プログラミング教室のテクノロ

Pythonの道

トレーニングドリル②

もくじ

・真偽値と条件分岐
・お買い物代金を計算しよう

条件分岐とは

ここからはしばらく条件分岐について学ぶ。
プログラミングでは、とある条件に当てはまるかどうかによって処理を分けることがよくある。
例えば「テストの点数（条件）によって成績（処理）を変える」というのも条件分岐である。

条件分岐のイメージ

if文

if文を用いると「もし○○ならばXXを行う」という条件分岐が可能になる。if文は以下のように書く。ifの後に条件式を指定し、その条件が成り立つときに実行する処理を次の行に書く。
詳しい書き方については次のスライドから見ていこう。

【コード】

```
score = 100
if score == 100:  ※条件式が成り立つ
    print('よくできました！')
```

【出力結果】

よくできました。※条件式が成り立つため実行される

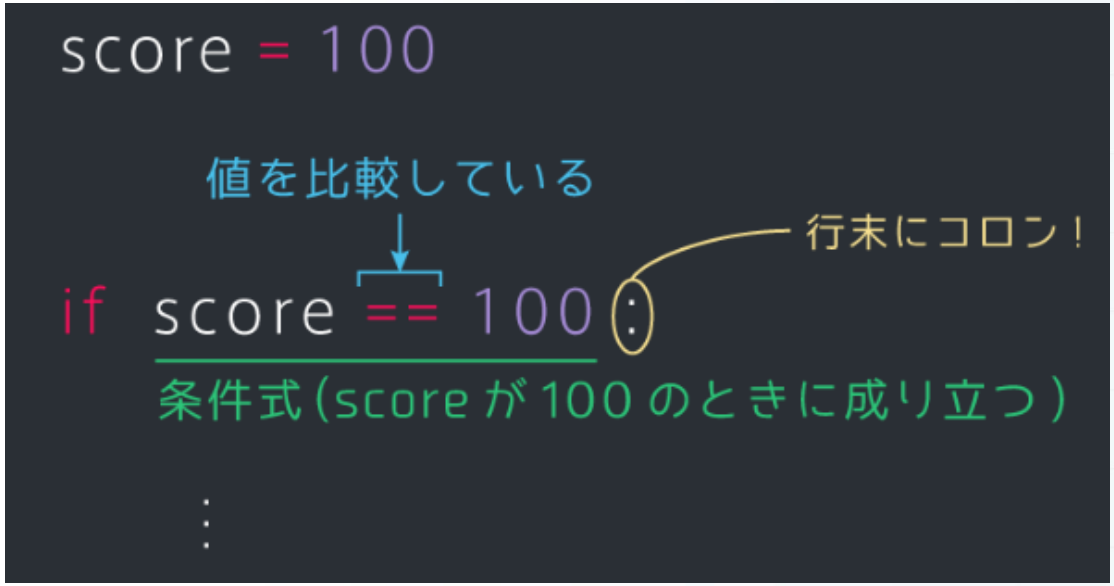
if文の条件式

まずは、if文の条件の作り方を詳しく見ていこう。
条件式の中では、２つの値を比較するための記号「比較演算子」がよく使われる。まずは下図の2つを覚えよう。
if文の条件部分は下図のように「if　条件式　:」のように書く。

【比較演算子：等しいかを調べる】

x　==　y　　※左右の値が等しい時成り立つ

x　!=　y　※左右の値が等しくない時成り立つ



インデント

if文の条件式が成立した時の処理を書くときには、インデント（字下げ）をする。図のように、処理がif文の中にあるかどうかはインデントによって判別される。条件が成立したときにif文の中の処理が実行される。Pythonではコードの見た目（インデント）がそのままプログラムの動作に影響するので、インデントに気をつけよう。

【コード】

```
score = 50
if score == 100:
    print(‘ よくできました！’ )
    print(‘ 次も頑張りましょう！’ )
```

※インデントをそろえる(半角スペース4つ分)

【出力結果】

※条件式が成り立たないため実行されない

インデント

【コード】

```
score = 50
if score == 100:
    print(‘よくできました！’)
print(‘次も頑張りましょう！’)
```

※if分の中身

※if分の外

※インデントがないため、if分の外と見なされる

【出力結果】

次も頑張りましょう　※条件式が成り立たないため実行されない

練習

問題を読んでコードを書いてみよう。

【問題】

```
x = 7 * 10
y = 5 * 6

# xが70と等しい場合に「xは70です」と出力してください

# yが40と等しくない場合に「yは40ではありません」と出力してください
```

練習

【答え】

```
x = 7 * 10
y = 5 * 6

# xが70と等しい場合に「xは70です」と出力してください
if x == 70:
    print('xは70です')

# yが40と等しくない場合に「yは40ではありません」と出力してください
if y != 40:
    print('yは40ではありません')
```

真偽値

if文の条件式の部分をもう少し詳しく見てみよう。
if文の比較演算子を用いた条件式の部分を出力してみると以下のように「True」が出力される。
この「True」とは一体なんだろうか？次のスライドで説明する。

```
score = 100
if score == 100:
    print('よくできました！')
```

```
score = 100
print(score == 100)
```

```
>_ コンソール
True
```

真偽値とは？

先ほどの例で出力された「True」は真偽値と呼ばれるもの。
真偽値を扱うデータ型、「真偽値型」には「True」と「False」という2つの値が存在する。比較演算子を用いた条件式の部分が、成り立つときは「True」、成り立たないときは「False」となる。「True」と「False」のそれぞれの頭文字は大文字なので注意する。

【コード】

```
print(3==3)  ⇒True
print(3==5)  ⇒False
```

【出力結果】

```
True
False
```

if文と真偽値

先ほどのスライドで紹介した真偽値に関する内容を踏まえて、
もう一度if文を見てみよう。
if文では条件式がTrueのときには処理が実行され、Falseのときには処理は実行されない。

【コード】

```
score = 100
if score == 100: ※True（条件式が成立する）
    print(‘よくできました！’)
```

【出力結果】

```
よくできました。
```

【コード】

```
score = 50
if score == 100: ※False（条件式が成立しない）
    print(‘よくできました！’)
```

【出力結果】

※条件式がFalseになるので実行されない

比較演算子(<,<=,>,>=)

比較演算子には、==と!=といった値の等しさを比較する記号以外に、値の大小を比較する記号が存在する。数学でも用いられる<, >という大小比較の記号です。「x > y」はxがyより大きければTrue, 小さければFalseを返す。「x < y」はその逆になる。また数学で同様に用いる≧, ≦という記号（以上や以下を表します）は、>=, <=と記述する。

【比較演算子　大小を比べる】

```
x < y      ※xがyより小さいとき　True
x <= y     ※xがyより小さいまたは等しいとき　True
x > y      ※xがyより大きいとき　True
x >= y     ※xがyより大きいときまたは等しいとき　True
```

【出力結果】

```
2 > 6          ※False
4 + 2 >= 6     ※True
8 / 4 < 5      ※True
2 * 5 <= 9     ※False
```

練習

問題を読んでコードを書いてみよう。

【問題】

```
x = 10
# xが30より大きい場合に「xは30より大きいです」と出力してください
```

```

```

```
money = 500
apple_price = 200
# moneyの値がapple_priceの値以上の時、「りんごを買うことができます」と出力してください
```

```

```

練習

【答え】

```
x = 10
# xが30より大きい場合に「xは30より大きいです」と出力してください
if x > 30:
    print('xは30より大きいです')

money = 500
apple_price = 200
# moneyの値がapple_priceの値以上の時、「りんごを買うことができます」と出力してください
if money >= apple_price:
    print('りんごを買うことができます')
```

条件に合致しない時の処理

if文を用いることで条件が成り立つ時の処理を行えるようになった。次は条件が成り立たなかった時に別の処理を行うような条件分岐の書き方を学う。

if文- 条件が成り立つとき

もし、テストの点数が100点だったら...

else, elif - 条件が成り立たなかったとき

もし、テストの点数が100点ではなかったら...

else

if文に「else」を組み合わせることで「もし○○ならばＸＸを行う、そうでなければ△△を行う」という条件分岐ができるようになる。if文の条件がFalseのとき、elseの処理が実行される。

【コード】

```
score = 50
if score == 100:
    print(‘ よくできました！’ )
else:
    print(‘ 頑張りましょう’ )
```

【出力結果】

頑張りましょう

練習

問題を読んでコードを書いてみよう。

【問題】

```
money = 100
apple_price = 200

if money >= apple_price:
    print('りんごを買うことができます')
# if文の条件に当てはまらない場合に「お金が足りません」と出力してください
```

練習

【答え】

```
money = 100
apple_price = 200

if money >= apple_price:
    print('りんごを買うことができます’ )

# if文の条件に当てはまらない場合に「お金が足りません」と出力してください
else:
    print('お金が足りません')
```

elif（1）

if文で、条件が成り立たなかった場合を複数定義したい場合は、「elif」を用いる。elifを用いると、「もし○○ならばxxを行う、△△ならば▲▲を行う、そうでなければ□□を行う」という処理ができるようになる。

【コード】

```
score = 70
if score == 100:  ※False
    print(‘ よくできました！’ )
elif score >= 60:  ※True
    print(‘ まずまずです’ )  ←実行される
else :
    print(‘ 頑張りましょう’ )
```

【出力結果】

まずますです。

elif（2）

elifはいくつでも書くことができるが、上から順に条件が成り立つか判断され、最初に条件に合致した部分の処理だけが行なわれる。

【コード】

```
score = 100
if score == 100: ※True　←最初の条件式が成立したのでそれ以降の処理は実行されない
    print(‘ よくできました！’ )
elif score >= 60:
    print(‘ まずまずです’ )
elif score > 40 :
   .
   .
```

【出力結果】

よくできました。

練習

問題を読んでコードを書いてみよう。

【問題】

```
money = 100
apple_price = 100

if money > apple_price:
    print('りんごを買うことができます')
# 変数の値が等しい場合に「りんごを買うことができますが所持金が0になります」と出力してください

else:
    print('お金が足りません')
```

練習

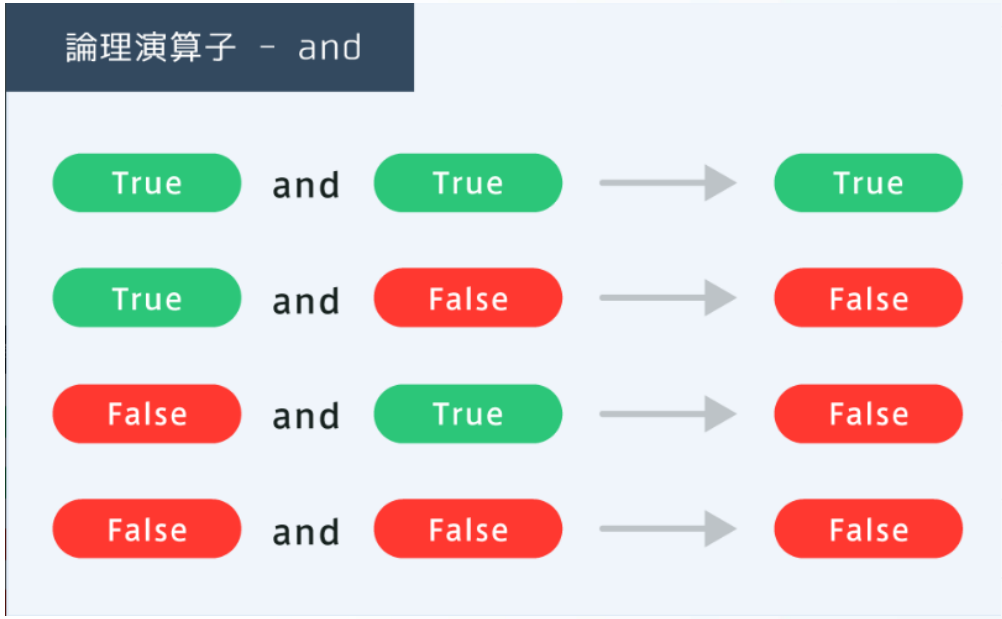
【答え】

```
money = 100
apple_price = 100

if money > apple_price:
    print('りんごを買うことができます')
# 変数の値が等しい場合に「りんごを買うことができますが所持金が0になります」と出力してください
elif money == apple_price:
    print('りんごを買うことができますが所持金が0になります')
else:
    print('お金が足りません')
```

条件式を組み合わせよう（and）

複数の条件式を組み合わせる方法を学んでいこう。
「条件1も条件2も成り立つ」というような場合の条件式は「and」を用いて、「条件1 and 条件2」のように書く。
「and」を用いて複数の条件式を組み合わせると、全ての条件式がTrueの場合に全体がTrueとなる。



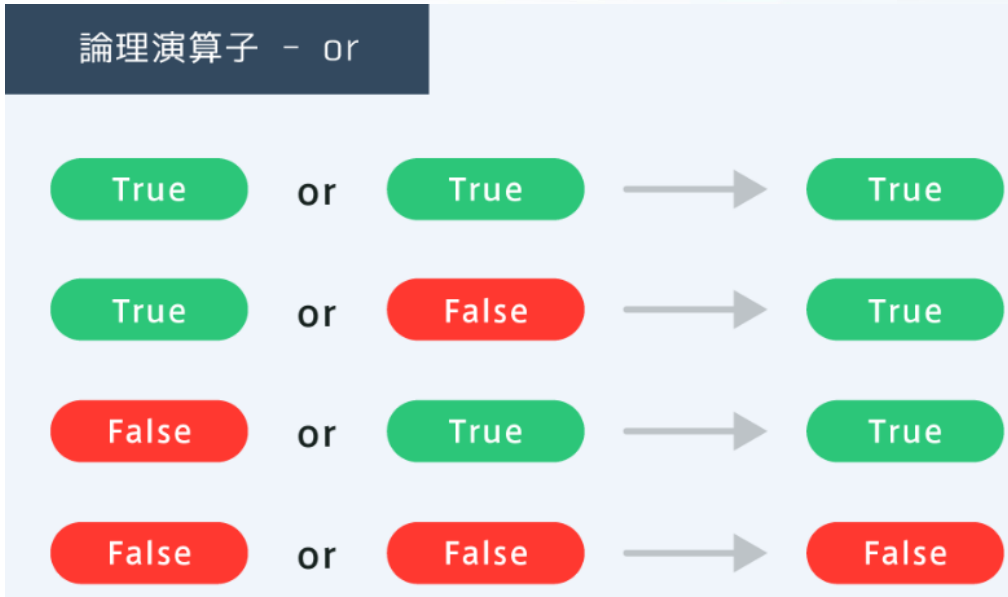

```
time = 14

if time > 10 and time < 18:
    print('就業時間です')

# 結果：就業時間です
```

条件式を組み合わせよう（or）

「条件1か条件2が成り立つ」というような場合の条件式は
「or」を用いて、「条件1 or 条件2」のように書く。
「or」を用いて複数の条件式を組み合わせると、複数の条件式のうち1つでもTrueであれば全体がTrueとなる。




```
time = 15

if time == 10 or time == 15 :
   print('おやつの時間です')

# 結果：おやつの時間です
```

条件式を組み合わせよう（or）

「not」を用いると、条件の否定をすることができる。
「not 条件式」のようにすると、条件式が「True」であれば全体が「False」に、「False」であれば「True」になる。

```
time = 9

          False
if not (time == 18):
        True

    print('退社時刻ではありません')

# 結果：退社時刻ではありません
```

andを使わない書き方

比較を連結してandの条件を下記のように続けて書くこともできる。

```
time = 14
      True          True
if (time > 10) and (time < 18):
    └──────True──────┘
    print('就業時間です')

# 結果：就業時間です
```

```
time = 14
      True    True
if (10 < time < 18):
    └──True──┘
    print('就業時間です')

# 結果：就業時間です
```

andを使わずに書くことができる

練習

問題を読んでコードを書いてみよう。

【問題】

```
x = 20
# xが10以上30以下の場合に「xは10以上30以下です」と出力してください
```

```

```

```
y = 60
# yが10未満または30より大きい場合に「yは10未満または30より大きいです」と出力してください
```

```

```

```
z = 55
# zが77ではない場合に「zは77ではありません」と出力してください
```

```

```

練習

【答え】

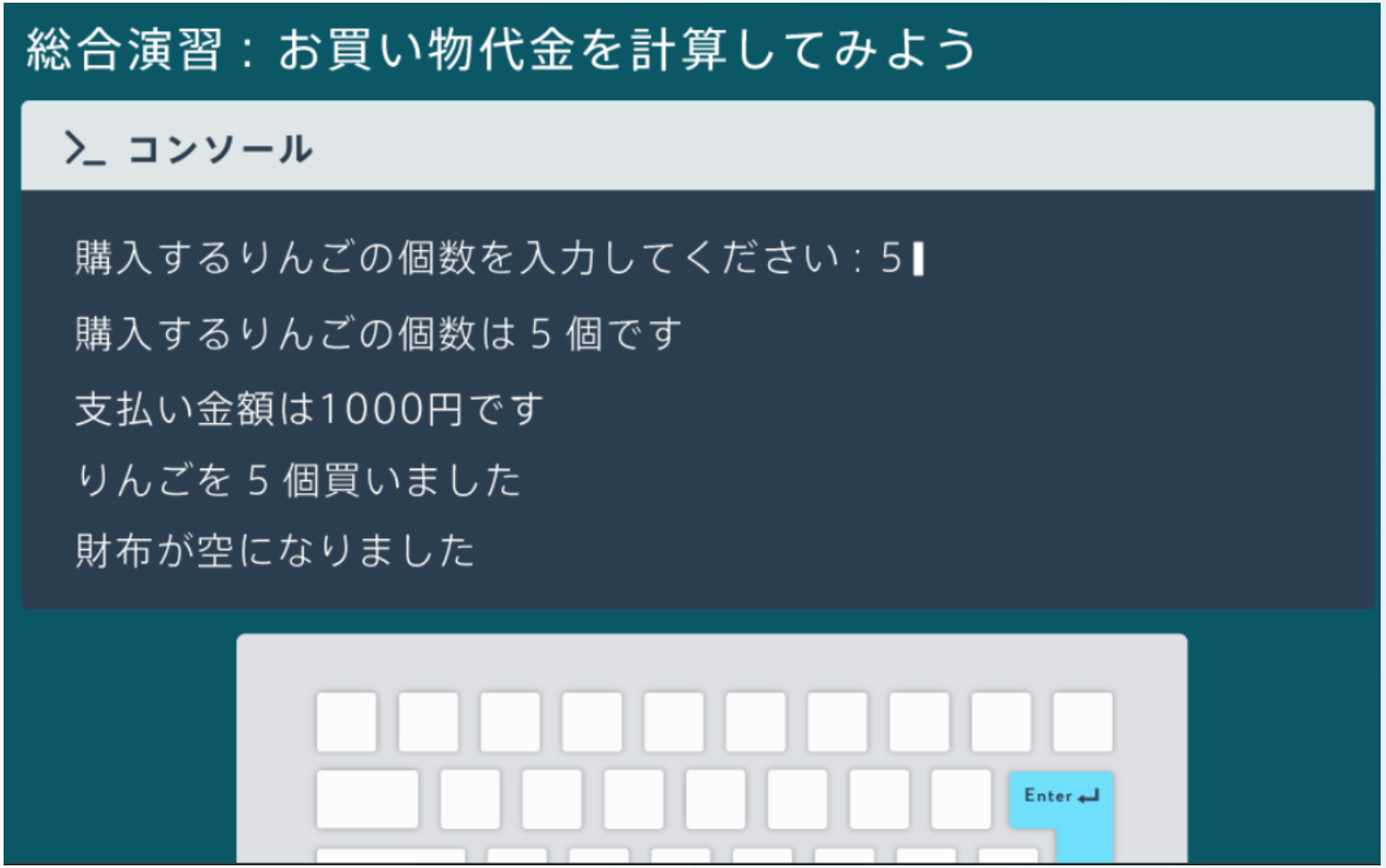
```
x = 20
# 変数xが10以上30以下の場合に「xは10以上30以下です」と出力してください。
if 10 <= x and x <= 30:
    print('xは10以上30以下です')

y = 60
# 変数yが10未満または30より大きい場合に「yは10未満または30より大きいです」と出力してください。
if y < 10 or 30 < y:
    print('yは10未満または30より大きいです')

z = 55
# 変数zが77ではない場合に「zは77ではありません」と出力してください。
if not z == 77:
    print('zは77ではありません')
```

お買い物代金を計算しよう

ここからは、いままで学んだことを組み合わせてお買い物代金を計算する簡単なプログラムを作っていく。
図のように、コンソールにりんごの個数を入力し、個数に応じた処理内容を出力する。



練習

問題を読んでコードを書いてみよう。

【問題】

apple_priceという変数に数値200を代入してください

countという変数に数値5を代入してください

total_priceという変数に、apple_priceとcountを掛けたものを代入してください

「購入するりんごの個数は○○個です」となるように出力してください

「支払い金額は○○円です」となるように出力してください

練習

【答え】

```
# apple_priceという変数に数値200を代入してください
apple_price = 200

# countという変数に数値5を代入してください
count = 5

# total_priceという変数に、apple_priceとcountを掛けたものを代入してください
total_price = apple_price * count

# 「購入するりんごの個数は○○個です」となるように出力してください
print('購入するりんごの個数は' + str(count) + '個です')

# 「支払い金額は○○円です」となるように出力してください
print('支払い金額は' + str(total_price) + '円です')
```

購入する個数を変えよう

1つ前の演習では購入するりんごの個数が決められていた。
ここでは購入するりんごの個数を自由に決められるようにしよう。

Before - 購入する個数が決められている

>_ コンソール

```
購入するりんごの個数は 5 個です
支払い金額は 1000 円です
```

After - 入力した値に応じて購入する個数を変更できる

>_ コンソール

```
購入するりんごの個数を入力してください：3
購入するりんごの個数は 3 個です
支払金額は 600 円です
```

入力を受け取ろう

「input」を用いると、コードを実行した際にコンソールに文字を入力できるようになり、その入力された値を受け取ることができる。「変数 ＝ input('コンソールに表示したい文字列')」のように使うとコンソールに入力された値が変数に代入される。

```
input_count = input('購入するりんごの個数を入力してください：')
```

コンソールから入力を受け取り、変数 input_count に代入

```
    ⋮

print('購入するりんごの個数は' + input_count + '個です')
```

```
購入するりんごの個数を入力してください：3
購入するりんごの個数は3個です
```

入力した値の型変換

inputでコンソールに入力された値を受け取れるようになった。しかし、inputで受け取った値は文字列型になっている。数値として扱いたい場合には型変換を使って下図のようにint型に変換しよう。

```
apple_price = 200
input_count = input('... りんごの個数を入力してください：')
                    input で受け取った値は文字列型

count = int(input_count)
        数値型に変換

total_price = apple_price * count
   ⋮                数値型      数値型
print('支払い金額は' + str(total_price) + '円です')
```

```
購入するりんごの個数を入力してください：3

購入するりんごの個数は 3 個です

支払い金額は 600 円です ←― 正しい金額が出力されている
```

練習

問題を読んでコードを書いてみよう。

【問題】

```
apple_price = 200

# inputを用いて入力を受け取り、変数input_countに代入してください


# input_countを数値として代入してください
count = 
total_price = apple_price * count

print('購入するりんごの個数は' + str(count) + '個です')
print('支払い金額は' + str(total_price) + '円です')
```

練習

【答え】

```
apple_price = 200

# inputを用いて入力を受け取り、変数input_countに代入してください
input_count = input('購入するりんごの個数を入力してください：')

# input_countを数値として代入してください
count = int(input_count)
total_price = apple_price * count

print('購入するりんごの個数は' + str(count) + '個です')
print('支払い金額は' + str(total_price) + '円です')
```

条件分岐をしよう

最後に、条件分岐を使って必要な処理を追加していこう。
以下のように、所持金と支払い金額を用いて条件分岐を行う。

練習

問題を読んでコードを書いてみよう。

【問題】

```
apple_price = 200
# 変数moneyに数値1000を代入してください

input_count = input('購入するりんごの個数を入力してください：')
count = int(input_count)
total_price = apple_price * count

print('購入するりんごの個数は' + str(count) + '個です')
print('支払い金額は' + str(total_price) + '円です')

# moneyとtotal_priceの比較結果によって条件を分岐してください
```

練習

【答え】

```
apple_price = 200
# 変数moneyに数値1000を代入してください
money = 1000

input_count = input('購入するりんごの個数を入力してください：')
count = int(input_count)
total_price = apple_price * count

print('購入するりんごの個数は' + str(count) + '個です')
print('支払い金額は' + str(total_price) + '円です')

# moneyとtotal_priceの比較結果によって条件を分岐してください
if money > total_price:
    print('りんごを' + str(count) + '個買いました')
    print('残金は' + str(money - total_price) + '円です')
elif money == total_price:
    print('りんごを' + str(count) + '個買いました')
    print('財布が空になりました')
else:
    print('お金が足りません')
    print('りんごを買えませんでした')
```

復習＆チャレンジ

ここまで習ったことをScratchでもできるかチャレンジしてみよう。
その過程でScratchでできること、Pythonでないとできないことを整理してみよう。
例えば、データの型という概念はscratchにはあっただろうか？

メモ

プログラミング教室のテクノロ

なまえ：