Python基礎研修
Python Basic Training

5.関数の活用

5.1 関数の基礎

5.2 関数の定義と利用

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

Python基礎研修の概要

1.基礎知識

2.実行環境の準備

3.Pythonの基本

1. 基本データ型
2. 変数と演算子

4.制御構文

1. 条件分岐
2. 反復処理（ループ）

5.関数の活用

1. 関数の基礎
2. 関数の定義と利用

6.モジュールとパッケージの活用

7.（付録）複合データ型

マッチスル

5.1 関数の基礎

Basic knowledge of Functions

章の目次

マッチスル

関数の基礎

- ✓ 関数とは
- ✓ print関数
- ✓ input関数
- ✓ len関数
- ✓ 対話型インタプリタによる実行

マッチスル

関数とは

- ✓関数とは、特定の処理をまとめたプログラムの部品です。
- ✓ある値（引数）を入力すると、処理が実行され、結果として戻り値が返されます。
- ✓例えば、ExcelのSUM(A1:A10)のように、引数（A1:A10の値）を与えると、合計値（戻り値）が返ります。

関数のイメージ

関数の書式

y = function (x)

戻り値（OUTPUT）　関数名　引数（INPUT）

マッチスル

print関数

デフォルト・標準設定
print(出力するデータ, sep = "", end = "¥n", file=ファイルオブジェクト, flush=False)

- ✓ コンソールに文字列を出力するための関数。
- ✓ **文字列を指定する際は、文字列の前後をダブルクォーテーションで括ること。**
- ✓ Python のソースコードを実行した人に、メッセージを伝えたい場合に使用する。
- ✓ メッセージを受け取ることで、プログラムが正しく動いているかどうかを確認できる。
- ✓ 文字列の末尾に自動的に改行が挿入されるが、引数の設定により抑制することも可能。

```
[5]: 1 print("Hello Python")
Hello Python

[6]: 1 print("Hello ", end="")
Hello

[7]: 1 print("Python")
Python
```

func_print.py

```
01: print("Hello Python")
02:
03: print("Hello ", end="")
04: print("Python")
05:
06:
07:
08:
```

今後、簡単なサンプルプログラムはこのように記述する。
サンプルと同じファイル名で作成し、pythonフォルダに保存すること。
コマンドプロンプトからスクリプトを実行し、実行結果を確認すること。

input関数

input関数

- ✓ input 関数は、キーボードからの入力を受け付けるための関数です。
- ✓ この関数を使用すると、プログラムの実行中にユーザーからの入力を受け取ることができます。
- ✓ 基本的な動作
- ✓ input 関数が呼び出されると、入力待ちの状態 になります。
- ✓ ユーザーがキーボードで入力し、Enterキーを押すと、その入力内容が文字列として返される。
- ✓ 返された文字列を変数に代入しない場合、入力内容は破棄される。

```
[8]: 1 input("What is your name? --> ")

What is your name? -->  Yama
[8]: 'Yama'

[9]: 1 val = input("Where are you from? --> ")
     2 print(val)

Where are you from? -->  Brazil
Brazil
```

func_input.py

```
01: input("What is your name? --> ")
02:
03: val = input("Where are you from? --> ")
04: print(val)
05:
06:
07:
08:
```

val は"変数"と呼ばれるものである。
変数については別の章にて説明する。
変数を指定する際は、ダブルクォーテーションで括ってはいけない。

len関数

マッチスル

len 関数とは？

len 関数は、指定したオブジェクトのサイズ（要素数）を取得する関数 です。
主に 文字列の文字数やリストの要素数 を調べる際に使用されます。

基本的な使い方

text = "Python" print(len(text)) # 文字列の長さを取得 # 出力: 6

numbers = [10, 20, 30, 40]

print(len(numbers)) # リストの要素数を取得 # 出力: 4

```
1 val = len("What is your name? --> ")
2 print(val)
```

```
23
```

```
1 print(len("Where are you from? --> "))
```

```
24
```

```
1 val = len(input("Where are you from? --> "))
2 print(val)
```

```
Where are you from? -->  Yama
4
```

func_len.py

```
01: val = len("What is your name? --> ")
02: print(val)
03:
04: print(len("Where are you from? --> "))
05:
06: val = len(input("Where are you from? --> "))print(val)
07: print(val)
08:
```

ある関数の引数として、別の関数を指定することも可能である。
その場合、内側の関数の戻り値が、外側の関数の引数となる。

コラム：対話型インタプリタによる実行

マッチスル

対話型インタプリタ

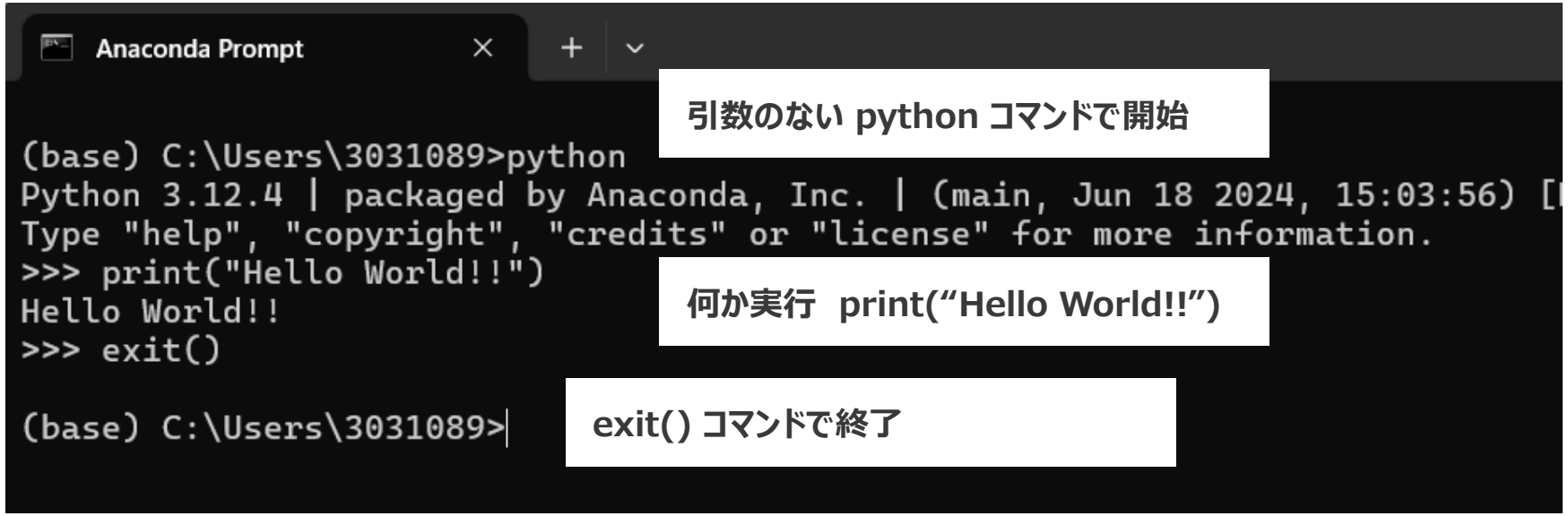
- ✓ コラム：対話型インタプリタによる実行
- ✓ 対話型インタプリタとは？
- ✓ Python には、対話型インタプリタ（Interactive Interpreter） を使用してコードを実行する方法があります。
- ✓ これは、Python ファイルを作成せずに、入力したソースコードを一行ずつ実行できる環境 です。

対話型インタプリタ

- ✓ ✅ メリット
- ✓ 簡単なコードの動作確認 に便利
- ✓ Python の学習やデバッグ に最適
- ✓ インストール直後にすぐ利用可能

対話型インタプリタ

- ✓ ❌ デメリット
- ✓ 複雑なプログラムの作成には向かない
- ✓ 入力したコードは保存されない（必要ならスクリプトファイル .py に記述）



章のまとめ

関数の基礎

関数とは

- ✓ ある値を入力値として与えると、何らかの結果を返すもの。
- ✓ 入力値である＜引数＞と、結果である＜戻り値＞がある。

Print関数

- ✓ コンソールに文字列を出力するための関数。
- ✓ 文字列を指定する際は、その前後をダブルクォーテーションで括ることが大切。

Input関数

- ✓ キーボードからの入力を受け付けるための関数。
- ✓ 関数によって取り込まれた文字列を受け取り、次の処理に繋げることが大切。

len関数

- ✓ 引数に指定されたオブジェクトのサイズを知るための関数。
- ✓ 引数に文字列を指定した場合は、文字数が返却される。

マッチスル

5.2 関数の定義

Function definition

章の目次

関数の定義

- ✓ **関数の定義とは**
- ✓ **関数の定義**
- ✓ **関数の呼び出し**
- ✓ **引数**
  - ✓ 引数の基礎
  - ✓ 引数の注意点
  - ✓ 引数の初期値
- ✓ **戻り値**
  - ✓ 戻り値の基礎
  - ✓ 戻り値の応用
- ✓ **練習問題**

マッチスル

関数の定義とは

- ある値（引数＝道具）を入力値として与えると、何らかの値（戻り値＝結果）を返すもの。
- Excelにて使用される関数と同じ考え方。
- 引数や戻り値を必要としない場合もある。
- **本章では、関数（function）の中で何を行わせるかを自身で定義する。**

関数のイメージ

関数の書式

関数の役割と定義・記述

マッチスル

関数の役割

- ✓ ①ある一連の複雑な処理や長い処理をまとめておきたい場合に、独自の関数を定義（作成）することが可能。プログラム全体が見やすくなる。例：AIの機械学習モデル、深層学習モデル、統計解析手法など
- ✓ ②何度も利用する処理は関数化し、必要な箇所で都度呼び出して使用するのが効果的。

関数の定義・記述

- ✓ ソースコードは上から順に処理するが、**定義した関数は呼び出されるまで実行されない。**
- ✓ 関数を定義する際は、**<def>** ステートメントを使用する。
- ✓ 関数内で実行したい処理を、コードブロック内に記述する。処理は、インデントを入れる（字下げ）。

```
def 関数名():
    実行したい処理
```

マッチスル

関数の呼び出し

- ✓ 定義した関数は、関数名と小括弧を記述して呼び出すことが可能。
- ✓ まずは、引数および戻り値のない関数の呼び出しについて紹介する。

関数名()

def_print.py

```
01: # 関数の定義
02: def hello_python():
03:     print("Hello ", end="")
04:     print("Python!!")
05:
06: # 関数の呼び出し
07: hello_python()
08:
```

＜定義した関数は呼び出されるまで実行されない＞性質により、
処理される順序は、7行目→2行目→3行目→4行目、となる。

マッチスル

引数の基礎

- ✓ 引数とは、呼び出し元から関数本体へ渡される値のこと。関数のインプット、入力値とも言う。
- ✓ ひとつの関数に対し、複数の引数を設定することが可能。

```
def 関数名(引数①, 引数②, 引数③):
    実行したい処理
```

def_argument1.py

```
01: # 関数の定義
02: def print_apple(weight, color):
03:     print("This apple weighs " + str(weight) + ".")
04:     print("The color of this apple is " + color + ".")
05:
06: # 関数の呼び出し
07: print_apple(100, "Red")
08:
```

マッチスル

引数の注意点

✓ 関数を呼び出す際、引数の数や順序などに注意する必要あり。

def_argument2.py

```
01: # 関数の定義
02: def print_apple(weight, color):
03:     print("This apple weighs " + str(weight) + ".")
04:     print("The color of this apple is " + color + ".")
05:
06: # 関数の呼び出し（エラー）
07: print_apple(100)
08: print_apple(100, "Red", "Japan")
09:
10: # 関数の呼び出し（エラー）
11: print_apple("Red", 100)
12:
13: # 関数の呼び出し
14: print_apple(color="Red", weight=100)
15:
16:
17:
```

呼び出し時の引数の数が異なると、その時点でエラーとなる。

引数は呼び出し時に指定した順に設定される。
そのため、結果的に4行目でデータ型のエラーが発生する。

引数は呼出し時に引数名を指定することが可能。
その場合は、順序が変わっても問題ない。

マッチスル

引数の初期値

- ✓ 呼び出し時に引数が指定されなかった場合に適用する、初期値を設定することが可能。
- ✓ 引数が複数ある場合、初期値を設定する引数群は後方にまとめて定義する必要あり。

```
def 関数名(引数①, 引数②, 引数③ = 初期値)
    実行したい処理
```

def_argument3.py

（ファイルを作成してコマンドプロンプトにて実行する場合）

```
01: # 関数の定義
02: def print_apple(weight, color="Green"):
03:     print("This apple weighs " + str(weight) + ".")
04:     print("The color of this apple is " + color + ".")
05:
06: # 関数の呼び出し
07: print_apple(100)
08: print_apple(100, "Red")
```

呼び出し時に引数を省略した場合、初期値が使用される。
省略しなかった場合、設定された値がそのまま使用される。

マッチスル

戻り値の基礎

- ✓ 戻り値とは、関数から呼び出し元へ返却される値のこと。
- ✓ 戻り値を定義する際は、<return>ステートメントを使用する。

```
def 関数名(引数①, 引数②, 引数③):
    実行したい処理
    return 戻り値
```

def_return1.py

```
01: # 関数の定義
02: def answer_apple(weight):
03:     answer = "This apple weighs " + str(weight) + "."
04:     return answer
05:
06: # 関数の呼び出し
07: msg = answer_apple(100)
08: print(msg)
```

呼び出し元へ返却された戻り値は、変数などに格納することが可能。
呼び出し元で処理しなければ、戻り値は無視されてしまう点に注意。

マッチスル

戻り値の応用

- ✓ 戻り値が呼び出し元へ返却されると、関数の呼び出し部分自体が値に変化。
- ✓ 戻り値を変数に格納できるほか、直接関数の引数として設定することも可能（関数の組み合わせ、関数の入れ子、関数ネストとも言う）。

```
変数名 = 関数名(引数①, 引数②, 引数③))

別の関数( 関数名(引数①, 引数②, 引数③) )
```

def_return2.py

```
01: # 関数の定義
02: def answer_apple(weight):
03:     answer = "This apple weighs " + str(weight) + "."
04:     return answer
05:
06: # 関数の呼び出し
07: print(answer_apple(100))
08:
```

戻り値は、直接別の関数の引数として設定することも可能。
すでに登場した以下のソースコードも、これを利用している。
→ int(input("Please put the weight of apple."))

章のまとめ

マッチスル

関数の定義

関数の定義とは

- ✓ 関数はビルトイン関数（あらかじめ定義された関数）のほか、まとめておきたい一連の処理を独自の関数として定義することが可能。
- ✓ 何度も利用する処理は関数化し、必要な箇所で都度呼び出して使用するのがスマートかつ効果的。

- ✓ 関数を定義する際は、＜def＞ステートメントを使用する。
- ✓ 関数内で実行したい処理を、コードブロック内に記述する。
- ✓ 定義した関数は、関数名と小括弧を記述して呼び出すことが可能。

引数

- ✓ 引数とは、呼び出し元から関数本体へ渡される値のこと。
- ✓ 引数の数や順序、初期値など、いくつかのポイントがある。

戻り値

- ✓ 戻り値とは、関数から呼び出し元へ返却される値のこと。
- ✓ 戻り値を定義する際は、＜return＞ステートメントを使用する。