CHaserのためのPython入門 基礎編

これは、U-16札幌プロコン事前講習会に向けて、メンター向けにPythonの基礎を示すドキュメントです。

用意するもの

- PC
- 『WindowsパソコンでCHaserを動かすまで』で用意したUSBメモリ

基本編

Pythonプログラムの実行方法は、主に2通りあります。

対話型シェルによる実行

対話型シェルとは、その名の通り「対話をしているようにPythonのプログラムを実行できる機能」のことです。キーボードを使ってPythonのプログラムをその場で打ち込み、結果を確認できます。

対話型シェルを立ち上げるには、USBメモリのWinPythonフォルダ内の「WinPython Command Prompt」を起動したあと、コマンドプロンプト（黒いコマンド入力画面）に「python」と入力しEnterキーを押下します。

>>> と書いてあるところにプログラムを入力すると、その場でプログラムが実行されます。
これは、ちょっとプログラムの動きを確認したいときに便利です。

ファイルからの実行

対話型シェルで実行したプログラムは、対話型シェルを終了すると消えてしまいます。CHaserのプログラムのように、一度書いたら残しておきたいようなプログラムはファイルに保存しておく必要があります。

テキストエディタ（今回はVSCode）を起動し、プログラムを書いてファイル名をつけて保存します。プログラムはUSBメモリ内のWinPythonフォルダのscriptsフォルダに保存してください。

ファイルとして保存したPythonプログラムを実行するには、「WinPython Command Prompt」を起動した後、以下のコマンドを入力します。（>記号はプロンプトなので無視してください。）

```
> python ファイル名
```

画面への表示

情報を画面に表示するには、`print()`関数を利用します。

例えば画面に"Hello"と表示するには、

```
print("Hello")
```

というプログラムを書きます。

関数は、コンピュータへの命令のようなものだと思ってください。

コメント

プログラム中には、コメントを加える事ができます。処理の際には無視されるので、注釈や覚え書きとして利用できます。

```
print("Hello Sapporo!") # Hello Sapporo! と表示する
# コメントは無視される
```

文字列

文字列とは、単語や文章のような文字の連なったものです。
文字列は、""で囲って表現します。

```
print("Hello")
print("Hello" + "World") # 文字列の連結（足し算）
print("Hello" * 5)       # 文字列の繰り返し（掛け算）, *（アスタリスク）は×（乗算記号）を意味する
```

数値

数値は、整数や小数などを扱えます。

```
print(22)
print(3.14)
```

四則演算が可能です。なお、乗算器号（×）は*を、除算記号（÷)は/を使うということに注意してください。

```
print(1 + 1)
print(5 - 2)
print(3 * 5)
print(15 / 3)
```

剰余（あまり）は%記号で求めることができます。

```
print(17 % 5)
```

べき乗は**を利用します。

```
print(2 ** 10) # 2の10乗, 1024が表示される。
```

また、演算子には優先順位があるので注意してください。

```
print(1 + 2 * 3)    # 7が表示される
print((1 + 2) * 3)  # 9が表示される
```

変数

文字列や数値などのデータを入れておく箱のようなものが変数です。変数には名前をつけて管理することができます。変数につけた名前を変数名といいます。

変数へデータを入れることを代入と呼びます。代入は以下のようにできます。

```
name = "Taro"
age = 20
weight = 60.5
```

print()関数を利用して変数の中身を表示することも可能です。

```
print(name)                              # Taro
print(age)                               # 20
print("私の名前は", name, "です。")     # 私の名前は Taro です。
```

また、一度作成した変数のデータを変更したり、変数同士で四則演算等することもできます。

```
dept = "人事部"
salary = 210000

dept = "総務部"          # 一度作った変数に違うデータを入れることができる
salary = salary * 1.2 # salaryの1.2倍の値をsalaryに代入（数学の等式とは違うので注意）
```

1ずつ増加・減少させるインクリメント・デクリメントも覚えておいてください。

```
age = 20
age += 1 # インクリメント（age = age + 1と同義）
age -= 1 # デクリメント（age = age - 1と同義）
```

変数名は以下のルールに従ってつけてください。

- 1文字目は英字か_（アンダーバー）
- 2文字目以降は英数字か_
- 予約語（後述）は使用できない
- 大文字と小文字は区別される（aとAは別の変数として扱われる）

予約語とは、Pythonプログラム内で特別な意味を持っている単語の事で、以下のようなものがあります。これらの単語を変数名として使うことはできません。（passwordやmy_classなど、予約語を含んでいる変数名は問題ありません。）

```
False       None        True        and
as          assert      break       class
continue    def         del         elif
else        except      finally     for
from        global      if          import
in          is          lambda      nonlocal
not         or          pass        raise
return      try         while       with
yield
```

また、エラー等は出ませんが実用上不都合がある変数名としては、以下のものがありますので、避けた方が良いでしょう。

- 先頭にアンダーバーがついた変数名 _var など
- 大文字で始まる変数名 Animal など

条件分岐

プログラムは上から順に実行されますが、条件によって実行する処理を変えたいときがあります。
そんな時には条件分岐を使います。

条件によって処理を分岐させる方法として、「if文」があります。

if文の書式は以下のとおりです。インデント（Tabキーを押下する）に注意してください。

```
if 条件式:
    処理
```

if文はこのように使えます。

```
age = 26

if age >= 25:
    # ここは実行される
    print("衆議院議員に立候補できます。")

if age >= 30:
    # ここは実行されない
    print("参議院議員に立候補できます。")
```

また、else句を用いて条件を満たさない時の処理を書くこともできます。

```
age = 19

if age >= 20:
    print("成人しています。")
else:
    print("成人していません。")
```

また、複雑な条件を処理するためにelif句を使うこともできます。ifが「もし、〜なれば」、elseが「そうでなければ、」とすると、elifは、「そうでなくてもし〜ならば」というイメージです。

```
a = 0

if a > 0:
    print("aは正の数です")
elif a < 0:
    print("aは負の数です")
else:
    print("aはゼロです")
```

条件式を記述するには、次のような比較演算子や論理演算子を使います。

比較演算子は以下のとおりです。

比較演算子	記述例	意味
==	a == b	aとbが等しい
!=	a != b	aとbが等しくない
>	a > b	bよりaが大きい
>=	a >= b	bよりaが大きいか、等しい
<	a < b	bよりaが小さい

<= a <= b bよりaが小さいか、等しい

論理演算子は以下のとおりです。

論理演算子	記述例	意味
and	a and b	aとbが共に成り立つ時
or	a or b	aまたはb少なくともどちらかが成り立つ時
not	not a	aが成り立たない時

リスト

複数のデータを一つにまとめたものをリストといいます。リストのひとつひとつのデータをリストの要素と呼びます。

リストは、以下のように作成します。

```
a = [1, 2, 3, 4, 5]
```

リストのある要素を参照するには、リスト名[n]というようにします。なお、Pythonにおけるリストの要素は0番目から数え始めることに注意してください。

```
b = [1, 2, 3]
print(b[0])
```

リストのサイズ（要素数）は、len()で取得できます。

```
print(len(b)) # 3
```

通常の変数同様、リスト全体または特定の要素を書き換えたり、追加することができます。

```
c = [1, 2, 3]
print(c) # [1, 2, 3]
c[0] = 5
print(c) # [5, 2, 3]

c.append(10) # リストの最後に10という要素を追加
print(c) # [5, 2, 3, 10]
```

繰り返し

決まった回数繰り返したい処理がある場合はfor文を、ある条件を満たしている間繰り返したい処理がある場合はwhile文を使います。

for文は以下のような書式です。for文は、リストの要素数のぶんだけ処理を繰り返したい時に便利です。

```
# ここでいうオブジェクトは、リスト変数等を示す。
for 変数 in オブジェクト:
    処理
```

例えば、果物の名前が入ったリスト型変数があり、その要素ひとつひとつを表示したい場合は以下のようなコードを記述します。

```
fruits = ["リンゴ", "なし", "バナナ"]
for item in fruits:
    print(item)
# リンゴ、なし、バナナが順に表示される。
```

上記のプログラムの処理の流れは以下のとおりです。

```
1) fruitsの0番目の要素"リンゴ"がitemに代入される
2) print(item)
3) fruitsの1番目の要素"なし"がitemに代入される
4) print(item)
5) fruitsの2番目の要素"バナナ"がitemに代入される
6) print(item)
```

また単にある回数だけ処理を繰り返したい時は、以下のようにrange()関数を利用します。

```
for i in range(10):
    print("Hello", i, "回目")
# Hello 0回目
# Hello 1回目
# Hello 2回目
#       .
#       .
#       .
```

次に、while文の説明です。while文の書式は以下のとおりです。

```
while 条件式:
    処理
```

例えば、ある変数の値が10以内の時だけ数値を表示するプログラムは以下のとおりです。

```
count = 0
while count < 10:
    print(count)
    count += 1
```自分のCHaserプログラムをつくる
===

# はじめに

自分のChaserプログラムを作ってみましょう。


# CHaserライブラリの命令

各命令を実行すると、value変数に周辺情報が格納されます。詳しくは、CHaser公式の以下ドキュメントを参照してください。

http://www.zenjouken.com/?action=common_download_main&upload_id=522


## get_ready()

ターンのはじめに実行する。value変数には周辺情報が格納されます。

```python
value = client.get_ready()
```

walk○○

指定された上下左右のどれかの方向に移動します。

```
value = client.walk_up()
value = client.walk_down()
value = client.walk_left()
value = client.walk_right()
```

look○○

指定された上下左右のどれかの方向に対して、正方形状に9マスの情報を取得します。value変数にはその9マスの情報が格納されます。

```
value = client.look_up()
value = client.look_down()
value = client.look_left()
value = client.look_right()
```

search○○

指定された上下左右のどれかの方向に対して、直線上に9マスの情報を取得する。value変数にはその9マスの情報が格納されます。

```
value = client.search_up()
value = client.search_down()
value = client.search_left()
value = client.search_right()
```

put○○

指定された上下左右のどれかの方向に対して、ブロックを配置する。value変数には周囲9マスの情報が格納されます。

```
value = client.put_up()
value = client.put_down()
value = client.put_left()
value = client.put_right()
```

サンプルのCHaserプログラムを改良する

USBメモリに入っているsample.pyをmychaser.pyという名前でコピーしましょう。（生徒さんには、自分の名前のファイル名を付けてもらったほうがわかりやすいかもしれません。takuya.pyなど。）

プログラムの改良方法

sampleのプログラムは、コメント部を除くと以下のようになっています。

```
import CHaser

def main():
    value = []
    client = CHaser.Client()

    while(True):
        value = client.get_ready()
        value = client.search_left()

        value = client.get_ready()
        if value[7] != 2:
            value = client.walk_down()
        else:
            value = client.put_up()

        value = client.get_ready()
        value = client.look_up()

        value = client.get_ready()
        value = client.put_right()
```

`while(True):`以降が改良するべき部分です。

プログラムを改良する際は、client.get_ready()を呼び出してからなにかの動作（ `client.walk_down()` 等）をすることに注意してください。

では早速、あなたのCHaserプログラムを改良してみましょう。

ランダムに動くようにする

現状だと、このプログラムは下方向に動き続けるのみです。そこで、上下左右いずれかの方向にランダムに動くようにしてみましょう。

0から3までの数値をランダムに出し、0なら上へ、1なら下へ、2なら左へ、3なら右へ移動するようにしましょう。

Pythonでランダムな数値を出すには、以下のようにプログラムを書きます。

```
import random

random.randint(0, 3) # 0から3までのランダムな数値を出す
```

これをCHaserプログラムに組み込んでみましょう。なお、import文はプログラムの最初にまとめて書くようにします。

```
import CHaser
import random # import文はプログラムの最初に書く

def main():
    value = []
    client = CHaser.Client()

    while(True):
        value = client.get_ready()
        value = client.search_left()

        value = client.get_ready()

        number = random.randint(0, 3)
        if number == 0:
            client.walk_up()
        elif number == 1:
            client.walk_down()
        elif number == 2:
            client.walk_left()
        elif number == 3:
            client.walk_right()


# 以下の部分はコメント化
#         if value[7] != 2:
#             value = client.walk_down()
#         else:
#             value = client.put_up()

#         value = client.get_ready()
#         value = client.look_up()

#         value = client.get_ready()
#         value = client.put_right()
```

これで、上下左右ランダムな方向に動き続けるCHaserができました。

しかしこれだと、これだと、ブロックに突っ込んで負けになってしまうので、進行したい方向にブロックがあったときは進まないような処理が必要です。

value変数の値と条件分岐をうまく利用して、ブロックがある時は別の行動（ただ単にlookやsearchするだけなど）を取るようプログラムを改良する方法を生徒と一緒に考えてよいでしょう。
CHaserの動かし方
===

はじめに

CHaserのサーバーやプログラムを動かして、ゲームを実施する方法をまとめたドキュメントです。

用意するもの

- PC
- 『WindowsパソコンでCHaserを動かすまで』で用意したUSBメモリ
- 自分のCHaserプログラム（ここではmychaser.pyとしますが、適宜読み替えてください。）

手順

サーバの起動

USBメモリ内に入っているサーバプログラムを起動します。

COOLとHOTどちらも生徒の作ったプログラムをつなぐ場合はそのままCOOLとHOTの接続開始ボタンを押します。
自分の作ったプログラムをテストしてみたい時はCOOLかHOTどちらかの"TCPユーザ"をクリックして"自動くん"にします。

プログラムの接続

USBメモリ内のPythonCommandPromptを開いて、以下のコマンドを入力します。なお、>はプロンプトなので入力しないでください。

```
> cd CHaser
> python mychaser.py
```

エラー等出なければ、接続情報等を聞かれるので入力してください。

```
ポート番号を入力してください → 2009  （COOLなら2009を、HOTなら2010を入力）
ユーザー名を入力してください → takuya  （生徒の名前等適当に）
サーバーのIPアドレスを入力してください → [IPアドレス]
```

IPアドレスは、

- サーバを起動したパソコンとCHaserプログラムを起動したパソコンが同一の場合は 127.0.0.1
- サーバを起動したパソコンとCHaserプログラムを起動したパソコンが別の場合はサーバプログラムのSERVER欄に書いてあるIPアドレス

をそれぞれ入力してください。

ゲーム開始

COOL側、HOT側どちらも準備完了になったら、ゲーム開始ボタンを押してください。

ゲーム終了

サーバプログラムの右上のXボタンを押してサーバプログラムを終了してください。
再度ゲームをする場合は、サーバの起動からやり直してください。