クレジット:
UTokyo Online Education Education コンピュータシステム概論 2018 小林克志

東京大学
THE UNIVERSITY OF TOKYO

CC BY NC ND

Python の基本

この資料はThe Python Tutorial (https://docs.python.org/3.6/tutorial/index.html#the-python-tutorial)(日本語版 (https://docs.python.jp/3/tutorial/)) および Python for Data Analysis:Wrangling with Pandas, Numpy and IPython (http://shop.oreilly.com/product/0636920050896.do)を参考に作成した。
このチュートリアルでは他のプログラミング言語を習得している方を想定し、Python の特徴などを説明する。

ミュータブル、イミュータブル(mutable, immutable)なオブジェクト

Python プログラミングでは変数・オブジェクトがmutable（可変）あるいはimmutable（不変）かに注意しなければならない。
たとえば、文字列、`"str"` は immutable で変更できない、一方リスト `["s", "t", "r"]` は mutable で変更可能となっている。

Python の組み込み型

これまでに説明した、数(int, float)型を含め以下のような組み込み型がある:

スカラー型:

1. 数
2. ブーリアン（真理値判定）
3. 文字列
4. None

データ構造:

1. リスト
2. タプル
3. 集合
4. 辞書

ここではスカラー型のブーリアン型から説明する。

ブーリアン型

条件分岐・ループといった制御構造で重要になる、ブーリアン（ブール、論理値）型は、 `True`あるいは `False` の２値をとる。
ブーリアン型の演算子として、`or`, `and`, `not` が用意されている:

In [ ]:
```
# すべて True
print(True)

print (not False)
print (True and True)
print (True or False)
print (not(False and False))
```

比較演算子

数値の比較演算子、<,<=, >, >=, ==, != はブーリアン型を返す。
また、変数・オブジェクトの同一性を比較する、is, is notもブーリアン型を返す。

詳しくはリファレンスマニュアル (https://docs.python.jp/3/library/stdtypes.html#comparisons)を参照のこと:

In [ ]:
```
# すべて True
print( 1 == 1)
print( 1 < 2)
print( 1 <= 1)

a = 1
b = 1
print( a is b)
c = a
print( a is c)

# 新しいオブジェクトが生成される
c = c + 1
```

文字列型

既定の文字コード(参考)

なにも指定がなければ Python は UTF-8 を既定の文字コードとして扱う。 したがって、コメントをはじめとする文字列に日本語も利用することができる。

文字列

プログラム中の文字列は引用符 ' あるいは二重引用符 " の囲みで記述する。
引用符はどちらを使ってもよい。
複数行にまたがる文字列は３つの引用符 ''' 、あるいは二重引用符 """ でブロックを囲む。 この場合改行も文字列の文字となることに注意する。

In [ ]:
```
a = "This is a comment."
b = """
    This is a block comennt.
    """
print(a)
print(b)
```

エスケープシーケンス

いずれかの引用符が文字列中にある場合、バックスラッシュ\でエスケープする。
見やすさのため文字列にない引用符が使われる。
付け加えると、\自身をエスケープするには、２回バックスラッシュを繰り返す。

したがって、以下の a,b は同じ文字列である:

In [ ]:
```
a = "I'm a student."
b = 'I\'m a student.'

a == b
```

raw 表記

Windows のディレクトリなど\ が多い場合は文字列前に r を加える raw 表記が使える:

In [ ]:
```
r'C:\System\System32\command.com'
```

文字列の連結

+ によって文字列の連結、* によって反復する。
引用符で囲った連続する文字列はそのまま連結されるため+ は不要:

In [ ]:
```
a = "foo "
b = "bar "

print(a+b)
print(a+b*3)

c = "foo " "bar "
d = "foo " + "bar "
print(c)
print(d)
```


文字列のインデックス（添字表現）

[ ]内の添字(インデックス)で文字列の位置を指定、文字列の一部を得る。
インデックスに数字のみ与えると、先頭から数えた位置の一文字の文字列を得る。
基本型に文字はないことに注意する。
インデックスに負数を与えると末尾から数えた位置となる。

インデックスに:を使うことで取り出す範囲を指定することができる（スライス）。
:の右辺を省略すると 0 (先頭)が、左辺を省略すると文字列の末尾とみなされる:

In [ ]:
```
alphabet = "abcdefghijklmnopqrstuvwxyz"
print(alphabet[0], alphabet[1])   # a, b
print(alphabet[-1], alphabet[-2]) # z, y

print(alphabet[0:3])              # abd
print(alphabet[-6:-3])            # uvw

print(alphabet[:3])               # abc
print(alphabet[-3:])              # xyz

print(alphabet[3:0])              # 逆順は空文字列となる
```

ただし、インデックス付きの文字列変数に代入するといった方法で文字列を変更することはできない:

In [ ]:
```
alphabet = "abcdefghijklmnopqrstuvwxyz"
alphabet[0]="A"                          # intend to change 1st ch
aracter
```

このような場合は、文字列を分解してリスト(後述)化し、その一部を置き換え、再結合する:

In [ ]:
```
alphabet = "abcdefghijklmnopqrstuvwxyz"
list_alpha = list(alphabet)              # ["a", "b", "c",....]
list_alpha[0] = "A"                      # ["a" -> "A", "b", "c",.
...]
"".join(list_alpha)
```

練習問題(文字列型)

文字列のインデックスとして [:]が与えられたとき、上のalphabet のどの範囲が得られるか予想する。 実際にセルで試して、予想と合っているか確認する。

In [ ]:
```
alphabet[:]
```


`str.format()` による整形

文字列を整形する `format()`メソッドが用意されている。 メソッドを呼び出す文字列には`{}` で囲われた置換フィールドを含む。 置換フィールドの数字は引数の順番を示す。 インデックスに続く `:`以降に文字列の長さ、型を指定することもできる:

In [4]:
```
#引数だけを指定
print('Apple:{0}, Orange:{1}'.format(10,20))

#次行は先頭の引数は浮動小数点、小数点以下2桁、2番目は文字列、3番目は整数.... とする書式
complex_format = '{0:.2f} {1:s} {2:d} {3:s}'
btc_eu = 7384.5
jp_eu = 0.0074
print(complex_format.format(btc_eu / jp_eu, 'JPY', 1, 'BTC'))
```

```
Apple:10, Orange:20
997905.41 JPY 1 BTC
```

型変換

これまで紹介したスカラー型(数、ブーリアン、文字列)は以下のようにおこなえる。 ここで、`type()` は変数の型を返す関数:

In [8]:
```
sval = '3.1415'    ; print(type(sval), sval)
```

```
<class 'str'> 3.1415
```

In [9]:
```
fval = float(sval) ; print(type(fval), sval)
```

```
<class 'float'> 3.1415
```

In [10]:
```
s2val = str(fval)  ; print(type(s2val), s2val)
```

```
<class 'str'> 3.1415
```

In [11]:
```
val = int(fval)    ; print(type(val), val)
```

```
<class 'int'> 3
```

In [12]:
```
val = bool(fval)   ; print(type(val), val)
```

```
<class 'bool'> True
```

In [13]:
```
cval = complex(fval); print(type(cval), cval)
```

```
<class 'complex'> (3.1415+0j)
```


日付・時刻

基本型ではないが、datetimeモジュールは、日付・時刻をあつかうdatetime, date, time　型を提供する。
日付・時刻には様々な書式がある、多様な書式に対応するには datetime.strptime(文字列,書式) が有用である。
詳細についてはリファレンスマニュアル (https://docs.python.jp/3/library/datetime.html)を参照のこと:

In [ ]:
```
from datetime import datetime, date, time
dt = datetime(2018,4,1, 23, 50, 20)

print(dt.date())                                        # 日付部分のみとりだす
print(dt.time())                                        # 時刻部分のみとりだす

                                                        # 同じ日時だが書式が違う、コンピュータの地域設定によってはエラーになるかもしれない
print(datetime.strptime(
    "2018/4/1 23:50:20", "%Y/%m/%d %H:%M:%S"))
print(datetime.strptime(
    "11PM 50:20 April 1, 2018", "%I%p %M:%S %B %d, %Y"))
```

