3. 値と変数：values and variables

プログラミング・データサイエンス I

2021/5/6

1　今日の目標

今日の目標

- 変数の型
- 複合代入演算子
- 文字列
- 論理演算

今日は、変数の型の話から始めます。その次に、プログラムを簡潔に書くための、複合代入演算子を紹介します。使いこなせると、スマートなプログラムを書けるようになります。その後で、文字列の処理を少し行います。最後は、論理演算です。

今日は、前回配布したプログラムを使います。

2　数値の演算：calculating values

数値の演算：calculating values

- 数値は、桁の制限があることに注意
- 数型：int 型
- 浮動小数型：float 型
- 複素数型:complex 型
  - 虚数単位は `j`
  - 例：`a = 3 + 2j`

今回は、変数とそこに格納できる値について考えます。プログラミングでは、数学とは少し異なる点があります。

コンピュータで扱うデータは、内部では、全て二進数で表現されています。また、メモ

リーは有限です。従って、数値には桁の制限があります。

数値などを保存するのが変数です。変数名には、アルファベットと数字を使うことができますが、必ずアルファベットで始まる必要があります。

Python では、数値には、「型」があります。int 型は、整数を扱うことができます。符号も付きます。float 型（浮動小数点型）は、小数点のある数値を扱うことができます。さらに、complex 型といって、複素数を扱うことができるものも用意されています。

2.1 varsAndTypes.ipynb

varsAndTypes.ipynb

- 変数の型を確かめる
- 演算によって型が変わる
- 文字列から数値へ、数値から文字列へ

前回配布した varsAndTypes.ipynb を VSCode で開けましょう。二番目のセルでは、一番目のセルで定義した変数の型を印刷しています (ソースコード 2.1)。type() で、型を調べることができます。実際に、一番目のセル、二番目のセルの順に実行してみましょう。

ソースコード 2.1　型の印刷

```
1 print(type(a))
2 print(type(b))
3 print(type(c))
```

4 番目のセルを見てください (ソースコード 2.2)。整数の積は整数のままですが、商は float 型になっています。

ソースコード 2.2　型の変化

```
1 d = a * b
2 f = a / b
3 print(d)
4 print(type(d))
5 print(f)
6 print(type(f))
```

強制的に型を変換する例が 5 番目のセルにあります (ソースコード 2.3)。g は float 型、h は int 型です。

ソースコード 2.3　型の変換

```
g = float(d)#int から float へ
print(g)
print(type(g))
h = int(g)
print(type(h))
```

プログラムの中で、数値とその数値を文字列にしたものは、別のものです。混ぜて演算を行うことはできません。後で詳しく説明しますが、Python では、文字列をシングルクォーテーションかダブルクォーテーションで囲って表します。ソースコード 2.4 では、変数 g が保持する数値を文字列に変換し、文字列'0.01' を float 型に変換しています。

ソースコード 2.4　型の変換:文字列

```
s = str(g)#float から str へ
print(s)
h = float('0.01')
print(h)
```

課題 1　文字列として作成した"1"と、数値 (int) をして作成した 1 を足すとエラーになることを確かめなさい。

3　複合代入演算子

複合代入演算子

- ある変数に演算を行い、元の変数に代入
- 慣れると、プログラムが書きやすい
  - 慣れるまでは、無理に使わない

前回、四則演算なおの基本となる演算を扱いました。Python には、プログラムを効率的に書くための複合代入演算子があります (表 1)。これらは、他のプログラミング言語にもよく見られます。

一つだけ説明します。最初の a += b を見てください。これは a = a + b と同じ意味です。a = a + b だと、両辺に a が現れて、=の記号が気持ち悪いと感じる人も居るでしょう。そこで、a += b と書くと、変数 a に b を足すという意味が、慣れると、わかりやすくなります。また、a += b のほうが少しだけ短くなります。

複合代入演算子は、慣れると非常に便利です。慣れない場合には、使う必要はありません。中途半端な理解で使うのは、むしろ危険です。

演算子	例	説明
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
//=	a //= b	a = a // b
%=	a %= b	a = a % b
**=	a **= b	a = a ** b

表 1　複合代入演算子

3.1　simpleSum1.ipynb

simpleSum1.ipynb

- 複合代入演算子の使用例
- データの和と計算する

二番目のセルを見てください。前回に simpleSum0.ipynb で扱った例と同じです。s2 という変数に、三つの変数の値を順に加えています。前回は普通の + 演算を使いましたが、今回は +=を使っています。

```
1 s2 = 0
2 s2 += a
3 s2 += b
4 s2 += c
5 print(s1,s2)
```

1 から 10 までの和計算もしてみましょう。3 番目のセルでは、s3 という変数に 1 から 10 まで足しています。for のところは、まだ扱っていませんが、k の値が 1 から 11 の一つ手前まで変化しながら繰り返します。

```
1 s3 = 0
2 for k in range(1,11):#k を 1から 10まで変化させる
3     s3 += k
4 print(s3)
```

4 文字列：Strings

文字列：Strings

- '' または、""で表記
- 文字列の連結
- 文字列と数値の連結に注意
- 文字列から文字を取り出す
- 部分文字列を取り出す
- immutable(変更不能) であることに注意

文字列の扱いです。先ほども出てきましたが、シングルクォーテーションまたはダブルクォーテーションで囲んで表記します。python では、二つのクォーテーションを区別しません。

文字列の様々な操作も可能です。

文字列は変更不能なものとして定義されています。必要な時に説明します。

`stringSamples.ipynb` を VSCode で開き、実行してみましょう。

最初のセルは、文字列を変数に代入する例です。`str3` がどのように印刷されるかを確認してください。

2 番目のセルでは、複数行にわかるテキストを変数に代入しています。ダブルクォーテーション 3 つで囲んでいます。

4 番目のセルを見てください (ソースコード 4.1)。文字列には位置の番号があります。先頭が 0 です。最後尾からも数えることができます。末尾が-1 です。範囲を指定することもできます。

最後のセルでは、文字列を繰り返したものを生成しています。

ソースコード 4.1　文字列の位置

```
1 alphabet = "abcdefghijklmnopqrstuvwxyz"
2 print(alphabet[5])
3 print(alphabet[−10])
4 print(alphabet[0:5])
5 print(alphabet[−10:−1])
```

5 論理演算：Logical operations

論理演算：Logical operations

- 二つの論理値
  True , False
- 論理演算
  and, or , not
- 比較演算の結果は論理値になることに注意
  論理演算可能

演算子	例	説明
==	a == b	a と b の値が等しい
!=	a != b	a と b の値が等しくない
>	a > b	a の値は b の値より大きい
>=	a >= b	a の値は b の値以上
<	a < b	a の値は b の値より小さい
<=	a <= b	a の値は b の値以下
is	a is b	a と b は同じオブジェクト
is not	a is not b	a と b は異なるオブジェクト

表 2　比較演算

最初の回に、「条件分岐」というキーワードが出てきました。ある条件を満たすとき、満たさないときに、異なる処理をするものです。

「条件」を記述するためには、比較という操作が必要になります。「等しい」、「以上」などです。表 2 を見てください。==が「値が等しい」ということを判定する記号です。

「オブジェクト」は馴染みがないかも知れません。オブジェクト (object) とは、モノという意味があります。これまで見てきた変数では、整数や浮動小数は数という一つの値でした。しかし、文字列は、文字の集まりですね。この後、さまざまな値の塊がでてきます。python では、変数は全てオブジェクトとして扱います。

「同じオブジェクト」というのは、わかりにくいので、少し先で説明します。

比較を行うと結果は、「真 (`True`)」と「偽 (`False`)」のいずれかになります。`True` か

False の値をとる変数のことを Boolean (ブール型) と言います。論理型なので、論理演算できます。and は「かつ」、or は「または」、そして not は否定です。

booleanTest.ipynb を開きましょう。最初のセルは、比較した結果がどのような値となっているかを調べています (ソースコード 5.1)。3 行目の右辺を見てください。() で論理演算を囲うことで、その結果を左辺に代入しています。実行すると、それぞれの結果が True または False となって返ってきていることがわかります。

ソースコード 5.1　論理演算とその結果の値

```
1  #Bool 型のテスト
2  x = 8
3  a1 = (0 <= x < 10)
4  print(a1)
5  a2 = (x >= 10)
6  print(a2)
7  a3=((0 <= x) and (x < 10))
8  print(a3)
9  a4 = (not a2)
10 print(a4)
```

ソースコード 5.2　論理演算とその結果の値

```
1  p = True
2  q = False
3  r = (p and q)
4  print(r)
```

二番目のセル (ソースコード 5.2) では、二つの論理変数 p と q に値を代入しています。その後で、論理積を r に代入しています。

三番目のセルを見てください (ソースコード 5.3)。abc という文字列を str0 に代入しています。str1 に str0 を代入したので、str1 にも abc が入っています。もっと大事なことは、str0 と str1 は同じオブジェクトであるということです。2 行目の直後に

```
print(str0 is str1)
```

として確かめてみましょう。True となります。

3 行目で str0 の後ろに def と文字列を追加しました。文字列は「変更不能」というのを前に言いました。str0 という名前を使っていますが、オブジェクトとして別物になっています。従って 6 行目では、False となります。

ソースコード 5.3　文字列が変更不能であること

```
str0 = "abc"
str1 = str0
str0 = str0 + "def"
print(str0)
print(str1)
print(str0 is str1)
```

課題 2　or と not の例題を作成し、動作を確かめなさい。

6　次回

プログラム言語には、良く使われる機能がライブラリとして備わっています。次回は、標準的なライブラリの使い方を学びます。教科書では、4 章「標準ライブラリ」です。