Python基礎研修
Python Basic Training

3.Pythonの基本

3.1 基本データ型

3.2 変数と演算子

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

3.1 基本データ型
Basic data types

章の目次

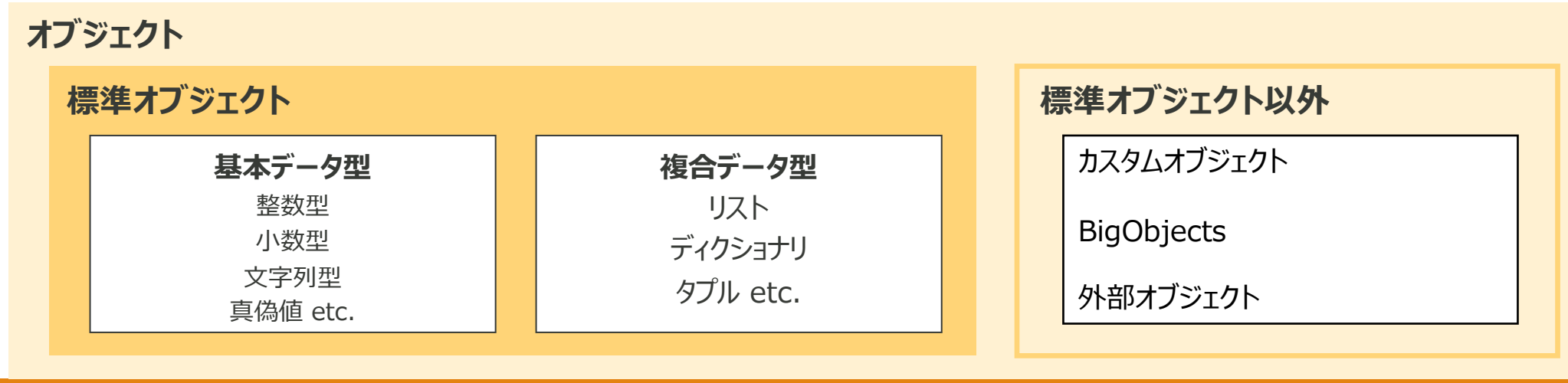
基本データ型とオブジェクト

- ✓ オブジェクト
- ✓ 整数型
- ✓ 小数型
- ✓ 文字列型
- ✓ ブール型と真偽値
- ✓ コラム
  - ✓ コラム①：コメント
  - ✓ コラム②：特殊文字

マッチスル

オブジェクト

- ✓ オブジェクトとは、Pythonで扱うデータや処理対象のことを指す。
- ✓ Pythonでは、すべてのデータがオブジェクトとして表現され、操作される。
- ✓ Pythonが標準で提供するオブジェクトを「標準オブジェクト（ビルトインオブジェクト）」と呼ぶ。
- ✓ 標準オブジェクトの種類
  - ✓ 基本データ型（本章で扱う数値や文字列などのデータ型）
  - ✓ 複合データ型（複数の値をまとめて扱うデータ型）



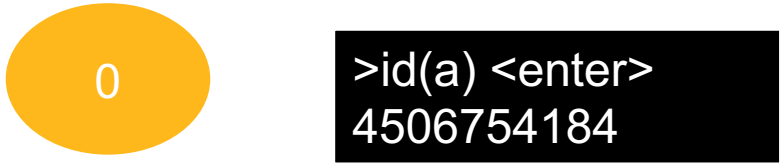
動的型付け
実行時にデータ型が決まる(インタプリタ言語)。実行時までデータ型が分からない。機械語に逐次翻訳されるので実行速度は遅め。

オブジェクトの「データ型」、「値」、コンピュータのメモリ上のアドレス

マッチスル

オブジェクトの「型」と「値」の例：

整数型オブジェクト



浮動小数点型オブジェクト

文字列型オブジェクト

整数型

整数型（int）

Pythonでは、数値をそのまま扱うことができる。その中で、小数を含まない数値は「整数型（int）」と呼ばれる。

整数型の特徴

•基本的な整数演算が可能（加算 +、減算 -、乗算 *、除算 // など）

•整数の範囲に制限がない（メモリが許す限り、大きな数値も扱える）

•異なる進数での表記が可能

- ✓ 8進数　：数値の先頭に"0o"と記述（数字の"0"とアルファベットの"o"）
- ✓ 16進数：数値の先頭に"0x"と記述（数字の"0"とアルファベットの"x"）

basic_int.py

```
01: print(25)
02: print(0o25)
03: print(0x25)
04: print(0xFF)
05:
06:
```

小数型

浮動小数点型（float）

Pythonでは、小数を含む数値を「浮動小数点型（float）」として扱う。

浮動小数点型の特徴

小数を扱う数値型（例：3.14、0.5、-2.7）

数学演算が可能（加算 +、減算 -、乗算 *、除算 / など）

指数表記（科学技術計算で便利） に対応

basic_float.py

```
01: print(2.5)
02: print(2.5e3) #2.5 に10^3=1000を掛ける
03: print(2.5e-3) #2.5を10^3=1000で割ること
04:
05:
06:
07:
08:
```

文字列型

マッチスル

文字列型（str）

- ✓ Pythonで文字列を扱う際、文字列をクォーテーション（' ' または " "）で囲むことで、文字列オブジェクトとして認識されます。
- ✓ Pythonでは、複数の方法で文字列を表現することができ、それぞれに特有の用途があります。

ダブルクォーテーション	最も一般的な記法。
シングルクォーテーション	一般的な記法だが、シングルクォーテーションを文字列として扱いたい場合（例：don't）は特殊文字を使用。
トリプル・ダブルクォーテーション	改行を含む文字列の記法。
トリプル・シングルクォーテーション	改行を含む文字列の記法だが、シングルクォーテーションと同様の注意点がある。

basic_str.py

```
01: print("What is your name?")
02:
03: print("""My name is Guido van Rossum,
04: from the Netherlands.
05: I like Monty Python's Flying Circus.""")
06:
```

ブール型と真偽値

マッチスル

真偽値（Boolean）

真偽値は、True または False のいずれかの値を取るデータ型です。
このデータ型は、以下のような二者択一の情報を表現するのに使用されます：

•「本当か嘘か」

•「成功か失敗か」

•「オンかオフか」

例

プログラミングにおいて、真偽値は特に制御構文（条件分岐や反復処理）で非常に重要です。条件式（if文やwhile文）で、True であれば処理を実行し、False であれば実行をスキップするという制御が行われます。

basic_boolian.py

```
01: print(True)
02: print(False)
03: print(1<2)
04: print(1+1==3)
05:
06:
07:
08:
```

コラム①：コメント

コメント

- ✓ コメントは、プログラムの実行対象とならない文章で、コードの理解を助けたり、他の開発者とのコミュニケーションを円滑にする役割があります。
- ✓ コメントを適切に活用することで、ソースコードが分かりやすく、保守性が向上します。
- ✓ Pythonでは、日本語を使っても問題なくコメントを記述できます。

コメントの書き方

- ✓ # を使用して行コメントを作成し、三重引用符を使用して複数行のコメントを作成します。左側の例：
- ✓ マジックコメントは特定の機能や設定に使用され、例えば文字コードを指定する際に利用されます。

basic_comment.py

```
01: # 名前を聞く
02: print("What is your name?")
03:
04: # 名前・出身・趣味を出力
05: print("""My name is Guido van Rossum,
06: from the Netherlands.
07: I like Monty Python's Flying Circus.""")
08: “””
09: print("Hello!!")
10: print("Hello World!!")
11: “””
12: print(“Hello to World!!”)
```

コラム②：特殊文字

マッチスル

特殊文字

- ✓ 特殊文字とは、改行やタブなど出力するための特別な文字のこと。
- ✓ 代表的な特殊文字は以下の通り。

改行	¥n	**¥（エンマーク）**	¥¥
ダブルクォーテーション	¥"	**水平タブ**	¥t
シングルクォーテーション	¥'	**垂直タブ**	¥v

basic_escape.py

```
01: # 名前を聞く
02: print("What is your name?")
03:
04: print("My name is Guido van Rossum.")          # 名前を出力
05: print("¥n")
06: print("I'm from the ¥"Netherlands¥".")         # 出身を出力
07: print("¥n")
08: print("I like Monty Python's Flying Circus.")  # 趣味を出力
```

章のまとめ

マッチスル

基本データ型

数値型

- ✓ Pythonでは、一般的な数値をそのまま扱うことができる。
- ✓ 小数を含まないものを＜整数型＞、小数を含むものを＜浮動小数点型＞と呼ぶ。

文字列型

- ✓ Pythonでは、一般的に対象の文字列の前後をダブルクォーテーションで括って指定する。
- ✓ 改行や文字としてのダブルクォーテーションなどを出力したい場合、特殊文字を使用する。

真偽値

- ✓ 真偽値とは、＜True＞または＜False＞を表すもの。
- ✓ プログラミングにおいて、制御構文（条件分岐と反復）の条件式として頻繁に登場する。

コメント

- ✓ プログラムの実行対象とはならない文章のことであり、ソースコードを見やすくする効果がある。
- ✓ Pythonでは、シャープ記号 ＜#＞ を記述した部分から行末までがコメントとして識別される。

マッチスル

3.2 変数と演算子

Variables and Operators

章の目次

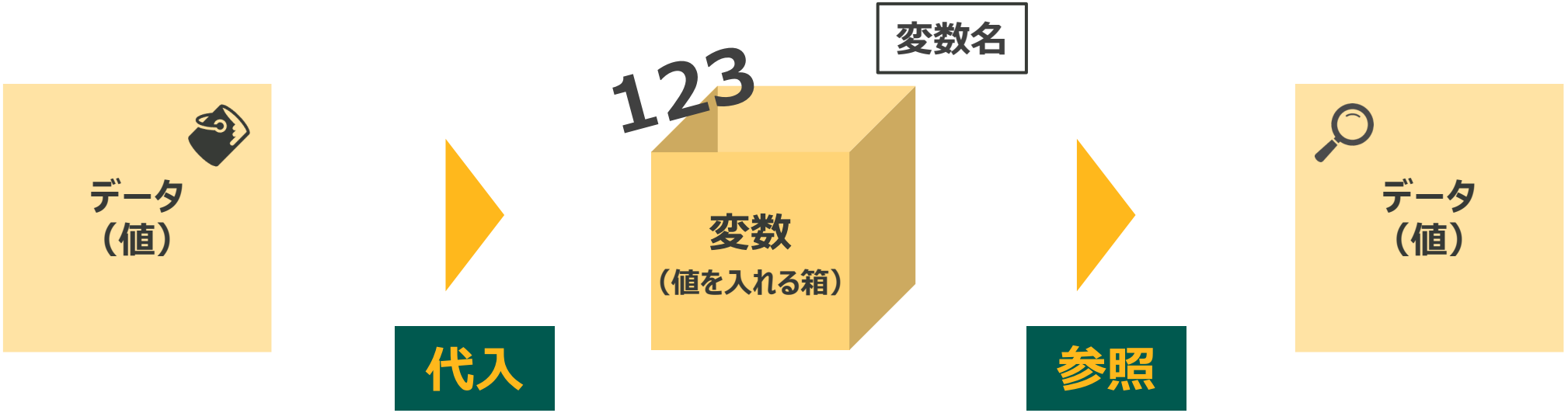
変数と演算子

1. **変数**
    1. 変数の命名規則
    2. 変数の代入
    3. 変数の参照
2. **演算子**
    1. 算術演算子
    2. 代入演算子
    3. 比較演算子
    4. 論理演算子
    5. 文字列結合演算子
3. **コラム**
    1. **コラム①**: 数値と文字列の変換
    2. **コラム②**: 変数のスコープ

マッチスル

変数

- ✓ 変数とは、プログラムで使用する数値や文字列などのデータを一時的に記憶しておく領域のこと。
- ✓ 変数には任意の変数名（他の変数と識別するための名前）を付けることが可能。
- ✓ 変数に値を入れることを＜代入＞、変数から値を取り出すことを＜参照＞と呼ぶ。
- ✓ Pythonでは、変数のデータ型を指定しなくても作成が可能。



マッチスル

変数の命名規則

- ✓ 変数名を付ける際、以下の命名規則に従う必要がある。
  - ✓ 使用できる文字は、アルファベット、数字、アンダースコア（下線）
  - ✓ 但し、先頭の文字に限り、アルファベットまたはアンダースコアのみ
  - ✓ 大文字と小文字は区別される
  - ✓ 全角文字は使用不可
  - ✓ 予約語は使用不可

予約語一覧

and	del	for	is	raise	assert	elif
from	lambda	return	break	else	global	not
try	class	except	if	or	while	continue
exec	import	pass	yield	def	finally	in
print	as	with	nonlocal			

マッチスル

変数の代入

- ✓ 変数へ値を代入する際は、代入演算子 <=> を使用する。
- ✓ 代入文の左辺に変数名、右辺に代入したい値を記述する。

変数の参照

- ✓ 変数に格納されている値を参照する際は、代入演算子は使用せず、変数名のみ記述する。
- ✓ 関数の引数として変数を指定すると、変数を参照して取り出した値が引数として設定される。

calc_variables.py

```
01: # 変数の作成・代入・参照
02: val = 25
03: print(val)
04:
```

マッチスル

参照と代入

- ✓ 変数の参照と代入を同時に行うことも可能。
- ✓ 現在格納されている値を演算などに利用し、すぐさま変数の上書きも行う内容。
- ✓ 数値のカウントアップなど、プログラミングにおいて目にする機会の多い記述方法。
- ✓ Pythonでは、**累算代入演算子**（+=）を用いることで、同様の結果を得ることも可能。

calc_reference.py

```
01: # 変数の作成・代入・参照
02: val = 25
03: print(val)
04:
05: # 変数の参照・代入（同時に実施）
06: val = val + 10
07: print(val)
08:
```

calc_reference.py

```
01: # 変数の作成・代入・参照
02: val = 25
03: print(val)
04:
05: # 変数の参照・代入（同時に実施）
06: val += 10
07: print(val)
08:
```

演算子

マッチスル

算術演算子

演算子	説明	例
`+`	加算	`5 + 3 → 8`
`-`	減算	`10 - 4 → 6`
`*`	乗算	`7 * 2 → 14`
`/`	除算	`8 / 2 → 4.0`
`//`	整数除算	`8 // 3 → 2`
`%`	剰余（余り）	`10 % 3 → 1`
`**`	累乗	`2 ** 3 → 8`

代入演算子

演算子	説明	例
`=`	代入	`x = 5`
`+=`	加算代入	`x += 3`（`x = x + 3` と同じ）
`-=`	減算代入	`x -= 2`（`x = x - 2` と同じ）
`*=`	乗算代入	`x *= 4`（`x = x * 4` と同じ）
`/=`	除算代入	`x /= 2`（`x = x / 2` と同じ）
`//=`	整数除算代入	`x //= 3`
`%=`	剰余代入	`x %= 3`
`**=`	累乗代入	`x **= 2`

文字列結合演算子

演算子	説明	例
`+`	文字列結合	`"Hello" + " World" → "Hello World"`
`*`	文字列の繰り返し	`"Hi" * 3 → "HiHiHi"`

比較演算子

演算子	説明	例
`==`	等しい	`5 == 5 → True`
`!=`	等しくない	`5 != 3 → True`
`>`	より大きい	`7 > 4 → True`
`<`	より小さい	`2 < 6 → True`
`>=`	以上	`5 >= 5 → True`
`<=`	以下	`3 <= 4 → True`

論理演算子

演算子	説明	例
`and`	かつ（AND）	`True and False → False`
`or`	または（OR）	`True or False → True`
`not`	否定（NOT）	`not True → False`

ビット演算子

演算子	説明	例
`&`	AND	`5 & 3 → 1`
`|`	|	OR
`^`	XOR	`5 ^ 3 → 6`
`~`	NOT（補数）	`~5 → -6`
`<<`	左シフト	`5 << 1 → 10`
`>>`	右シフト	`5 >> 1 → 2`

算術演算子	四則演算を行う演算子
代入演算子	左辺の変数に右辺の値を代入する演算子
比較演算子	等価判定や大小比較を行う演算子
論理演算子	論理演算を行う演算子
文字列結合演算子	文字列を結合する演算子
ビット演算子	ビット演算を行う演算子

号のこと。

下の通り。

コラム①：数値と文字列の変換

マッチスル

数値と文字列の変換

- ✓ 数値と文字列の演算する場合、基本的にはエラーとなる。
- ✓ 数値と文字列は、関数を用いて相互に変換することが可能。

数値を文字列に変換	str関数
文字列を数値に変換	int関数

calc_change.py

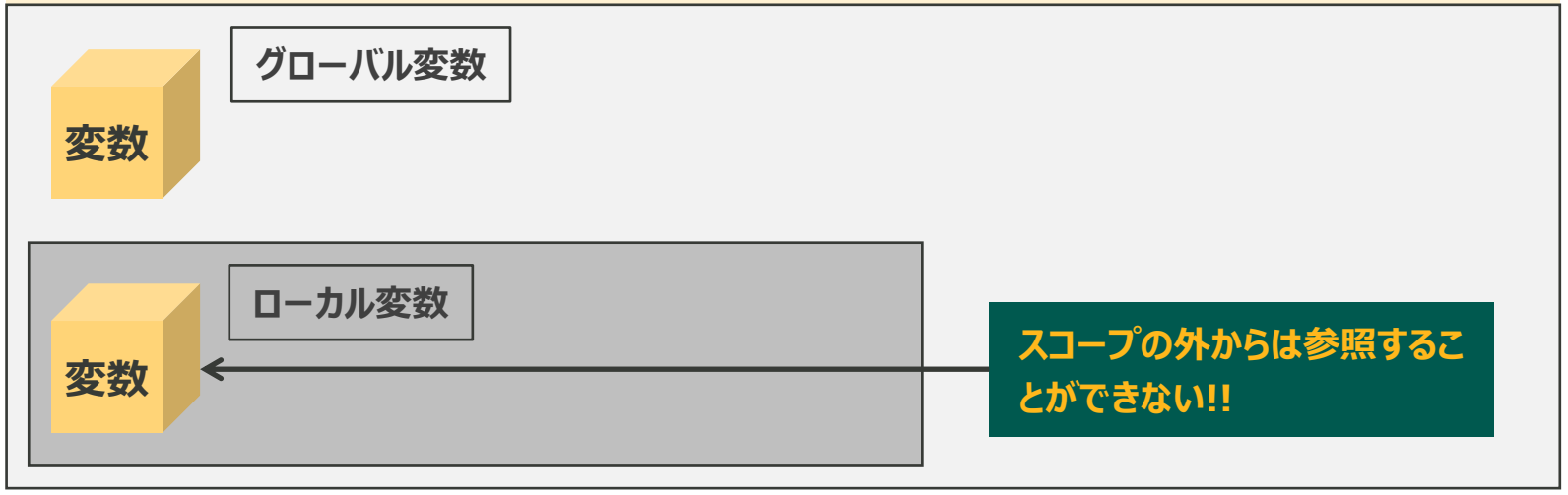
（ファイルを作成してコマンドプロンプトにて実行する場合）

```
01: print(40 + 10)                  # 演算結果：50    （数値）
02: print(str(40) + str(10))        # 演算結果：”4010”（文字列）
03: print(“40” + “10”)              # 演算結果：”4010”（文字列）
04: print(int(“40”) + int(“10”))    # 演算結果：50    （数値）
05:
06:
07:
08:
```

コラム②：変数のスコープ

変数のスコープ

- ✓ スコープとは、変数毎に設定される存在範囲のこと。
- ✓ スコープの外からは代入や参照を行うことはできない。
- ✓ 必要より狭いスコープしか設定していない場合、変数へのアクセスができず正常に動作しない恐れがある。
- ✓ 必要より広いスコープを設定している場合、変数を不必要に書き換えてしまい誤動作を招く恐れがある。
- ✓ プログラムが複雑になってくると、適切なスコープを設定することはとても大切となる。




```python
x = 10

def modify_x():
    global x
    x = 20  # グローバル変数を変更

modify_x()
print(x)  # 出力: 20
```

```python
def my_function():
    y = 5  # ローカル変数
    print(y)  # 出力: 5

my_function()

print(y)  # エラー: NameError: name 'y' is not defined
```

```python
def outer():
    a = 5

    def inner():
        nonlocal a
        a = 10

    inner()
    print(a)  # 出力: 10

outer()
```

章のまとめ

マッチスル

基本データ型

変数

- ✓ プログラムで使用する数値や文字列などを一時的に記憶しておく領域のこと。
- ✓ Pythonでは、データ型を指定しなくても変数の作成が可能。

変数の利用

- ✓ 変数に値を格納することを代入、変数から値を取り出すことを参照と呼ぶ。
- ✓ 変数の参照と代入を同時に行うことも可能。

演算子

- ✓ 加減乗除を行う算術演算子や、代入演算子、文字列結合演算子がある。
- ✓ 制御構文の条件式を作る際に使用する比較演算子や論理演算子がある。

数値と文字列の変換

- ✓ 数値を文字列に変換する際は、str関数を使用する。
- ✓ 文字列を数値に変換する際は、int関数を使用する。