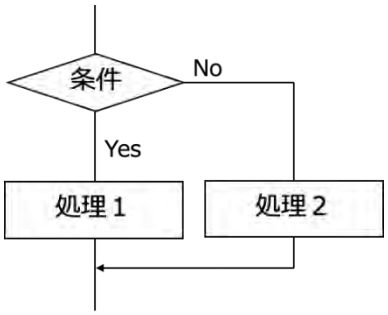
◯プログラムの基本構造

１．順次構造



プログラムの基本中の基本、上から下へ順番に実行される。

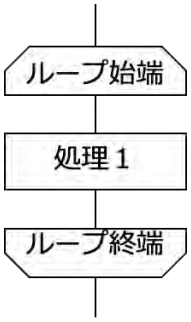
２．選択構造



条件分岐の構造

```
if 論理式:
    処理１
else:
    処理２
```

３．反復構造



繰り返しの構造

```
for 変数 in range:
    処理

while 条件:
    処理
```

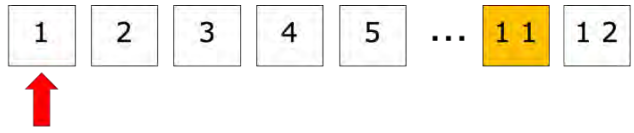
これらを組み合わせることによって、全てのプログラムが完成する。

5-2　探索のプログラム　理論編

・多くのデータから目的のものを見つけること…探索

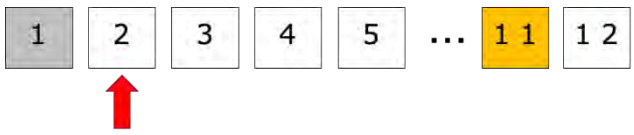
・**線形探索** or **二分探索**を学習する。

◯**線形探索**…端から順に目的の値を探し出す



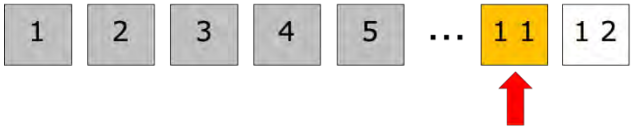
１１を探したいときに、端から順番に１１かどうかを判定し調べていく。

1＝11？



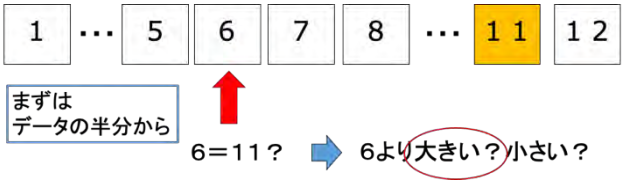
2＝11？

発見！

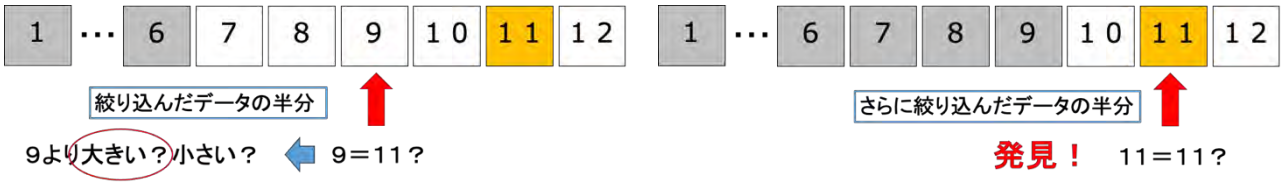


11＝11？

◯**二分探索**…探索範囲を半分に狭めていき、探索値を絞り込む



まずはデータの半分から調べていく。もし見つからなければ、求めているデータが半分の値よりも大きいか、小さいかによってデータを絞り込んでいく。



5-3　探索のプログラム　コード編

例題１　次のプログラムは線形探索を行っているものである。各行でどんな操作・処理をしているか確認せよ。

```
a = [57, 16, 99, 10, 45]:
n = len(a)
print('番号', ' ', 'データ')

for i in range(0, n, 1):
    print(i, '      ',a[i])
print(' ')
s = int(input('探索値の入力→'))

for i in range(0, n, 1):
    if a[i] == s:
        print(s, 'は', i, '番目に存在します')
        break
```

【実行結果】

番号　データ

0　　57

1　　16

2　　99

3　　10

4　　45

探索値の入力→99

99は2番目に存在します

len(a)は配列の長さを出力する関数である。線形探索を行っている部分は下から4行のところである。配列の長さは5、インデックスは4までなので、range(0, 5, 1)で0から4まで指定することが出来ている。要素の1つずつを探索値sと比較し、合致すれば出力の上breakでループを終了している。

例題２　次のプログラムは二分探索を行っているものである。各行でどんな操作・処理をしているか確認せよ。

```
a = [10, 16, 45, 57, 99]:
n = len(a)
print('番号', ' ', 'データ')
for i in range(0, n, 1):
    print(i, '      ',a[i])
print(' ')
s = int(input('探索値の入力→'))
i = 0; j = n – 1
while i <= j:
    m = int((i + j)/2)
    if a[m] == s:
        print(s, 'は', m '番目に存在')
        break
    if a[m] > s:
        j = m – 1
    else*
        i = m + 1
```

【実行結果】

番号　データ

0　　10

1　　16

2　　45

3　　57

4　　99

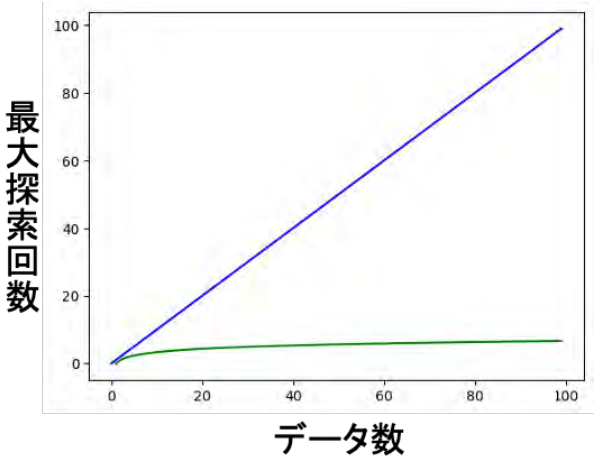
探索値の入力→57

57は3番目に存在します

i が範囲の左端、j が右端を決めている変数である。while 以降が探索をしている部分で、m のところが範囲の中央値を設定している。まずは中央値が探索値と一致しているかを判定し、一致していない場合は大小関係を調べている。もし探索値が中央値より小さい場合は、右端の j を現在の中央値の一つ左に設定、中央値より大きい場合は、左端の i を現在の中央値の一つ右に設定している。

5-4　探索のプログラム　まとめ 編

どちらも目的の値を求めるものであったが、より効率的なものはどちらだろうか。また、全体のデータ数が多くなれば、最大探索回数はどのように変化するだろうか。右のグラフがそれを示したものである。線形探索は端から一つずつ値を比較するため、データ数の影響が大きい。一方二分探索については、データ数が多くなっても、最大探索回数はあまり変化しない。しかし、二分探索を行うためには、予めデータが順に並んでいることが条件であるため、次に進める整列のプログラムが必要になってくる。



5-5　整列のプログラム　理論編

・データを昇順（小さい順）や降順（大きい順）に並べるプログラム
・多くのアルゴリズムがあるが、
　今回は**交換法（バブルソート）**と**選択法（選択ソート）**を学習する

◯**交換法（バブルソート）**…隣同士のデータを比較し、順になるように並び替えていく
昇順に並べていくとすると

[3 1] 5 4 2	3 と 1 を比較
[1 3] 5 4 2	昇順でないので、交換する
1 [3 5] 4 2	3 と 5 を比較
1 3 [5 4] 2	3 と 5 は昇順なのでそのまま、5 と 4 を比較し入れ替える
1 3 4 [5 2]	5 と 2 を比較し入れ替える
1 3 4 2 5	最も右にきた 5 は最大値であり、確定する。以降繰り返し。

○選択法（選択ソート）…最小値や最大値を探し、端から順番に並べていく

[3 1] 5 4 2	3 と 1 を比較→ 1 が現段階の最小値
3 [1 5] 4 2	現段階の最小値 1 と 5 を比較→ 1 が現段階の最小値
3 [1] 5 [4] 2	現段階の最小値 1 と 5 を比較→ 1 が現段階の最小値
3 [1] 5 4 [2]	現段階の最小値 1 と 2 を比較→1 が最小値で確定
[1 3] 5 4 2	最小値 1 を左端に並び替える→左端確定
1 [3 5] 4 2	3 と 5 を比較→ 3 が現段階の最小値、以降最小値探索→ 2 が最小値
1 [2] 5 4 [3]	最小値 2 を左から 2 番めと交換、以降繰り返し

5-6　整列のプログラム　コード編

例題 3　次のプログラムは交換法のプログラムである。各行で行っている操作・処理をしているか確認せよ。

```
a = [5, 3, 8, 2, 9, 1, 4, 6]:
num = len(a)

count = 0

while count < num:
    for i in range(0, num-1):
        if a[i] > a[i+1]:
            temp = a[i]
            a[i] = a[i+1]
            a[i+1] = temp

    count = count + 1
    print(a)
```

【実行結果】

```
[3, 5, 2, 8, 1, 4, 6, 9]
[3, 2, 5, 1, 4, 6, 8, 9]
[2, 3, 1, 4, 5, 6, 8, 9]
[2, 1, 3, 4, 5, 6, 8, 9]
[1, 2, 3, 4, 5, 6, 8, 9]
[1, 2, 3, 4, 5, 6, 8, 9]
[1, 2, 3, 4, 5, 6, 8, 9]
[1, 2, 3, 4, 5, 6, 8, 9]
```

全体として、要素の数だけ左から交換をしていけばよいので、変数 count で回数をカウントしている。for の部分では、隣り合ったデータを比較するループ。temp =…から 3 行は、数値を入れ替えるために行っている。

練習１　例題３のプログラムにおいて、次の箇所を変更すれば、どのような不都合が生じるだろうか。

```
temp = a[i]
a[i] = a[i+1]
a[i+1] = temp
```

```
a[i] = a[i+1]
a[i+1] = a[i]
```

例題４　次のプログラムは選択法のプログラムである。各行で行っている操作・処理をしているか確認せよ。

```
a = [5, 3, 8, 2, 9, 1, 4, 6]
num = len(a)

for i in range(0, num-1):
    min = i

    for j in range(i +1, num):
        if a[min] > a[j]:
            min = j

    a[i], a[min] = a[min], a[i]
    print(a)
```

【実行結果】

```
[1, 3, 8, 2, 9, 5, 4, 6]
[1, 2, 8, 3, 9, 5, 4, 6]
[1, 2, 3, 8, 9, 5, 4, 6]
[1, 2, 3, 4, 9, 5, 8, 6]
[1, 2, 3, 4, 5, 9, 8, 6]
[1, 2, 3, 4, 5, 6, 8, 9]
[1, 2, 3, 4, 5, 6, 8, 9]
```

最初の for のループに関して、選択法は端から１つずつ値を選択し、交換する。また、最後の２つは同時に確定するので、配列の要素数－１回でループは十分である。今回は要素数が８なので、全体は７回のループで終了している。２つめの for のループに関して、最小値が入っている配列のインデックスを指定し、そのインデックスの値と、確定していない要素を順番に比較している。もし、暫定の最小値を下回る数値が出たとき a[min] > a[j]のときは、新たに暫定最小値を min = j で設定している。配列の要素の交換に関しては、python では上のコードのように記述することで交換することができるので、変数 temp を使う必要がない。

練習２　例題３と４のプログラムはそれぞれ昇順に並び替える際のプログラムである。これを降順に並び替えるためにはどこを変更すればよいか。

5-7　整列のプログラム　まとめ編

今回は２つの整列のプログラムを扱ったが、この２つではどちらのほうが効率の良いアルゴリズムだろうか。値を比較する回数はどちらも同じであるが、値を交換する回数は明らかに選択法の方が少ない。今回の２つのアルゴリズムは、数ある整列のプログラムの中でも効率はあまり良くない。その他どのような効率の良いアルゴリズムがあるか、ぜひ調べてみてほしい。

5-8　プログラムの改良例

バブルソートのプログラムの書き方は１つではない。少し改良したものを掲載しておく。任意の整数を並び替えることも出来るし、無駄なループがないように整列が完成した段階で、ループが抜けれるようなフラグを立ててコーディングしている。その他良い効率のよいコードもあるはずなので、プログラムを作成する際は、常に効率の良いコードをめざしてほしい。

```
print('８個の整数を並び替えます')
for i in range(0,8):
    a[i] ＝ int(input(f'{i+1}つめを入力してください　'))
#input のカッコの中にはそのままだと変数が使えない・・・。

print('並び替えを開始します')
print()
print(a,'スタート')

num = len(a)
flag = True

while flag :　　#フラグで判断してみた
    flag = False
#もし入れ替えが生じなければ False のままでループから抜ける

    for i in range(0, num-1):
        if a[i] > a[i+1]:
            a[i], a[i+1] = a[i+1], a[i]
            flag = True
    print(a)
```

【参考】⑹HTML・CSS 基本～アプリ作成のために～

第1章　はじめに

　HTML（HyperText Markup Language）は、Web ページの骨組みを作るための言語である。ほとんどの Web ページが HTML で構築されていて、Web ブラウザはこの HTML を読み取り、ページを表示している。HTML はプログラミング言語とは異なり、Web ページに表示させたいテキストに対して、「タグ」と呼ばれるものをつけていく事によって、テキストの表示のさせ方を指示する。

(例)

```
<!DOCTYPE html>
<html>
<head>
    <title>はじめての HTML</title>
</head>
<body>
    <h1>ようこそ!</h1>
    <p>これは HTML の基本的なページです。</br>
     <span>以下のリンクをクリックしてください</span></p>
    <a href="https://www.example.com">サンプルサイトへ</a>
</body>
</html>
```

＜ ＞で記述されているものがタグ

“ようこそ”が h1 タグに挟まれることによって認識されている。

(表示例)

基本的には**テキストをタグで挟んで記述させる形**で適用する。

基本形：<◯◯>～～～</◯◯>

開始タグ　　終了タグ

　一方 CSS（Cascading Sytle Sheets）は、HTML で書かれたものを装飾する、つまりデザインを司る技術であり、文字や背景の色、画像などの大きさや表示位置を指定するなど、Web ページのスタイルを変更するために利用される言語である。一般的には Web サイト全体のデザインを統一させるために、HTML ファイルとは別のファイルで作成されることが多い。

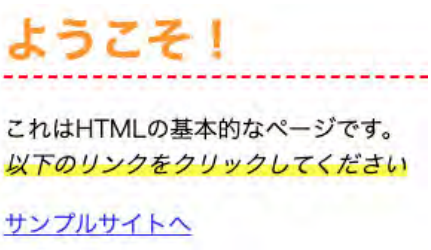
(例)

```
h1{
   color: orange;
   border-bottom: dashed 2px red;
}
span{
   font-style: italic;
   background: linear-gradient(transparent 50%, yellow 50%);
}
```

h1 の箇所についてデザインを指定

span の箇所についてデザインを指定

(表示例)



次のページからは、HTML と CSS の基本的な書き方を掲載するが、すべてを網羅しているわけではないので、必要に応じて都度調べて記述していってほしい。

第2章 HTML

2-1 HTML の基本構造

```
<!DOCTYPE html>
<html>
<head>
    <title>はじめての HTML</title>
</head>
<body>
    <h1>ようこそ！</h1>
</body>
</html>
```

html の基本構造は左のようになっている。実際に公開されている Web ページでも確認してほしい。

・<!DOCTYPE html>：HTML5 であることを宣言

・<html> ～ </html>：HTML 文書全体を囲むタグ

・<head> ～ </head>：Web ページに関係する情報を記述するところ。メタ情報。

・<title> ～ </title>：ページのタイトル

・<body> ～ </body>：ページの本文を記述

2-2 テキストの表示に関連するタグ

○見出し **<hn> ～ </hn> n には 1～6 が当てはまる**

html	表示	説明
** **ようこそ** ** ** **ようこそ** ** ** **ようこそ** ** ** **ようこそ** ** ** **ようこそ** ** ** **ようこそ** **	**ようこそ** **ようこそ** **ようこそ** **ようこそ** **ようこそ** **ようこそ**	Web ページの見出しに使う記述。 数字が大きくなるほど、見出しの文字のサイズが小さくなる。 「h」は heading の頭文字。

○段落 ** ～ ** ・ 改行 **
**

html	表示
ようこそ ** **こんにちは** ** ** **初めての html 基本をしっかり** **学んでいこう。** **	**ようこそ** こんにちは 初めてのhtml 基本をしっかり 学んでいこう。

説明
タグで挟むことによって、挟まれたところを一段落としてグループ化することができる。段落が変われば自動的に改行される。「p」は paragraph の頭文字。 html 上で改行しても(4 行目基本の前)、実際には改行されない。 を使うことで、改行させることができる。このタグは一ついれるだけで成立する。「br」は break の略。

○強調 ** ～ ・ ～ **

html	表示
****初めて****の html ****しっかり****学んでいこう。	**初めて**のhtml *しっかり学んでいこう。*
説明	
タグを使うことで、テキストを太字に変えることができる。タグを使うことで、テキストを斜体に変えることができる。どちらのタグも適応させたい部分をタグで挟むと使用できる。	

2-3 特殊にテキストを表示させるタグ

○リスト　箇条書き** ～ **、番号付き** ～ **

html	表示	説明
** ** ** **国語** ** ** **数学** ** ** **英語** ** ** ** ** ** ** **基礎** ** ** **標準** ** ** **発展** ** ** **	• 国語 • 数学 • 英語 1. 基礎 2. 標準 3. 発展	リストとして表示させるときに用いるタグ。 「ul」は unordered list の略 「ol」は ordered list の略 「li」は list item の略である。

○表　**<table>,<tr>,<th>,<td>**

html	表示	説明
<table> **<tr>** **<th>**名前**</th>** **<th>**年齢**</th>** **</tr>** **<tr>** **<td>**太郎**</td>** **<td>**20**</td>** **</tr>** **</table>**	**名前 年齢** 太郎 20	表形式で表示させるときに用いるタグ。 全体を<table> ～ </table>ではさみ、 <tr> ～ </tr>で行ごとで設定していく。 <th>は見出しの設定、<td>はそれ以外の要素の設定。 「tr」は table row 「th」は table header 「td」は table data の略である。

2-4 その他の表示

◯画像の表示 **<img>**

html	表示	説明
 こんにちは **<img src="https://placehold.jp/3d4070/ffffff/150x150.png?text=dummy" alt="説明">** 	こんにちは  	<imgタグで画像を表示させることができる。 <imgの中に src=”画像のリンクなど“を入れることで、表示させる画像を指定することができる。 この src のように、タグに情報をふかするものを属性という。 <タグ 属性=“属性値”> 左の見本では、img タグに src 属性と説明書きができる alt 属性を追加している。

◯リンクの表示 **<a>**

html	表示
 続きは以下のリンクへ！ ****こちら！**** 	続きは以下のリンクへ！ こちら！
説明	
<aタグで挟むことで、他ページへのリンクを設定することができる。<aタグの中に href 属性でリンク先の URL を設定する。上の例では google へリンクが設定されており、「こちら!」をクリックするとページが移動する。<aタグで挟むものは画像などでもリンクの設定ができる。	

第3章 CSS

3-1 CSS の基本的な適用方法

HTML 文書に CSS を適用するには、いくつかの方法があるが、基本的には HTML ファイルとは別に CSS ファイルを作り、そこで記述し適用するのが一般的である。HTML 文書の head 要素に記述したり、HTML のタグの属性として追加することも可能であるが、管理や修正が煩雑になるためあまり使用しないほうが良いだろう。

外部 CSS ファイルを用いる場合は、HTML 文書の head タグ内に、<link rel=”stylesheet”

href=”style.css”>を追加することで、適用することができる(href の中身は必ず同じにはならない)。本解説では外部 CSS ファイルでの記述を想定している。

3-2 CSS の基本記述方法

html	CSS	表示例

** ようこそ **
** はじめまして、こんにちは。**
** **
	p{
	font-size : 20px;
	color : red;
	}
		ようこそ
		はじめまして、こんにちは。

CSS の記述の基本は右のとおりである。
どこに適用するか:セレクタ、何を変えるか:プロパティ、どのように変えるか:値となっている。今回の見本であれば、
「p タグの内容のフォントのサイズを 20px で赤色に変更する」という CSS である。

基本	例

セレクタ{
プロパティ1 : 値1;
プロパティ2 : 値2;
}
	p{
	font-size : 20px;
	color : red;
	}

3-3 セレクタの種類

セレクタに当てはまるのは p や h3 のようなタグだけでなく、もう少し細かく設定することができる。

①ID セレクタ

html	CSS	表示例

<h3>見出し 1 </h3>
<h4 id="special">見出し 2 </h4>
<h4>見出し 3 </h4>
	#special{
	border-bottom: 1px solid red;
	}
		文章 1
		文章 2
		文章 3

文章2の p タグに id=”special”をつけ、CSS では#special と指定したことで、文章2にだけデザインが適用された。

②クラスセレクタ

html	CSS	表示例

<h3 class="aaa">見出し 1 </h3>
 文章 2
 文章 3
	.aaa{
	border-bottom: 1px solid red;
	}
	
	h3.aaa{
	border-left: 5px solid red;
	}
		見出し 1
		文章 2
		文章 3

HTML 上で見出し1と文章2のタグについて class="aaa"というクラスを設定し、CSS 上で.aaa{ }と記述することで aaa が設定されている見出し1と文章2に下線がついた。さらに、h3.aaa (タグ. クラス名)とすることで、更に細かく指定し、見出しの左側に線を追加した。

③ID セレクタとクラスセレクタの違い

どちらも部分的に選択し、デザインを変えるときに使う記述であるが、大きな違いとして、ID はその HTML 文書では1度しか使えず、クラスは何度でも使える点である。基本的には class のほうが使い勝手が良い。また、タグには ID もクラスも両方設定することができるが、ID のほうが優先される。一つのタグについてクラスは複数種類つけることも可能である。

3-4 その他の設定方法 と

html	表示例
** ** 　あいうえお 　かきくけこ ** ** さしすせそ た****ちつて****と な** **にぬね** **の	あいうえお かきくけこ さしすせそ たちつてと な にぬね の
CSS	

.aaa{
　font-size: 24px
}	

もも特定の範囲を指定して、デザインを変えることができるタグである。タグ自体に意味はなく、このタグで分けられた範囲を指定することができる。 この2つの違いは、 は塊で設定し、は部分的な設定に適しているところである。また で囲んだ「にぬね」が改行されているが、これも divの影響である。

したがって、グループ全体で設定したいときは 、文章の一部分だけ変更したい場合はで設定する方が良い。

3-5 プロパティの例

CSS のプロパティはたくさんの種類があるため、ここではすべての紹介はできないが、よく出てくるプロパティを例にあげておく。なお、上で見本がないものを取り上げる。

プロパティ	内容	値の例
font-weight	文字の太さを指定	400, bold, bolder
text-align	文字の行揃え、均等割付	left(左揃え)
background	背景	blue
padding	要素の周りの余白	10px