Python基礎研修
Python Basic Training
6.モジュールとパッケージ　Module and Package
7.複合データ型　Composite data type

研修概要

マッチスル

研修概要

- ✓ 本研修は、これからPythonを使ったプログラミングを始めたい方を対象としています。
- ✓ Pythonによる基本的な構文に加え、関数やモジュールの使い方なども学習します。

前提知識

- ✓ プログラミングの基礎知識を有していること。

研修目標

- ✓ Pythonによる、条件分岐や反復などの基本的な構文について理解すること。
- ✓ Pythonによる、関数の定義や利用、モジュールの利用について理解すること。

目次

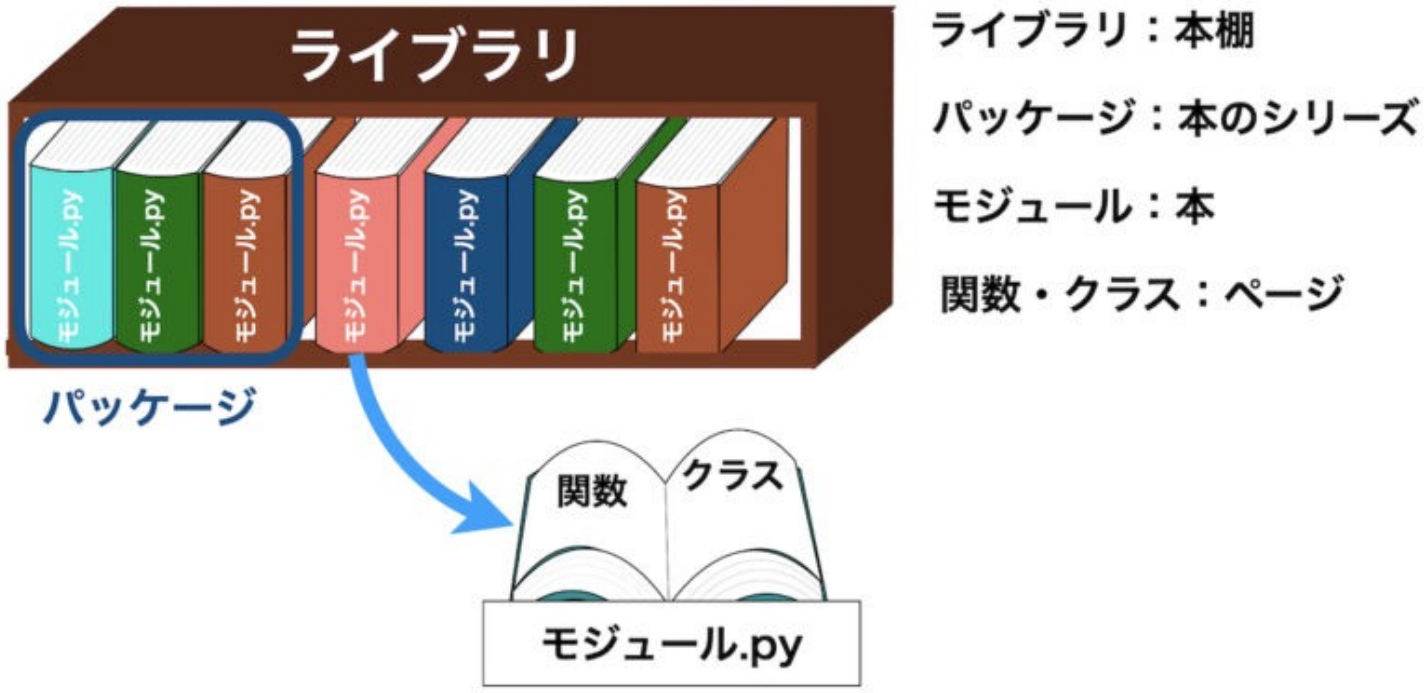
マッチスル

6.モジュールとパッケージ
Module and Package

章の目次

モジュールとパッケージ

- ✓ モジュールとは
- ✓ モジュールのインポート
  - ✓ 標準モジュールのインポート
  - ✓ 必要な箇所のみのインポート
  - ✓ 外部モジュールのインポート
- ✓ パッケージとは
- ✓ パッケージのインポート
- ✓ 練習問題



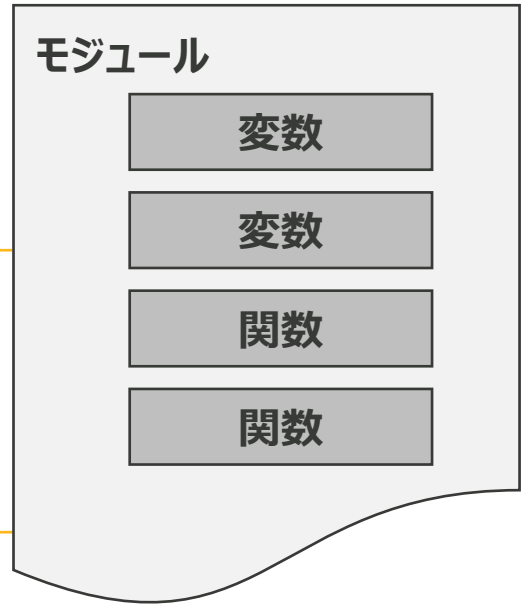
https://programming-surgeon.com/lecture/library-module/

マッチスル

モジュールとは

- ✓ モジュールは.pyファイルで、プログラムを構成する関数やクラスの集まりのこと。
- ✓ モジュールは、大きく分けて以下の二種類に分類することが可能。

標準モジュール	ビルトインモジュールとも呼ばれ、初めから組み込まれたモジュールのこと。インポートすればそのまま利用できる点が特徴。標準モジュールの種類や機能は公式のライブラリリファレンスにすべて掲載されている。 例：os.py→osの機能をインポートimport os→自分のいるフォルダの場所を出力（current working directory）os.getcwd()
外部モジュール	公式に提供されるモジュール以外を指し、**これまで自身が作成したプログラム（Pythonファイル）もすべて外部モジュールに該当する。**その他、団体や有志によって作成され、公開されているモジュールも含まれる。 例：numpy, matplotlib, pandasなど→インストールが必要、またはanacondaやgoogle collaboratoryを使う→import pandas as pd →csvファイル読み込みpd.read_csv(“file.csv”)



モジュールの活用

- ✓ あるモジュールから、別のモジュールの機能をインポートする（取り込む）ことが可能。
- ✓ インポートにより、モジュールを機能単位に分割すること、再利用することなどが可能。

モジュールのインポート

マッチスル

標準モジュールのインポート

- ✓ 利用したいモジュールのインポートを行うには、 <import> ステートメントを使用する。
- ✓ インポートしたモジュールの変数や関数を参照する際は、ドット記号で結んで記述する。

```
import モジュール名      # モジュールをインポート

モジュール名.変数名      # インポートした変数を参照
モジュール名.関数名      # インポートした関数を参照
```

module_import_builtin.py

```
01: import random
02:
03: week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]
04:
05: day = random.choice(week)
06: print(day)
07:
08: print(str(random.randint(1,100)))
```

random.choice ：複合データからランダムに選択する関数
random.randint：指定した間にあるランダムな整数を返却する関数

マッチスル

必要な箇所のみのインポート

- ✓ モジュールから利用したい部品（変数や関数）のみインポートすることも可能。
- ✓ 必要な箇所のみのインポートを行うには、<from>ステートメントを使用する。

```
from モジュール名 import 変数名
from モジュール名 import 関数名
```

module_import_necessary.py

```
01: from random import choice, randint
02:
03: week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]
04:
05: day = choice(week)
06: print(day)
07:
08: print(str(randint(1,100)))
```

マッチスル

外部モジュールのインポート（1/2）

- ✓ 自身が作成したプログラム（Pythonファイル）もインポート可能。
- ✓ ファイル名から拡張子（.py）を除いたものがモジュール名となる。

module_import_myself.py

```
01:
02: import def_argument1
03:
04:
05: def_argument1.print_apple(100, "Green")
06:
```

モジュール名として、拡張子を除いたファイル名を指定。

自身で作成した print_apple 関数を呼び出す。

def_argument1.py

```
01:
02: def print_apple(weight, color):
03:     print("This apple weighs " + str(weight) + ".")
04:     print("The color of this apple is " + color + ".")
05:
06: print_apple(100, "Red")
```

module_myself_import.py から呼び出される print_apple 関数。

マッチスル

外部モジュールのインポート（2/2）

- ✓ importステートメントでは、実際にはモジュールを一度実行している。
- ✓ そのため、前スライドのソースコードの実行結果は以下の通りとなる。

■ 選択コマンドプロンプト

```
C:¥python>python module_myself_import.py
This apple weighs 100.
The color of this apple is Red.
This apple weighs 100.
The color of this apple is Green.
```

def_argument1.py の6行目による出力
✓ 本来、意図していない出力？

module_myself_import.py の5行目による出力
✓ 本来、意図した通りの出力

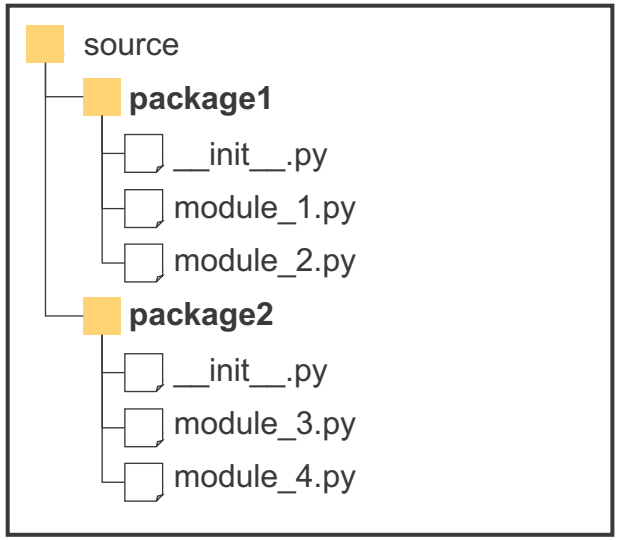
- ✓ 将来、インポートする可能性のあるモジュールは、ソースコード内部の構成も大切となる。
- ✓ 外部モジュールをインポートして利用する際は、必要なモジュールが揃っていること前提。
- ✓ そのため、他者が作成したものを使用する際は、ダウンロードもしくはインストールが必要なケースもある点に注意。

マッチスル

パッケージとは

- ✓ パッケージとは、複数のモジュールを目的毎に整理して階層化した"箱"のこと。
- ✓ 慣例的に、パッケージディレクトリには＜__init__.py＞ファイルを配置する。

モジュール ▶ **パッケージ** ▶ **ライブラリ**



代表的な標準ライブラリ

- ✓ math
- ✓ random
- ✓ datetime

代表的な外部ライブラリ

- ✓ Numpy
- ✓ Pandas
- ✓ Pillow
- ✓ Openpyxl

マッチスル

パッケージのインポート

- ✓ パッケージのインポートを行うには、同様に <import> ステートメントを使用する。
- ✓ パッケージ内のモジュールを指定してインポートするには、ドット記号で結んで記述する。
- ✓ インポートしたモジュールの関数や変数を参照する際も、ドット記号で結んで記述する。

```
import パッケージ名.モジュール名          # パッケージ配下のモジュールをインポート

パッケージ名.モジュール名.変数名          # インポートした変数を参照
パッケージ名.モジュール名.関数名          # インポートした関数を参照
```

- ✓ インポートする際の名称は、<as> キーワードにより別名を付与すること可能。
- ✓ インポートしたモジュールの関数や変数も、別名を用いて参照することが可能。

```
import パッケージ名.モジュール名 as 別名    # パッケージ配下のモジュールを別名でインポート

別名.関数名                               # インポートした関数を別名を使って参照
```

章のまとめ

マッチスル

モジュールとパッケージ

モジュールとは

- ✓ プログラムを構成する関数などの集まりのことで、他のモジュールの機能をインポートして利用することが可能。
- ✓ モジュールは、公式に組み込まれて提供される標準モジュールと、それ以外の外部モジュールに分類される。

標準モジュールのインポート

- ✓ 利用したいモジュールのインポートを行うには、＜import＞ステートメントを使用する。
- ✓ インポートしたモジュールの関数などを参照する際は、ドット記号で結んで記述する。

外部モジュールのインポート

- ✓ 自身が作成したプログラムも外部モジュールの一種であり、同様にインポートが可能。
- ✓ ファイル名から拡張子（.py）を除いたものがモジュール名となる。

パッケージとは

- ✓ パッケージとは、複数のモジュールを目的毎に整理して階層化した箱にあたるもの。
- ✓ 複数のパッケージをまとめたものがライブラリであり、様々な種類が提供されている。

マッチスル

7.複合データ型
Composite data type

章の目次

マッチスル

複合データ型

- ✓ 複合データ型とは
- ✓ リスト
- ✓ ディクショナリ
- ✓ タプル
- ✓ 複合データ型と比較演算子
- ✓ 練習問題

マッチスル

複合データ型とは

- ✓ 複数のオブジェクトを内包し、ひとつにまとめたデータ型。
- ✓ 代表的な複合データ型として、以下の三種類が存在。
  - ✓ リスト　　（list）
  - ✓ ディクショナリ（Dictionary）
  - ✓ タプル　　（tuple）

-

```
01: # リストの例
02: l = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]
03:
04: # ディクショナリの例
05: d = {"Sun":"Sunday", "Mon":"Monday", "Tue":"Tuesday", "Wed":"Wednesday", "Thu":"Thursday", "Fri":"Friday",
"Sat":"Saturday"}
06:
07: # タプルの例
08: t = ("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
09: print(l)
10: print(d)
11: print(t)
```

リスト（1/3）

リスト

- ✓ 複数の要素を順番に並べて格納した、最も基本的な複合データ型。
- ✓ 他のプログラミング言語における、配列（Array）と似た機能を持つ。

リストの特徴

- ✓ 要素は順序性が保証されており、インデックスを用いてアクセス可能。
- ✓ 要素は重複した値の格納や、数値や文字列を混在しても構わない。
- ✓ データ型は **<list型>** として扱われる。

リスト（2/3）

リストの記述方法

- ✓ カンマで区切られた要素を、**＜大括弧＞**で囲む形式で記述。
- ✓ リスト全体を変数に代入し、使用することが可能。

```
[要素1，要素2，要素3]

変数名 = [要素1，要素2，要素3]
```

リストの参照方法

- ✓ リストのインデックスを、**＜大括弧＞**で指定して参照。

```
変数名[インデックス]

week[3]    # 記述例：list型変数weekの3番目の要素を参照
```

マッチスル

リスト（3/3）

リストの操作の例

comp_list.py

```
01: week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
02:
03: print(week[0])            # 要素番号0の要素を取得
04: print(week[2])            # 要素番号2の要素を取得
05: print(week[2:5])          # 要素番号2から4までの要素を取得（要素番号5は含まない点に注意）
06: print(len(week))          # len関数：要素数を取得
07:
08: week[0] = "Holiday"       # 要素番号0の要素を"Holiday"に更新
09: week.append("Saturday")   # append関数：リストの一番最後に新しい要素"Saturday"を追加
10:
11: del week[1]               # del文：要素番号1の要素を削除
12: week.remove("Tuesday")    # remove関数："Tuesday"に一致する最初の要素を削除
13:
14: for day in week:
15:     print(day)
16:
17:
```

マッチスル

ディクショナリ（1/3）

ディクショナリ

- ✓ キー値（Key）と値（Value）の対応によって格納した複合データ型。
- ✓ 他のプログラミング言語における、ハッシュや連想配列と似た機能を持つ。

ディクショナリの特徴

- ✓ 要素の順序性はなく、キー値（Key）を元に値（Value）へアクセス可能。
- ✓ キー値はひとつのディクショナリ内で一意に識別できる必要があり、重複不可。
- ✓ データ型は**<dict型>**として扱われる。

マッチスル

ディクショナリ（2/3）

ディクショナリの記述方法

- ✓ キー値と要素の対応をコロンで表現し、カンマで区切って列挙し、**＜中括弧＞**で囲む形式で記述。
- ✓ ディクショナリ全体を変数に代入し、使用することが可能。

```
{Key:Value, Key:Value, Key:Value}

変数名 = {Key:Value, Key:Value, Key:Value}
```

ディクショナリの参照方法

- ✓ ディクショナリのキー値を、**＜大括弧＞**で指定して参照。

```
変数名[Key]

week["Sat"]    # 記述例：dict型変数weekの、キー値が"Sat"の要素を参照
```

マッチスル

ディクショナリ（3/3）

ディクショナリの操作の例

comp_dictionary.py

```
01: week = {"Sun":"Sunday", "Mon":"Monday", "Tue":"Tuesday", "Wed":"Wednesday", "Thu":"Thursday", "Fri":"Friday"}
02:
03: print(week["Sun"])        # キー値が"Sun"の要素を取得
04: print(week["Tue"])        # キー値が"Tue"の要素を取得
05: print(len(week))          # len関数：要素数を取得
06:
07: week["Sun"] = "Holiday"   # キー値が"Sun"の要素を"Holiday"に更新
08: week["Sat"] = "Saturday"  # ディクショナリに新しいキー値"Sat"と要素"Saturday"を追加
09:
10: del week["Mon"]           # del文：キー値が"Mon"の要素を削除
11:
12: for day in week.keys():
13:     print(day)
14:
15: for day in week.values():
16:     print(day)
17:
```

タプル（1/2）

タプル

- ✓ 複数の要素を順番に並べて格納した複合データ型。
- ✓ 通常のリストとは異なり、一度定義すると編集不可。

タプルの特徴

- ✓ 要素は順序性が保証されており、インデックスを用いてアクセス可能。
- ✓ 編集不可である特徴を利用し、言語実装内など特殊な場面で用いられることが多い。
- ✓ 通常、複合データ型を利用する際は、リストまたはディクショナリを選択するのが一般的。
- ✓ データ型は **<tuple型>** として扱われる。

マッチスル

タプル（2/2）

タプルの記述方法

- ✓ カンマで区切られた要素を、**＜小括弧＞**で囲む形式で記述。
- ✓ タプル全体を変数に代入し、使用することが可能。

```
(要素1，要素2，要素3)

変数名 = (要素1，要素2，要素3)
```

タプルの参照方法

- ✓ タプルのインデックスを、**＜大括弧＞**で指定して参照。

```
変数名[インデックス]

week[3]    # 記述例：tuple型変数weekの3番目の要素を参照
```

マッチスル

複合データ型と比較演算子

- ✓ 複合データ型のメンバーとして、存在するかどうかを確認するための演算子。
- ✓ ディクショナリのキー値を対象に確認したい場合は、valuesメソッドを使用。
- ✓ 複合データ型の比較演算子として、以下の二種類が存在。
  - ✓ **in** ：要素やキー値が含まれているかどうかを確認
  - ✓ **not in**：要素やキー値が含まれていないかどうかを確認

comp_in.py

```
01: list_week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
02: print("Sunday" in list_week)            # 変数list_weekの要素に"Sunday"が含まれるかどうか    → True
03: print("Holiday" not in list_week)       # 変数list_weekの要素に"Holiday"が含まれないかどうか → True
04:
05: dict_week = {"Sun":"Sunday", "Mon":"Monday", "Tue":"Tuesday", "Wed":"Wednesday", "Thu":"Thursday", "Fri":"Friday"}
06: print("Sunday" in dict_week)            # 変数dict_weekのキー値に"Sunday"が含まれるかどうか  → False
07: print("Sunday" in dict_week.values())   # 変数dict_weekの要素に"Sunday"が含まれるかどうか    → True
08: print("Sun" not in dict_week)           # 変数dict_weekのキー値に"Sun"が含まれないかどうか   → False
```

章のまとめ

マッチスル

複合データ型

複合データ型とは

- ✓ 複数のオブジェクトを内包し、ひとつにまとめたデータ型。
- ✓ 代表的な複合データ型として、＜リスト＞ ＜ディクショナリ＞ ＜タプル＞ が存在。

リスト（list / list型）

- ✓ 複数の要素を順番に並べて格納した、最も基本的な複合データ型。
- ✓ 要素は順序性が保証されており、インデックスを用いてアクセス可能。

ディクショナリ（dictionary / dict型）

- ✓ キー値（Key）と値（Value）の対応によって格納した複合データ型。
- ✓ 要素の順序性はなく、キー値（Key）を元に値（Value）へアクセス可能。

タプル（tuple / tuple型）

- ✓ 複数の要素を順番に並べて格納した複合データ型だが、一度定義すると編集不可。
- ✓ 編集不可であることを利用し、言語実装内など特殊な場面で用いられることが多い。