プログラミング初級（Python）

多重分岐

早稲田大学グローバルエデュケーションセンター

分岐処理の深淵

最初に正しく、次に速く。

— ケント・ベック

多重分岐とは？

- 前回は分岐処理として**基本的な条件分岐**と**簡単な多重分岐**を学んだ。
- 分岐が３つ以上あるものをここでは多重分岐と呼ぶことにする。
  - 単一分岐：if文
  - 複数分岐：if–else文
  - 多重分岐：if–elif–else文

  ※一般的に定着している区分ではない。
- 今回は**応用的な多重分岐**を学ぶ。
  - 条件に論理を利用。条件として論理を組み合わる。条件の順番を考える。
  - 構文が長い。構文が新しい。構文が複雑。
- 前回は分岐処理を行うための条件としてブーリアンや比較演算を利用。
- 今回は分岐処理を行うための条件として**論理演算**を中心に学ぶ。
- キーワード：if–elif–else文、論理演算を用いた分岐処理、**in演算子**、条件の簡略化、**三項演算子**、**セイウチ演算子**(愛称)、**入れ子**

論理演算と分岐処理

- 論理演算とは論理演算専用の**演算子（オペレータ）**を用いることにより**被演算子（オペランド）**の論理関係を評価することができる演算。
  論理演算子の例　　**論理積**（**and**を用いた論理）の式：オペランド１ and オペランド2
  論理和（**or**を用いた論理）の式　：オペランド1 or オペランド2
  否定（**not**を用いた論理の）式　：not(オペランド)
- オペランドには、**値**や**式**が利用できる。

演算子「**and**」 (かつ) の演算結果

	右辺 True	右辺 False
左辺 True	True	False
左辺 False	False	False

演算子「**or**」 (または) の演算結果

	右辺 True	右辺 False
左辺 True	True	True
左辺 False	True	False

- 演算子 and は、左辺と右辺の論理値が両方TrueであったときだけTrueを返し、どちらか一方がFalse か両方 False のときに False を返す。
- 演算子 or は、左辺と右辺の論理値のどちらか、あるいは両方がTrueであったときにTrueを返し，両方が False のときだけ False を返す。
- 優先順位は、比較演算子よりも低く、代入演算子よりは高い。結合規則は、左から右。
- 分岐処理の条件には、**値**や**式**を指定した。論理演算にも、**値**や**式**を指定することができる。
  → **分岐処理の条件として論理演算を指定**することができる。

論理演算の練習

```
>>> x = 10
>>> x > 3
True
>>> x < 5
False
>>> x > 3 and x < 5
False
>>> x > 3 or x < 5
True
>>> not x
False
>>> not x > 3
False
>>> not x < 5
True
>>> not(x)
False
>>> not(x > 3 and x < 5)
True
>>> not(x > 3 or x < 5)
False
>>> x > 3 and not(x < 5)
True
```

論理積の例。

論理和の例。

括弧をつけても良い。

部分的なnot演算子。

```
>>> 3 < x and x < 5
False
>>> 3 < x or x < 5
True
>>> 3 < x and 5 > x
False
>>> (3 < x) or (x < 5)
True
>>> 3 < x < 5
False
>>> x = 4
>>> 3 < x < 5
True
>>> y = 20
>>> 3 < x < 5 < y
True
>>> 3 < x < 5 > y
False
>>>
```

リテラルを左辺に書いても良い。

リテラルを左辺に書いても良い。

数学みたいにかける。これは、
3 < x and x < 5と同じ意味。

3 < x and x < 5 and 5 > y と
同じ意味。

論理演算を用いた分岐処理（and編）

```
1 n = int(input('Enter an integer: '))
2
3 if n == 0:
4     print('It is 0.')
5 elif n >= -9 and n <= 9:
6     print(f'{n} is a one-digit integer.')
7 else:
8     print(f'{n} is a two or more-digit integer.')
```

integer_and.py

実行結果の例

```
Enter an integer: 0
It is 0.
```

```
Enter an integer: -4
-4 is a one-digit integer.
```

```
Enter an integer: 16
16 is a two-digit integer.
```

このプログラムの特徴

- 変数 n の値を0または1桁もしくは2桁以上の整数に判定する。
- if節では、変数nの値が0と等しい場合に、それをそのまま0と判定する。
- elif節では、if節の条件がFalseで、変数nの値 が−9以上でかつ9以下の場合にnの値を1桁の整数と判定する。
- else節では、 if節とelif節の条件が共にFalseの場合、変数nの値を2桁の整数と判定する。

論理演算を用いた分岐処理（or編）

```
1 n = int(input('Enter an integer: '))
2
3 if n <= -10 or n >= 10:
4     print(f'{n} is a two or more-digit integer.')
5 elif n == 0:
6     print('It is 0.')
7 else:
8     print(f'{n} is a one-digit integer.')
```

integer_or.py

実行結果の例

```
Enter an integer: 16
16 is a two or more-digit integer.
```

```
Enter an integer: 0
It is 0.
```

```
Enter an integer: -4
-4 is a one-digit integer.
```

このプログラムの特徴

- 変数nの値を0または1桁あるいは2桁以上の整数に判定する。
- if節では、 変数nの値 が−10以下または10以上の場合に2桁の整数と判定。
- elif節では、if節の条件がFalseとなり、変数nの値が0と等しい場合にそれをそのまま0と判定。
- else節では、if節、elif節の条件が共にFalseの場合、変数nの値を1桁の整数と判定。
- 動作結果は前のプログラムと同じになる。

論理演算を用いた分岐処理（not編）

```
1 math = int(input('Math: '))
2 eng = int(input('English: '))
3
4 if math >= 60 and eng >= 60:
5     print('Pass.')
6 else:
7     print('Fail.')
```

pass_fail.py

```
1 math = int(input('Math: '))
2 eng = int(input('English: '))
3
4 if not(math >= 60 and eng >= 60):
5     print('Fail.')
6 else:
7     print('Pass.')
```

pass_fail_not.py

実行結果の例

```
Math: 60
English: 60
Pass.
```

```
Math: 40
English: 60
Fail.
```

このプログラムの特徴

- math、eng が両方とも60以上なら合格とする。それ以外は不合格と表示する。
- 条件: math >=60 and eng >=60
- Trueで合格、Falseで不合格

```
Math: 30
English: 30
Fail.
```

このプログラムの特徴

- math、eng が両方とも60以上なら合格とする。それ以外は不合格と表示する。
- 条件: not(math >=60 and eng >=60)
- Trueで不合格、Falseで合格

季節を判定するプログラム

```
1  month = int(input('Enter a month, and I will display the season: '))
2
3  if month >= 3 and month <= 5:
4      print(f'Month {month} is spring.')
5  elif month >= 6 and month <= 8:
6      print(f'Month {month} is summer.')
7  elif month >= 9 and month <= 11:
8      print(f'Month {month} is autumn.')
9  elif month == 1 or month == 2 or month == 12:
10     print(f'Month {month} is winter.')
11 else:
12     print(f'Month {month} is invalid. Please enter a number between 1 and 12.')
```

下の書き方でもOK

```
elif ( month == 1 or
       month == 2 or
       month == 12  ):
```

実行結果の例

```
Enter a month, and I will display the season: 3
Month 3 is spring.
```

```
Enter a month, and I will display the season: 7
Month 7 is summer.
```

```
Enter a month, and I will display the season: 2
Month 2 is winter.
```

このプログラムの特徴　　season1.py

- ３ヶ月ごとに区切って季節を判定するプログラム。
- 3月、4月、5月：春
- 6月、7月、8月：夏
- 9月、10月、11月：秋
- 12月、1月、2月：冬
- 冬の月数はは連続して表せないため、論理式を変更している。

季節を判定するプログラム：連結版

```
1  month = int(input('Enter a month, and I will display the season: '))
2
3  if 3 <= month <= 5:
4      print(f'Month {month} is spring.')
5  elif 6 <= month <= 8:
6      print(f'Month {month} is summer.')
7  elif 9 <= month <= 11:
8      print(f'Month {month} is autumn.')
9  elif month == 1 or month == 2 or month == 12:
10     print(f'Month {month} is winter.')
11 else:
12     print(f'Month {month} is invalid. Please enter a number between 1 and 12.')
```

下の書き方でもOK

```
elif ( month == 1 or
       month == 2 or
       month == 12  ):
```

実行結果の例

```
Enter a month, and I will display the season: 3
Month 3 is spring.
```

```
Enter a month, and I will display the season: 7
Month 7 is summer.
```

```
Enter a month, and I will display the season: 2
Month 2 is winter.
```

このプログラムの特徴　　season2.py

- ３ヶ月ごとに季節を判定するプログラム。
- 3月、4月、5月：春
- 6月、7月、8月：夏
- 9月、10月、11月：秋
- 12月、1月、2月：冬
- 冬の月数はは連続して表せないため、論理式を変更している。
- 一部の論理式を連結している。

多重分岐のプログラム： if-elif-else文の利用

```
1  letter = 'a'
2  if letter == 'a':
3      print(letter, 'is a vowel.')
4  elif letter == 'e':
5      print(letter, 'is a vowel.')
6  elif letter == 'i':
7      print(letter, 'is a vowel.')
8  elif letter == 'o':
9      print(letter, 'is a vowel.')
10 elif letter == 'u':
11     print(letter, 'is a vowel.')
12 else:
13     print(letter, 'is not a vowel.')
```

値を変えてみよう。

vowel_elif.py

実行結果の例

```
a is a vowel.
```

```
v is not a vowel.
```

このプログラムの特徴

- 与えられた文字が母音か子音かを判定するプログラム。
- if-elif-else文を利用した例。
- if節に母音を一つ割り当てて、elif節に残りの母音を割り当てて１文字を比較する条件。
- 母音を全てチェックして該当しなければelse節で子音として処理する。

多重分岐のプログラム：if-else文と論理式の利用

値を変えてみよう。

```
1 letter = 'a'
2 if letter == 'a' or letter == 'e' or letter == 'i' or letter =='o' or letter == 'u':
3     print(letter, 'is a vowel.')
4 else:
5     print(letter, 'is not a vowel.')
```

vowel_logic.py

実行結果の例

```
a is a vowel.
```

```
v is not a vowel.
```

このプログラムの特徴

- プログラム「vowel_elif.py」と実行結果は同じ。
- if-else文を利用した例。
- if節に論理式を用いて条件を一つにした例。

多重分岐のプログラム：in演算子の利用

値を変えてみよう。

```
1 letter = 'a'
2 if letter in 'aeiou':
3     print(letter, 'is a vowel.')
4 else:
5     print(letter, 'is not a vowel.')
```

vowel_in.py

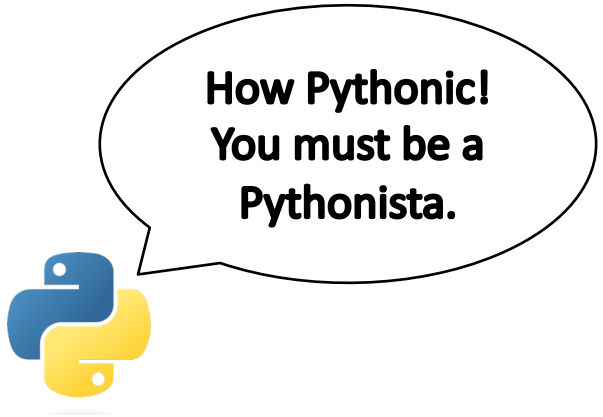
実行結果の例

```
a is a vowel.
```

```
v is not a vowel.
```

このプログラムの特徴

- プログラム「vowel_elif.py」および「vowel_logic.py」と実行結果は同じ。
- if-else文を利用した例。
- if節にin演算子を用いて条件を一つにした例。



推敲の余地がある分岐処理の例（if文の列記）

```
1 score = int(input('Enter a score between 0 and 100: '))
2
3 if score >= 90:
4     print(f'You have scored {score}, which is an excellent score.')
5 if score >= 60 and score < 90:
6     print(f'You have scored {score}, which means you have passed.')
7 if score < 60:
8     print(f'You have scored {score}, which means you have failed.')
```

score_if.py

実行結果の例

```
Enter a score between 0 and 100: 100
You have scored 100, which is an excellent score.
```

```
Enter a score between 0 and 100: 60
You have scored 60, which means you have passed.
```

```
Enter a score between 0 and 100: 40
You have scored 40, which means you have failed.
```

このプログラムの特徴

- **if文を 3個 列記**して分岐処理を行う例。
- 入力値が90以上、60以上90未満、あるいは60未満となる条件で場合分け。
- どのような入力値でも**3、5、7行目のif文で必ず条件判断**（条件の評価）が行われる。
- 条件判断に要する評価回数は**５回**。
- しかし、例えば**3行目のif文で条件式が真**となった場合、このプログラムの条件では、５行目以降の条件式は**必ず偽**と評価されるため、命令の実行が不要。**推敲の余地あり**。
- チェックすべき条件の値が共通であるならif-else文やif-elif-else文の利用が検討可能。

推敲の余地がある分岐処理の例（if節とelif節の併用）

```
1 score = int(input('Enter a score between 0 and 100: '))
2
3 if score >= 90:
4     print(f'You have scored {score}, which is an excellent score.')
5 elif score >= 60 and score < 90:
6     print(f'You have scored {score}, which means you have passed.')
7 elif score < 60:
8     print(f'You have scored {score}, which means you have failed.')
```

score_elif.py

実行結果の例

```
Enter a score between 0 and 100: 100
You have scored 100, which is an excellent score.
```

```
Enter a score between 0 and 100: 60
You have scored 60, which means you have passed.
```

```
Enter a score between 0 and 100: 40
You have scored 40, which means you have failed.
```

このプログラムの特徴

- ２つ目以降のif文をelif節に置き換えた例。
- このように書けばどの節でも必ず条件判定が行われる訳ではないため改善といえる。
- しかし、最後のelif節に分岐処理が移行した場合は条件判断に要する評価回数が５回。
- つまり、評価回数が**最大５回**という意味。
- この場合は改善前のプログラムと効率が同じとなることから、**推敲の余地がまだあり**。
- 上位の条件式の逆の式の真偽値を考えることにより、簡略化の検討が可能。

推敲の余地がある分岐処理の例（if節とelif節の併用および式の簡略化）

```
1 score = int(input('Enter a score between 0 and 100: '))
2
3 if score >= 90:
4     print(f'You have scored {score}, which is an excellent score.')
5 elif score >= 60 and score < 90:
6     print(f'You have scored {score}, which means you have passed.')
7 elif score < 60:
8     print(f'You have scored {score}, which means you have failed.')
```

score_elif2.py

実行結果の例

```
Enter a score between 0 and 100: 100
You have scored 100, which is an excellent score.
```

```
Enter a score between 0 and 100: 60
You have scored 60, which means you have passed.
```

```
Enter a score between 0 and 100: 40
You have scored 40, which means you have failed.
```

このプログラムの特徴

- １つ目のelif節の条件式から「and score < 90」を削除。
- このように書けば条件判断に要する評価回数が**最大3回**となり、改善できたといえる。
- 式「score >= 90」の逆の式は「score < 90」。
- １つ目のelif節に分岐処理が移っているということはこの時点で式「score >= 90」の評価は偽。
- 偽以外のものは真。ならばこの時点で式「score < 90」の評価は演算せずとも真と導くことができる。
- 「and score < 90」部分は「and True」や「* 1」しているようなもの。あってもなくても影響がないので真偽値は式「score >= 60」に依存する。
- 要するに、５回必要だった演算を２回分減らすことができ、評価回数が最大３回となる。
- しかし、このプログラムには、**推敲の余地がまだあり**。
- 同様に１つ目のelif節の逆の条件式の真偽値を考えることにより、最終節に対して簡略化の検討が可能。

推敲の余地がある分岐処理の例（if–elif–else文の使用）

```
1 score = int(input('Enter a score between 0 and 100: '))
2
3 if score >= 90:
4     print(f'You have scored {score}, which is an excellent score.')
5 elif score >= 60:
6     print(f'You have scored {score}, which means you have passed.')
7 else score < 60:
8     print(f'You have scored {score}, which means you have failed.')
```

実行結果の例

```
Enter a score between 0 and 100: 100
You have scored 100, which is an excellent score.
```

```
Enter a score between 0 and 100: 60
You have scored 60, which means you have passed.
```

```
Enter a score between 0 and 100: 40
You have scored 40, which means you have failed.
```

このプログラムの特徴　score_else.py

- 最後の節をelifからelseに変更し、elif節で指定していた条件式「score < 60」を削除。
- このように書けば条件判断に要する評価回数が**最大２回**となり、改善できたといえる。
- 式「score >= 60」の逆は「score < 60」。
- 最後の節に分岐処理が移っているということはこの時点で式「score >= 60」の評価は偽であり、逆の式「score < 60」の評価は演算せずとも真だと導くことができる
- よって、条件を省略できるelse節を適用。
- 確認すべき条件が「score < 60」しか残っていないので式の真偽値を考慮するまでもなく、単にelse節を使うと考えても良い。

清書版

```
1 score = int(input('Enter a score between 0 and 100: '))
2
3 if score >= 90:
4     print(f'You have scored {score}, which is an excellent score.')
5 elif score >= 60:
6     print(f'You have scored {score}, which means you have passed.')
7 else:
8     print(f'You have scored {score}, which means you have failed.')
```

score.py

実行結果の例

```
Enter a score between 0 and 100: 100
You have scored 100, which is an excellent score.
```

```
Enter a score between 0 and 100: 60
You have scored 60, which means you have passed.
```

```
Enter a score between 0 and 100: 40
You have scored 40, which means you have failed.
```

下位の節では条件を簡略化できない分岐処理の例

```
1 score = int(input('Enter a score between 0 and 100: '))
2
3 if score >= 60 and score < 90:
4     print(f'You have scored {score}, which means you have passed.')
5 elif score >= 90:
6     print(f'You have scored {score}, which is an excellent score.')
7 else:
8     print(f'You have scored {score}, which means you have failed.')
```

score_prob01.py

実行結果の例

```
Enter a score between 0 and 100: 60
You have scored 60, which means you have passed.
```

```
Enter a score between 0 and 100: 100
You have scored 100, which is an excellent score.
```

```
Enter a score between 0 and 100: 40
You have scored 40, which means you have failed.
```

このプログラムの特徴

- if節で60以上かつ90未満の条件を指定し、分岐処理を行っている例。
- elif節とelse節では簡略化ができない。
- 簡略化できるかどうかは条件式の内容とその順番次第。

アルゴリズムに問題がある分岐処理の例

```
1 score = int(input('Enter a score between 0 and 100: '))
2
3 if score >= 60:
4     print(f'You have scored {score}, which means you have passed.')
5 elif score >= 90:
6     print(f'You have scored {score}, which is an excellent score.')
7 else:
8     print(f'You have scored {score}, which means you have failed.')
```

score_prob02.py

実行結果の例

```
Enter a score between 0 and 100: 60
You have scored 60, which means you have passed.
```

```
Enter a score between 0 and 100: 100
You have scored 100, which means you have passed.
```

```
Enter a score between 0 and 100: 40
You have scored 40, which means you have failed.
```

このプログラムの特徴

- elif節の条件式が真とならない例。
- if節の条件式が偽となるのはscoreが60未満の場合。
- したがって、elif節では条件が常に偽となる。
- すなわち、scoreが90以上でもelif節は実行されることがなく、if節が常に実行される。
- 結論：**条件式の設定は非常に大事**

三項演算子

```
1 n = int(input('Please enter an integer: '))
2
3 if n % 2 == 0:
4     print(f'{n} is an even number.')
5 else:
6     print(f'{n} is an odd number.')
```

even_odd.py

```
1 n = int(input('Please enter an integer: '))
2 print(f'{n} is an even number.' if n % 2 == 0 else f'{n} is an odd number.')
```

even_odd_ternary.py

実行結果の例

```
Please enter an integer: 2
2 is an even number.
```

```
Please enter an integer: 3
3 is an odd number.
```

このプログラムの特徴

- 条件式が True の場合は左側の値，Falseの場合は右側の値となる

```
値 if 条件式 else 値
```

```
変数 = 値1 if 条件式 else 値2
```

セイウチ演算子:=

```
a = 5
if a > 3:
    print(a)
```

```
if (a := 5) > 3:
    print(a)
```

walrus.py

実行結果の例

```
5
```

このプログラムの特徴

- 英語圏では、:-) を顔文字として使うことがあり。（意味はスマイリー）
- これに似ていて、:= が牙が生えているように見えるのでセイウチという愛称で呼ばれている。
- Python 3.8 から導入された代入演算子の１つ。

分岐処理の入れ子

```
1 n = int(input('Please enter a positive integer: '))
2
3 if n > 0:
4     if n % 2 == 0:
5         print(f'{n} is a positive even number.')
6     else:
7         print(f'{n} is a positive odd number.')
8 else:
9     print('A non-positive value was entered.')
```

内側のif文

外側のif文

if_else_nest.py

実行結果の例

```
Please enter a positive integer: 10
10 is a positive even number.
```

```
Please enter a positive integer: 17
17 is a positive odd number.
```

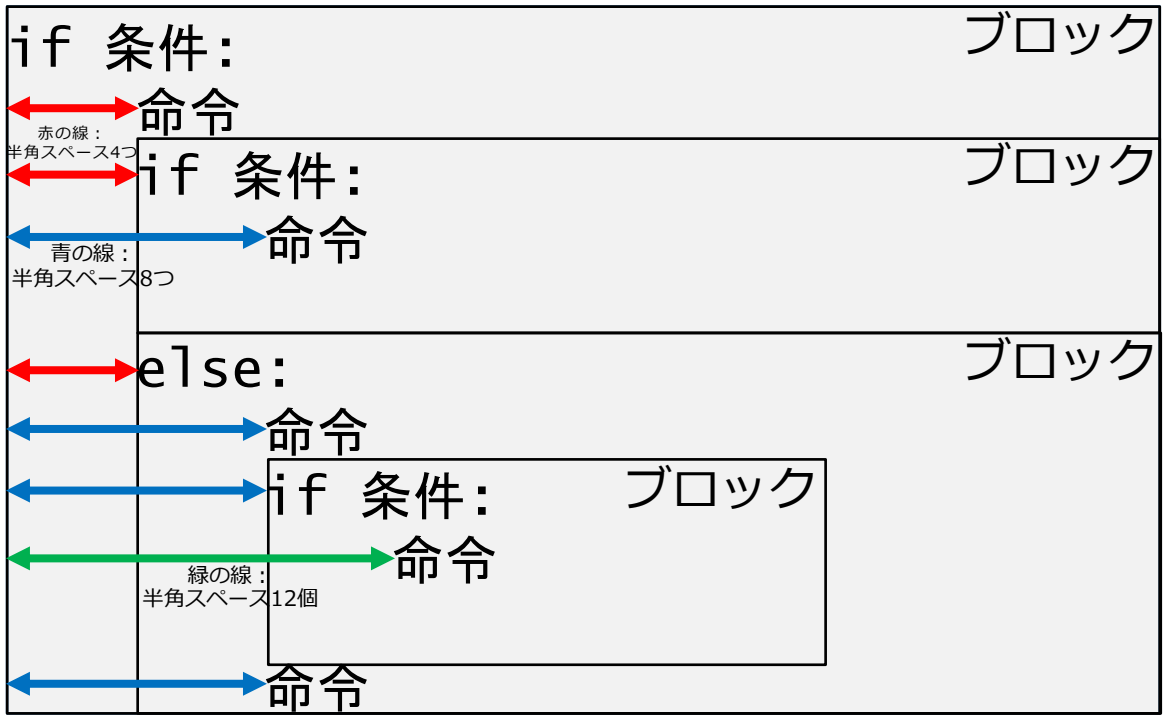
```
Please enter a positive integer: -5
A non-positive value was entered.
```

このプログラムの特徴

- 正の整数を偶数か奇数か判定する。
- もし0以下の値が入力されたらその旨を表示する。
- 内側のif-else文は、外側のif-else文より、インデントを作る。
- 内側のif-else文における命令は、内側if-else文を起点にインデントを作る。
- このように、**ある文の中に同じ文が入っている状態**を**入れ子**という。
- 入れ子の深さに制限はないが、作りすぎは難読化の元。

入れ子の構造

構文



- ブロック：プログラムにおける命令のまとまり。
- ブロックの中に複数のブロックが含まれることもある。
- ブロックごとに文のインデントが異なることに留意。

まとめ

- 論理演算には**論理積(and)**と**論理和(or)**と**否定(not)**があり、これらを組み合わせると様々な論理が表現できる。
- 論理演算には**演算子（オペレータ）**と**被演算子（オペランド）**が必要。
- 分岐処理の条件には値と式を利用。論理演算には値や式を利用。分岐処理の条件として論理演算が利用可能。
- 論理演算の例：「3 < x and x < 5」「3 < x and 5 > x」「3 < x < 5」
- if-elif-else文を用いた多重分岐：**in演算子**でパイソニックにできることも。
- 多重分岐では条件の内容と順番が重要。
- 新しい演算子：**三項演算子** `値 if 条件式 else 値` 、**セイウチ演算子** **:=**
- 分岐処理の**入れ子**
  - あるブロックの中に別のブロックが入り込んだ状態。これを入れ子構造という。
  - **入れ子ごとにインデント**が必要。入れ子の数に制限はないが、**深入りには注意**。

次回予告　**反復処理**