ArcGIS®

Python の基本構文

esri ジャパン

目次

第 1 章 演習環境の構築 5
演習 1: 演習環境の構築 7
ステップ 1: PyCharm のインストール 7
ステップ 2: PyCharm を日本語表示に変更 8
ステップ 3: PyCharm の設定 9
第 2 章 Python の基本構文 13
概要 15
Python スクリプトの統合開発環境 (IDE) 15
PyCharm 16
Python の基本構文 16
変数 17
Python の一般的なデータ型 17
文字列 (1) 18
文字列 (2) 18
クラス 19
組み込み関数 (1) 19
引数と戻り値 20
組み込み関数 (2) 20
モジュール (1) 21
モジュール (2) 21
文 (ステートメント) 22
文 (2) 22
文 (3) 23
その他の構文 23
補足資料 24
Python のコーディング スタイル 24
よく使用するキーワード ① 25
よく使用するキーワード ② 25
よく使用するキーワード ③ 26
よく使用するキーワード ④ 26
一般的な例外 27
演習 2: Python の基礎を学習 29
ステップ 1: スクリプト ファイルの作成 29
ステップ 2: 変数の操作 32
ステップ 3: 組み込み関数の利用 35
ステップ 4: モジュールの操作 37
ステップ 5: 条件分岐の操作 40
ステップ 6: ループの操作 42

質問の解答 45

アイコンの説明

演習時間：演習時間の目安です。

ノート：特定のトピック、手順に関する追加の情報、例外事項や特記事項を示します。

ティップス：概念の理解や手順を実行するための簡単なヘルプです。

外部リソース：トピックに関する参考資料です。

ベスト プラクティス：目的や優先事項を効率よく達成するためのガイドラインです。

警告：間違えやすい箇所や避けるべき操作です。

1

演習環境の構築

演習 1: 演習環境の構築

演習のポイント:

✓ PyCharm のインストール
✓ PyCharm の日本語表示設定

演習時間: 20 分

ステップ 1: PyCharm のインストール

Python を実行するためのアプリケーションは多数存在しますが、本コースでは PyCharm を使用します。

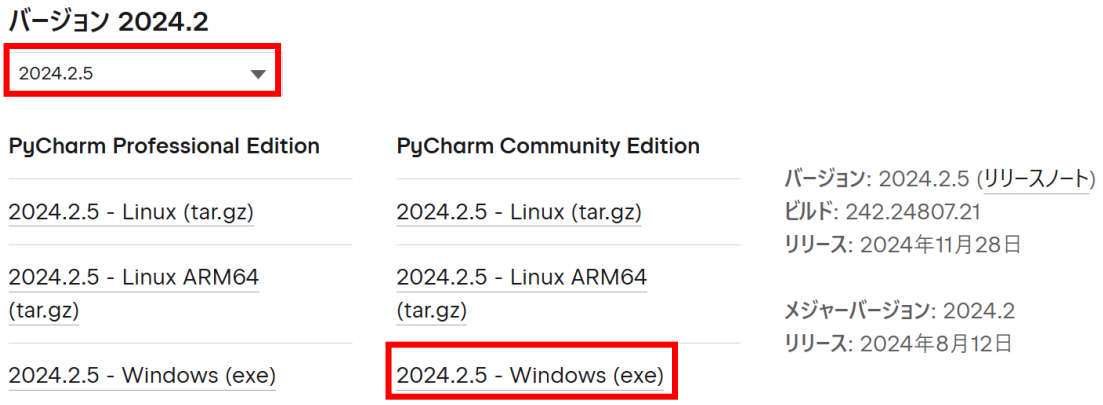
① Web ブラウザーで「PyCharm」を検索します。

② PyCharm のトップ ページで [ダウンロード] をクリックします。

③ 次に、[他のバージョン] をクリックします。

上記の [ダウンロード] をクリックすると、最新バージョンがダウンロードされます。このテキストでは PyCharm Community Edition の**バージョン 2024.2.5** を使用します。

④ [バージョン 2024.2] 欄を探し、ドロップダウン リストから「2024.2.5」を選択し、[PyCharm Community Edition] の「2024.2.5 -Windows (exe)」をクリックします。



⑤ ダウンロードした exe ファイルを実行し、ウィザードに従ってインストールを行います。

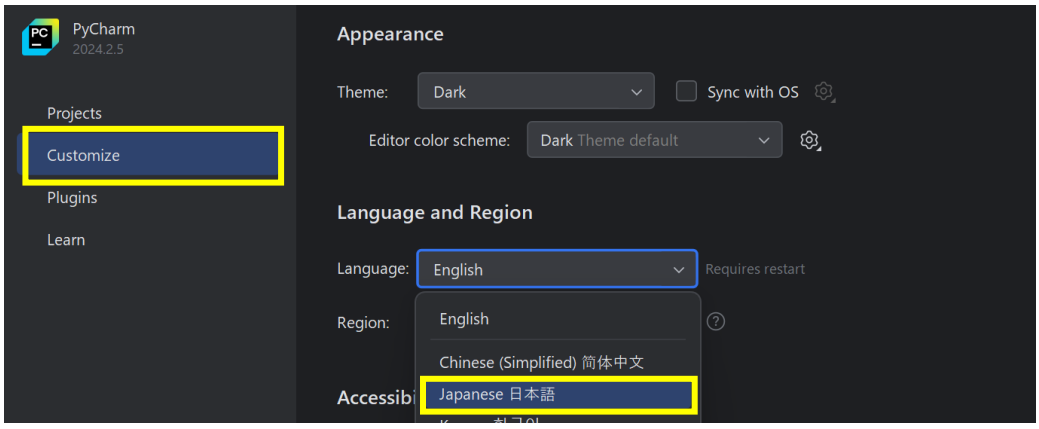
途中のウィザードにある [インストール オプション] にて [デスクトップショートカットの作成] 欄の「PyCharm Community Edition」にチェックを入れてインストールすることで、デスクトップに PyCharm のショートカットが作成されます。

ステップ 2: PyCharm の日本語表示設定

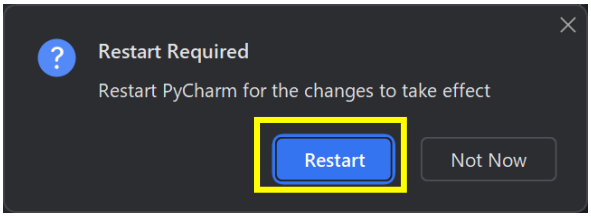
デフォルトは英語での表示のため、日本語で表示できるように設定を行います。

① PyCharm を開きます。

② [Customize] をクリックし、[Language:] のドロップダウンから [Japanese 日本語] を選択します。



③ 下記のダイアログ表示後に、[Restart] をクリックし、PyCharm に日本語言語パックを適用し、PyCharm を再起動します。



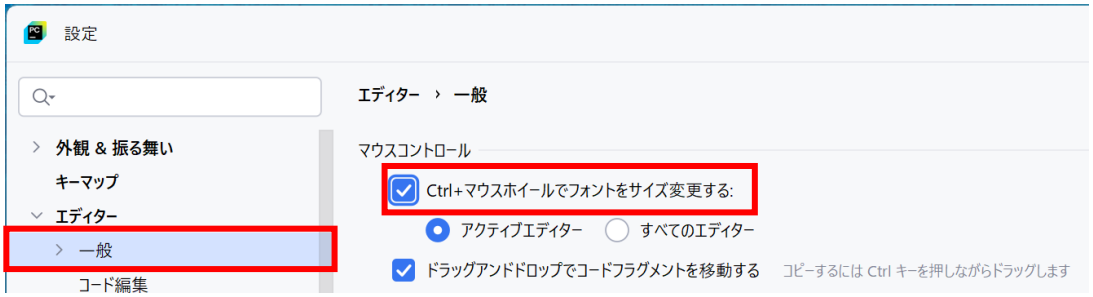
再起動後、日本語表示に変更されます。

ステップ 3: PyCharm の設定

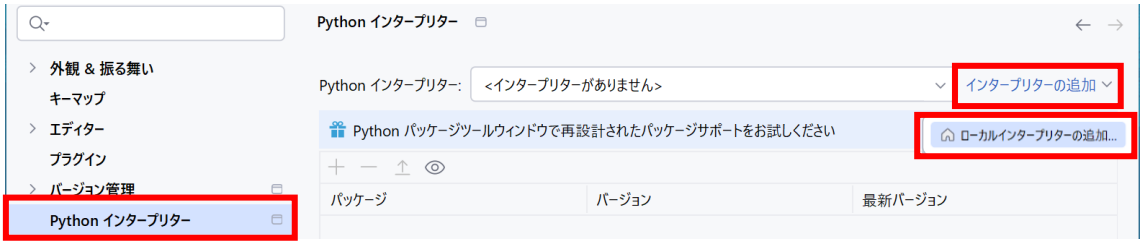
① [カスタマイズ] をクリックし、[すべての設定] をクリックします。

[テーマ] のドロップダウンから表示色を変更することが可能です。デフォルトは [Dark] ですが、テキストでは PyCharm の画面が見やすいように、[Light] に変更しています。

② [エディター] を展開し、[一般] を選択し、[Ctrl+マウスホイールでフォントをサイズ変更する] にチェックを入れます。

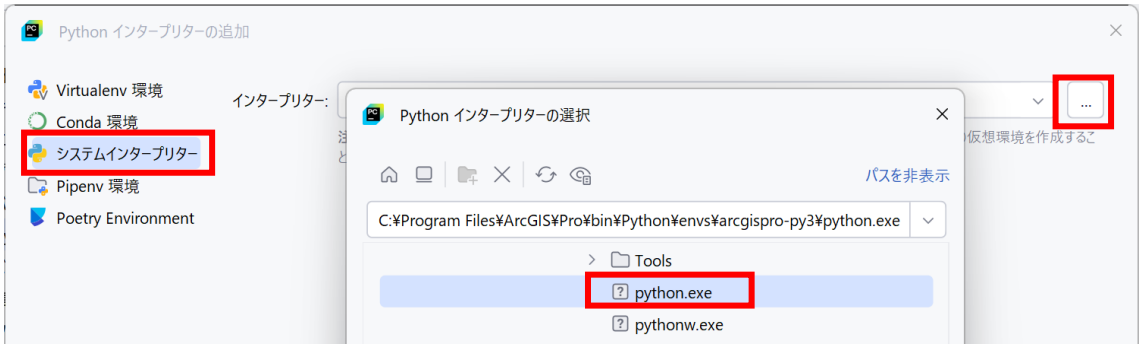


③ [Python インタープリター] を選択し、[インタープリターの追加] → [ローカルインタープリターの追加] をクリックします。



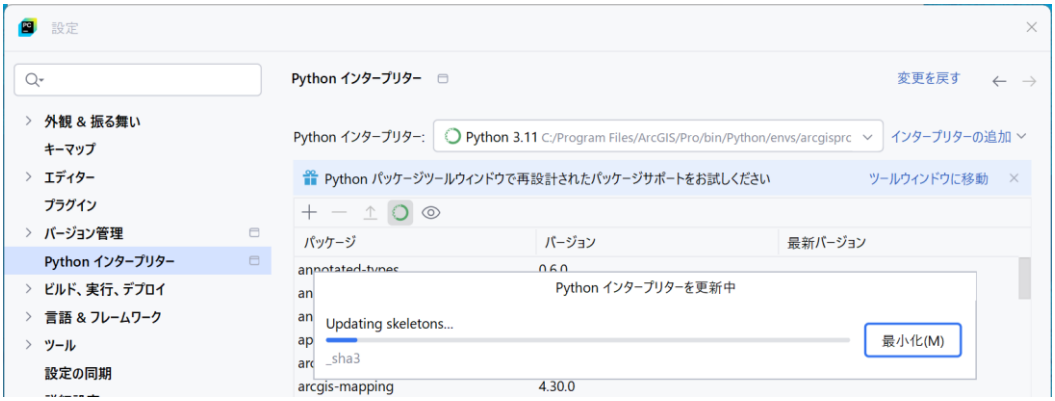
④ [Python インタープリターの追加] ダイアログで [システムインタープリター] を選択 → [...] をクリック → [Python インタープリターの選択] ダイアログで下記パスにある「python.exe」を選択 → [OK] を 2 回クリックします。

- C:¥Program Files¥ArcGIS¥Pro¥bin¥Python¥envs¥arcgispro-py3

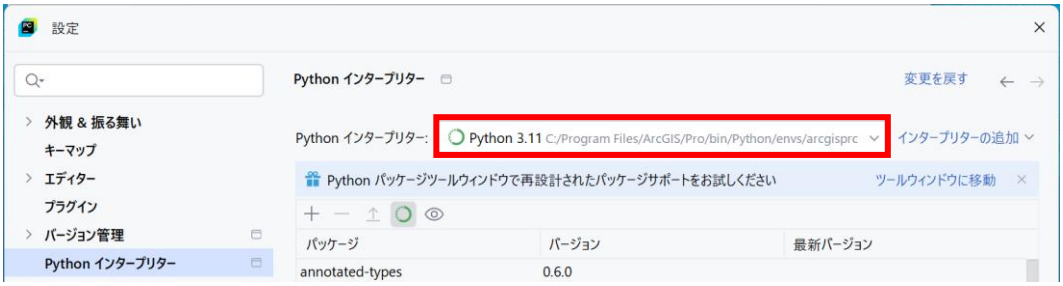


上記の中に指定のファイルを見つけることができない場合は、ArcGIS Pro のインストール時に他のフォルダーを指定して ArcGIS Pro をインストールしている可能性があります。デスクトップ上に配置されている「ArcGIS Pro」のアイコン右クリックし、[ファイルの場所を開く] を選択すると、どこに ArcGIS Pro がインストールされているかを確認することができます。

Python インタープリターの更新が始まります。



⑤ 更新終了後、[Python インタープリター] に「Python 3.11」と表示されていることを確認し、[適用] をクリックします。



⑥ [OK] をクリックし、変更を適用します。

⑦ PyCharm のウィンドウ右上の [×] をクリックし、PyCharm を閉じます。

> ArcGIS Pro をお持ちでない方は、下記 PyCharm のヘルプ ページを参考に、Microsoft ストアから Python をインストールし、ヘルプ ページの手順に従い設定を行ってください。
> PyCharm ヘルプページ: 新しい Python インタープリターの作成

> PyCharm に関する操作につきましては、下記、公式ヘルプ ページをご参考ください。
> https://pleiades.io/help/pycharm/2024.2/installation-guide.html

> 「ArcGIS Pro: Python スクリプト入門」をリモートでご受講予定の方で、かつ、自前の環境でご受講予定の方は、必ず、講習会のテキストにある演習 1 (演習の環境設定) を確認し、受講の準備を行ってください。

2

Python の基本構文

概要

- Python とは
- 開発環境について
- Python 基本構文の理解

概要

概要

- Python の開発環境
- Python 言語の基本構文の理解

Python スクリプトの統合開発環境 (IDE)

Python スクリプトの統合開発環境 (IDE)

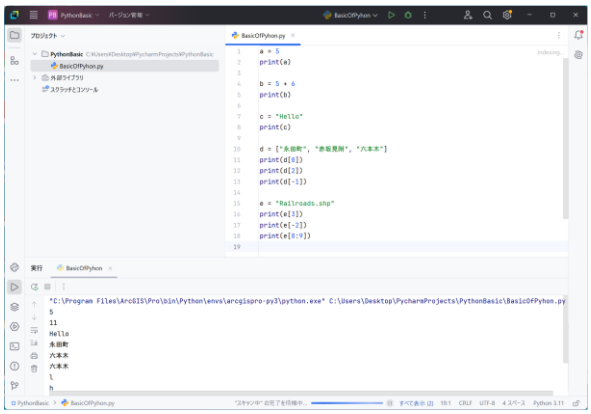
- コードエディター、デバッガーなど開発に必要な機能がパッケージされたアプリケーション

- Python でよく使われる IDE
  - PyCharm
  - PyScripter
  - Visual Studio
  - Visual Studio Code (VS Code)
  - IDLE
    - Python に付属

PyCharm

PyCharm

- Windows、Mac OS、Linux
- 日本語表示が可能
  - 演習 1 に記載あり



Python の基本構文

- 変数
  - 数値や文字など、データを格納
- データ型
  - 文字列、数値、リストなど様々な種類が存在
- 組み込み関数
  - あらかじめ Python に組み込まれている基本的な機能
- モジュール
  - 機能が格納されているファイル
  - インポート (読み込み) することで利用可能
- 文 (ステートメント)
  - プログラムに指示を行う文

変数

変数

- データを格納できる入れ物
- 変数は動的に入力
  - キーワードによる宣言不可
  - 変数名に日本語・予約語は使用不可
  - 数字から始まる名前は使用不可

```
path = r"D:¥Student¥PYTS¥Data¥Geodatabase¥CountyData.gdb¥ServiceAreas"
```

- 変数は大文字・小文字を区別

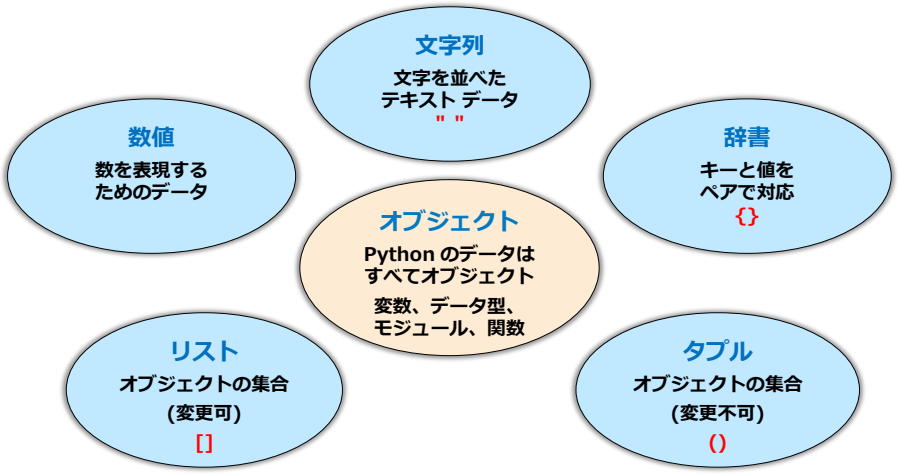
```
scale = 10000
SCALE = 20000
```

2 つは異なる変数

- 変数にはさまざまなデータ型を格納可能
  - 文字列、数値、リスト、タプル、辞書、ファイル

Python の一般的なデータ型

Python の一般的なデータ型



文字列 (1)

文字列 (1)

- シングル クォーテーション、ダブル クォーテーション
  トリプル クォーテーション
  - 文字列として扱うには、クォーテーションで囲む `a = 'python'` or `a = "python"`
  - トリプル クォーテーションは改行を含んだ文字列を記述可能

```
a = '''
python
Lecture'''
```

or

```
a = """
python
Lecture"""
```

- エスケープ シーケンス
  - ¥t 水平タブ "D:¥test" → D: est
  - ¥n 改行 "D:¥nakamura" → D:
    akamura
- raw 文字列
  - エスケープ シーケンスによる文字列の置き換えを無効化
    path = r"D:¥test" → D:¥test

文字列 (2)

文字列 (2)

- 文字コード指定が可能
  - スクリプトの 2 行目までにコメント文で指定

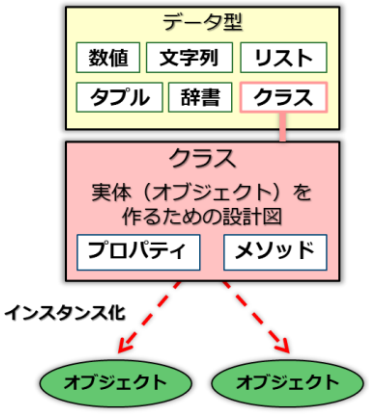
`#coding:cp932` または `#coding:Shift_JIS` または `#coding:utf-8`

  - Python 3 のデフォルトの文字コードは UTF-8
  - 指定する文字コードはファイルに依存
    - PyCharm で作成されるファイルの文字コードは UTF-8

クラス

クラス

- Python のデータ型
- クラスからオブジェクトを作成して利用 (インスタンス化)
- 各クラスにはプロパティとメソッドが存在
  - プロパティ
    - それぞれのクラスが持つ性質
    - 読み取り、書き込みの 2 種類
  - メソッド
    - それぞれのクラスが持つ独自の関数
- 独自にクラスの定義が可能



組み込み関数 (1)

組み込み関数 (1)

- Python をインストールすることで利用可能
- ユーザーが定義せずに使用可能な関数

組み込み関数

len()	range()
max()	min()
file()	pow()
round()	print()
help()	str()
dir()	type()
input()	sorted()

引数と戻り値

- 引数
  - 処理の実行に使用する数値や文字列
  - 関数名の後ろの丸括弧内に記述
  - 関数によって引数の数、与えるオブジェクトの型が異なる
- 戻り値
  - 関数が結果として返す値
  - 関数によって返されるオブジェクトの型が異なる
- 例
  - pow(x, y)：累乗を計算（x の y 乗を返す）

```
var = pow(10, 2)      # 関数名(引数)
print(var)
```

```
100                   # 戻り値
```


組み込み関数 (2)

- Python には多くの組み込み関数が存在
  - len()：長さ (リストの場合は、要素数) を返す

```
fc = "Railroads.shp"
print(len(fc))
fields = ["OID", "Shape", "Name"]
print(len(fields))
```

```
13      #戻り値
3
```

  - type()：データ型を返す

```
fc = "Railroads.shp"
print(type(fc))
fields = ["OID", "Shape", "Name"]
print(type(fields))
```

```
<class 'str'>
<class 'list'>
```

  - round()：数値を丸める

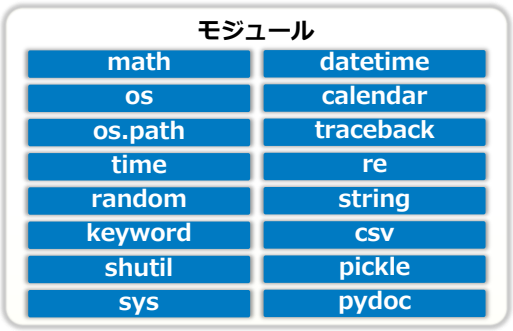
```
vCoord = 1811884.623694
print(round(vCoord))
```

```
1811885
```

モジュール (1)

モジュール (1)

- Python をインストールすることにより
  さまざまなモジュールもインストールされる
- 外部で配布されているモジュールも数多く存在



モジュール (2)

モジュール (2)

- 多くの関数はモジュールからインポートが必要
  - math モジュール：数学関数にアクセス

```
import math
print(math.sqrt(64))
print(math.pow(10, 2))
```

→

```
8.0      #戻り値
100.0
```

  - random モジュール：乱数に関する関数にアクセス

```
import random
print(random.randint(0, 10))
print(random.randint(0, 10))
```

→

```
7
3
```

  - os.path モジュール：パス名を操作

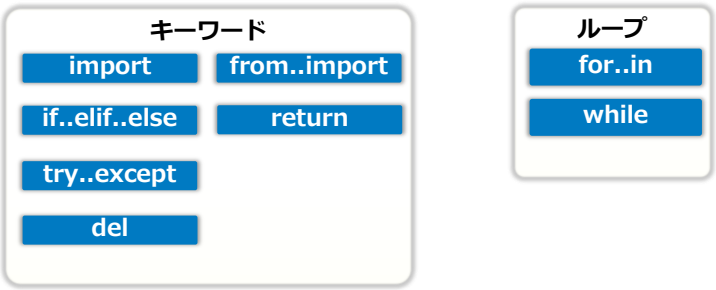
```
import os.path
print (os.path.basename(r"D:¥Student¥Streets.shp"))
print (os.path.dirname(r"D:¥Student¥Streets.shp"))
```

→

```
Streets.shp
D:¥Student
```

文 (ステートメント)

- プログラミングに指示を行う文
- 単純文、複合文
- 式との組み合わせでスクリプトを構成



文 (2)

- 単純文
  - import：モジュールをインポート

```
import math
import random
import os
```

  - from..import：モジュール内の特定の関数をインポート

```
from math import
sqrt
print(sqrt(64))
```

  - import..as：モジュールに任意の名前を付けてインポート

```
import math as m
print(m.sqrt(64))
```

文 (3)

- 複合文
  - if..elif..else 文：条件分岐

```
shp = "polygon"
if shp == "point":
    print("ポイントです")
elif shp == "polyline":
    print("ポリラインです")
elif shp == "polygon":
    print("ポリゴンです")
else:
    print("不明です")
```

コロンとインデントによる
ブロック表現

```
#戻り値
ポリゴンです。
```

  - for..in 文：ループ処理

```
fcs = ["徳島.shp", "香川.shp", "愛媛.shp", "高知.shp"]
for fc in fcs:
    print(fc.replace(".shp", "_temp.shp"))
```

```
徳島_temp.shp
香川_temp.shp
愛媛_temp.shp
高知_temp.shp
```


その他の構文

- コードのコメント化
  - 半角シャープ記号（#）
- 大文字小文字の区別
  - 区別あり
    - 変数名
    - 文 (ステートメント)
    - 関数
    - ジオプロセシング関数やクラス名
  - 区別なし
    - パス名

補足資料

esri ジャパン

補足資料

Python のコーディング スタイル

Python のコーディング スタイル

- 記号はすべて半角で記述
  - = + () “” [] - , :
- スペースもすべて半角で入力
- Python のコーディング規約である PEP8 を参考に記述
  - https://pep8-ja.readthedocs.io/ja/latest/

```
# 正しい:
d = ["永田町", "赤坂見附", "六本木"]

# 間違い:
d= [ "永田町 " , " 赤坂見附" ,   "六本木" ]
```

よく使用するキーワード ①

- 変数
  - データを格納できる入れ物
  - 大文字・小文字が区別される
- 文字列
  - String 型
  - Unicode 型
    - 世界中の文字を扱えるようにした文字列の型
- 文字コード
  - コンピュータが文字を扱うために割り当てる数値のこと

よく使用するキーワード ②

- CP932
  - Microsoft 社が Shift_JIS を独自に拡張した文字コード
- Shift_JIS
  - コンピューターで日本語を表示するために 1 文字ずつ番号を割り当てた文字コード
  - 日本語の場合、1 文字を 2 バイトで表現
- UTF-8
  - Unicode で定義された文字集合体をバイト列に変換する世界基準な文字コード
  - 日本語の場合、1 文字を基本的には 3 バイトで表現
- Unicode
  - 世界中の文字を一元的に扱えるようにした文字集合

よく使用するキーワード ③

- 関数
  - さまざまな処理が機能として まとまっているもの
  - 関数にまとめることで、 一連の処理を繰り返し呼び出せる
- 引数
  - 関数に引き渡す値のこと
- 戻り値
  - 関数が呼び出し元に返す値
- モジュール
  - Python の定義や文が入った ソースファイル
- パッケージ
  - 複数のモジュールで構成されたもの

よく使用するキーワード ④

- クラス
  - オブジェクトの設計図
  - プロパティとメソッドを持つ
- プロパティ
  - オブジェクトが保持する情報 の取得や設定が可能
- メソッド
  - オブジェクトに何か処理をさせる
- オブジェクト
  - 設計図から作った実体
  - 実際にプロパティやメソッドを 使用できるもの
- インスタンス化
  - あるクラスのオブジェクトを 作成すること
- インスタンス
  - 作成されたオブジェクトのこと

一般的な例外

- NameError
  - 未定義の変数などを参照した場合に発生
- IndexError
  - リスト内の要素をインデックスを使用して参照した際に、要素数を超える数を指定した場合に発生
- TypeError
  - 関数の引数として、正しくない型を設定した場合に発生
  - 文字列と数値を連結させるなど、演算子を使った演算をする場合に発生
- ImportError
  - 存在しないモジュールをインポートしようとした場合に発生

演習 2: Python の基礎を学習

この演習では、Python スクリプト言語の基礎を学習します。具体的には、コメントの書き方、変数の型、組み込み関数、モジュール、文字の連結、文 (ステートメント) と制御構造などを学習します。

演習時間: 40 分

演習のポイント:

✓ スクリプト ファイルの作成
✓ 変数の操作
✓ 組み込み関数の利用
✓ モジュールの操作
✓ 条件分岐の操作
✓ ループの操作

ステップ 1: スクリプト ファイルの作成

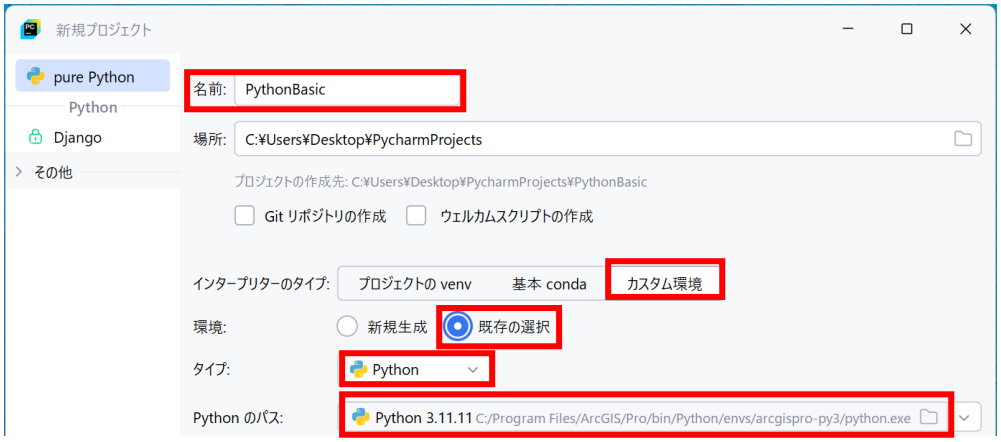
① PyCharm を起動します。

PyCharm を初めて起動すると、「PyCharm へようこそ」画面が表示されます。新規にプロジェクトを作成しましょう。

② [新規プロジェクト] をクリックします。

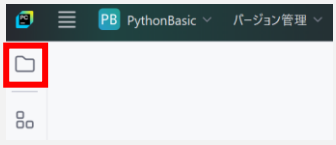
③ [新規プロジェクト] ダイアログで下記を設定し、[作成] をクリックします。

- [名前]:「PythonBasic」
- [場所]: デフォルトのままで構いませんが、必要に応じてご変更ください
- [インタープリターのタイプ]:「カスタム環境」を選択
- [環境]:「既存の選択」を選択
- [タイプ]:「Python」を選択
- [Python のパス]: 下記のパスであることを確認
  「C:¥Program Files¥ArcGIS¥Pro¥bin¥Python¥envs¥arcgispro-py3¥python.exe」

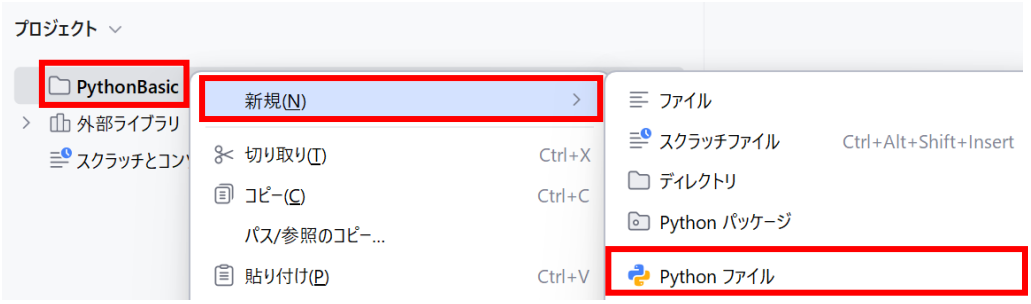


プロジェクトが開きます。

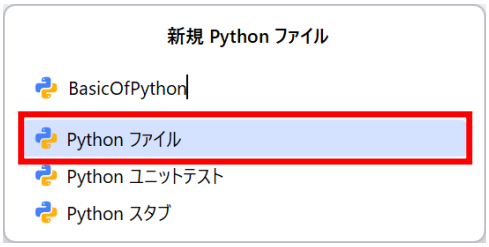
プロジェクトツールウィンドウが開いていない場合は、画面左にあるツールウィンドウバーの最上部にある [プロジェクト] をクリックし、開きます。



④ プロジェクトツールウィンドウに表示されている [PythonBasic] フォルダーを右クリック → [新規] → [Python ファイル] をクリックします。



⑤ 「BasicOfPython」と名前を付けて、[Python ファイル] をダブルクリックします。



「PythonBasic」フォルダー内に「BasicOfPython.py」ファイルが作成され、ファイルがエディター上で開かれます。

⑥ ウィンドウヘッダーの左から 2 つ目のアイコン、[メインメニュー] をクリックし、[ファイル] メニューを開きます。

⑦ [ファイル] メニューにある [設定] をクリックし、[プロジェクト: PythonBasic] → [Python インタープリター] を選択し、「Python 3.11 C:¥Program Files¥ArcGIS¥Pro¥bin¥Python¥envs¥arcgispro-py3¥python.exe」が設定されていることを確認します。

選択されていない場合は、ドロップダウンから「Python 3.11 C:¥Program Files¥ArcGIS¥Pro¥bin¥Python¥envs¥arcgispro-py3¥python.exe」を選択してください。ドロップダウンに上記が表示されない場合は、**演習 1 のステップ 3 の ③ ~ ⑥** を行ってください。

⑧ 確認後、[OK] をクリックし、ダイアログを閉じます。

ステップ 2: s変数の操作

Python を含むすべてのスクリプト言語で、変数が使用できます。変数はプログラム上にある情報を一時的に保存しておく貯蔵タンクのようなものと考えてください。変数の値は、プログラムの実行中に変更することができます。

Python では、変数の型を宣言することなく、変数に値を代入できます。Python は Python 自身でデータの型（タイプ）を判断します。これを動的な型付けといいます。変数には、文字列、数値、リスト、タプル、ディクショナリ、ファイルなどを含む、さまざまなタイプのデータを格納することができます。

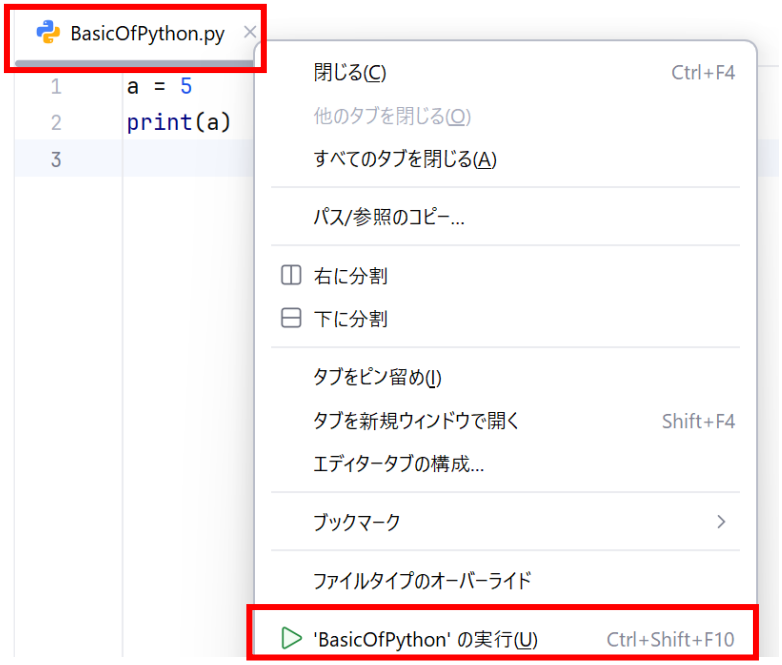
まず、変数に数値を格納してみましょう。

① 以下のコードを記述します。

```
a = 5
print(a)
```

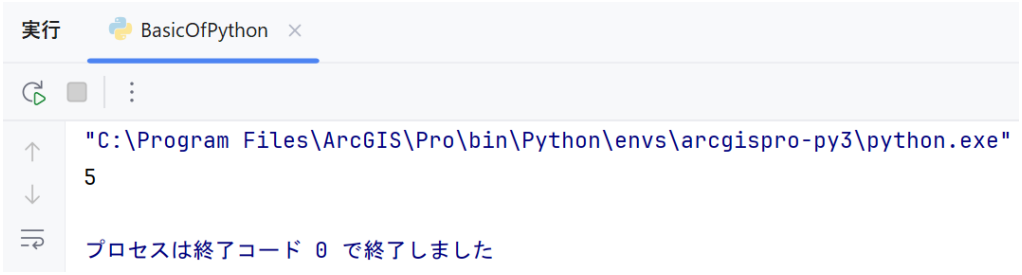
記号は半角で入力を行ってください。

② 「BasicOfPytnon.py」タブを右クリックし、['BasicOfPython' の実行] をクリックし、スクリプトを実行します。



記述してすぐに実行できるコーディングのことをスクリプトといいます。

print 関数は文字列や数値、変数に格納された値を表示する関数のため、画面下部の[実行ツールウィンドウ] に、変数「a」に格納した値が表示されました。



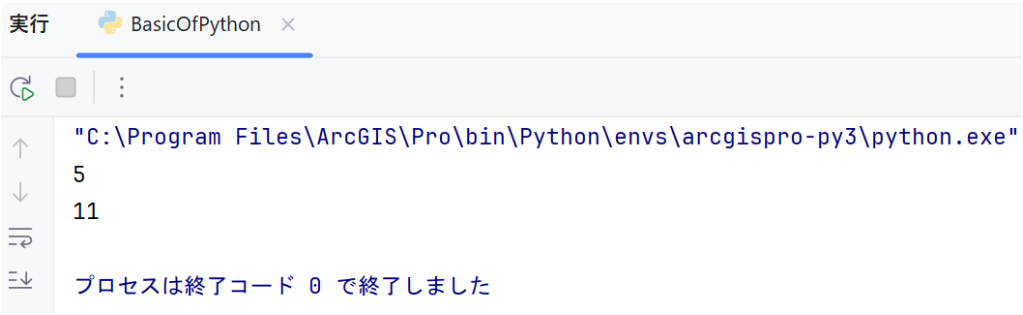
スクリプトの実行は、以下の方法から可能です。
・[メインメニュー] → [実行] メニュー → [実行...] → ['スクリプト名']
・[実行] メニュー → [実行_'スクリプト名'] / Shift キー + F10 キー
・コードを記述している箇所で右クリック → ['スクリプト名' の実行]
・[スクリプト名.py] タブ右クリック → ['スクリプト名' の実行]
・上部のバーにある再生のアイコンをクリック

次に、変数に算術演算式の結果を代入します。

③ 「print(a)」の次の行から、以下のコードを記述します。

```
b = 5 + 6
print(b)
```

④ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。



[実行ツールウィンドウ] に変数「a」、「b」に格納された値が表示されます。「b」の値には「5 + 6」の合計値である「11」が保持されています。

変数には文字列を格納することも可能です。文字列はシングル、またはダブルクォーテーションで囲む必要があります。

⑤ 「print(b)」の次の行に、以下のコードを記述します。

```
c = "Hello"
print(c)
```

⑥ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

[実行ツールウィンドウ] の最後の行に、「Hello」という単語が表示されます。

変数には、複数の値を持つオブジェクトを格納することも可能です。リストもその 1 つで、取り出す時はインデックス（位置）に基づいて リストから個々の値（または値の一部）を抽出することができます。

⑦ 「print(c)」の次の行に、以下のコードを記述します。

```
d = ["永田町", "赤坂見附", "六本木"]
print(d[0])
print(d[2])
print(d[-1])
```

行数が増えて見えづらい場合は途中に空白行（改行）を入れても構いません。

⑧ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

[実行ツールウィンドウ] の最後の 3 行に、「永田町」、「六本木」、「六本木」という出力結果が表示されます。d[0] はリストの最初の値である「永田町」が出力結果として返されます（リスト内のインデックス番号は 0 から始まります）。d[2] はリストの 3 番目の値である「六本木」が出力結果として返されます。d[-1] はリストの右から最初の値である「六本木」が出力結果として返されます。

負の値のインデックスは右端からの位置を表します。

また、リスト以外に文字列に対してもインデックス番号で文字を指定することができます。

⑨ 「print d[-1]」の次の行に、以下のコードを記述します。

```
e = "Railroads.shp"
print(e[3])
print(e[-2])
print(e[0:9])
```

⑩ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

[実行ツールウィンドウ] の最後の 3 行に、「l」、「h」、「Railroads」という出力結果が表示されます。e[3] は文字列の 4 文字目である「l」が出力結果として返されます。e[-2] は文字列の右から 2 文字目である「h」が出力結果として返されます。e[0:9] は文字列の最初の文字 (インデックス = 0) から 9 文字目 (インデックス = 8) までを取り出します。

R a i l r o a d s

[0][1][2][3][4][5][6][7][8]

次のステップでは、組み込み関数とその使い方を学習します。関数の学習を行う前に、既存のコードをコメント化します。

⑪ スクリプト内の実行可能なコードをすべて選択し、ハイライト状態にします。

⑫ Ctrl キー + / キーを押下し、コメント化します。

それぞれのコードの前にハッシュ記号 (#) が付きます。すべてのコードがコメント化されたため、どのコードも実行されません。

ステップ 3: 組み込み関数の利用

スクリプトを記述する際に、2 つの文字列を結合する場合は、プラス記号 (+) を使用します。

① スクリプト内の「# print e[0:9]」の次の行に、以下のコードを記述します。

```
f = "Streets"
g = ".shp"
print(f + g)
```

② [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

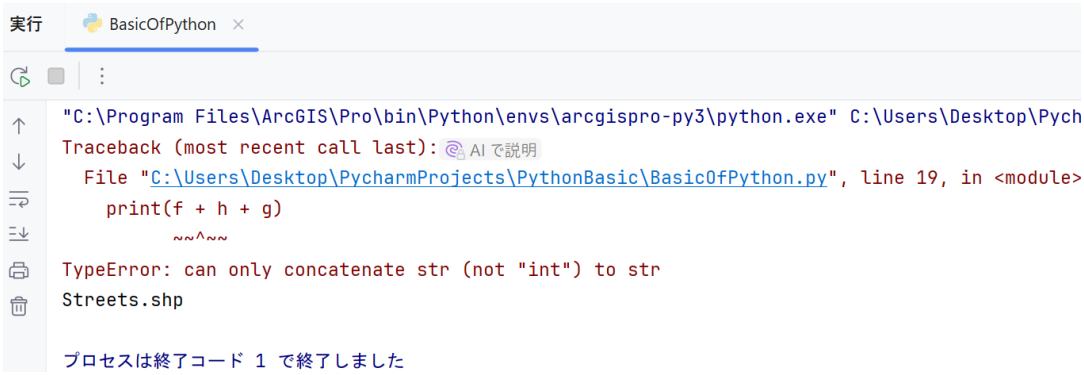
2 つの文字列が連結され、出力結果として「Streets.shp」という文字列が表示されます。また、場合によってはファイル名の末尾に数字を付けるといったように文字列と非文字列を連結させることもあります。

③ 「print(f + g)」の次の行に、以下のコードを記述します。

```
h = 2
print(f + h + g)
```

④ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

下記のようなエラーが発生します。



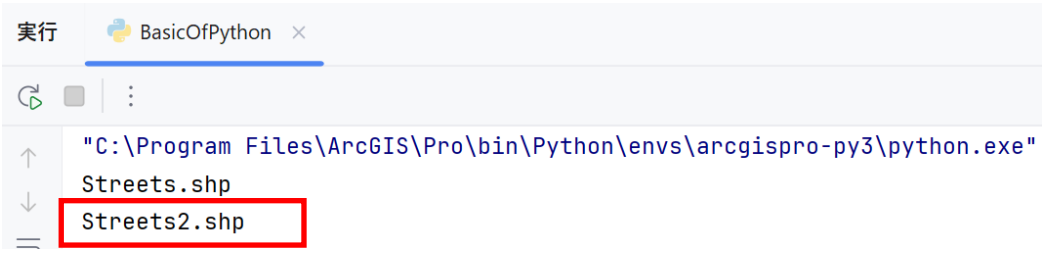
このエラーは文字列と非文字列を連結しようとしたために発生しました。文字列として連結させるためには、非文字列に str 関数を使用し、文字列へ変換する必要があります。str 関数は組み込み関数の 1 つです。その他の組み込み関数について、今後の演習でさらに学習していきます。

⑤ 最後の print 関数を以下のように修正します。

```
h = 2
print(f + str(h) + g)
```

⑥ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

3 つの文字列が連結され、出力結果として「Streets2.shp」と表示されます。



⑦ 実行可能なすべてのコードを選択し、Ctrl キー + / キーを押下し、コメント化します。

整数のべき乗を計算する pow 関数を使ってみましょう。

⑧ 「# print(f + str(h) + g)」の次の行に、以下のコードを記述します。

```
i = pow(2, 3)
print(i)
```

⑨ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

pow 関数は第一引数に入れた数を x、第二引数に入れた数値を y とする場合、x の y 乗を返す関数となるため、「2 x 2 x 2」が計算され、計算結果の「8」が表示されます。

次に文字列の長さやリスト内の要素数を取得することができる len 関数を使ってみましょう。

⑩ 「print(i)」の次の行に、以下のコードを記述します。

```
j = "esri japan"
k = ["a", "b", "c", "d"]
print(len(j))
print(len(k))
```

⑪ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

len 関数の結果として、「10」と「4」という値が表示されます。最初の値は、スペースを含む文字列の長さを示しています。2 つ目の値は、リスト内の要素数 ("a"、"b"、"c"、"d" の 4 つの要素) を示しています。

Python で使用できる関数は組み込み関数だけではありません。次のステップでは、組み込み関数ではない別の関数へアクセスする方法を学習します。

ステップ 4: モジュールの操作

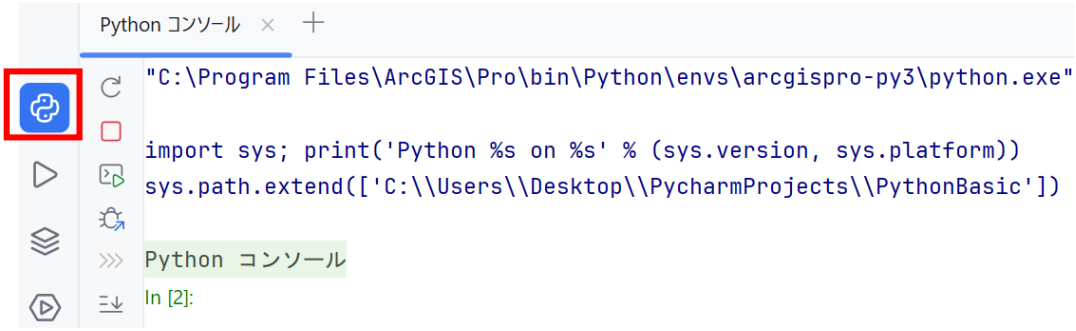
Python には、利用可能な組み込み関数のセットに加え、後からインポートできる標準モジュールに格納された何百もの関数が存在します。モジュールに格納された関数を利用するためには、モジュールにアクセスする必要があります。モジュールにアクセスするためには、まずモジュールをインポートします。このステップでは、[エディター] の代わりに、[Python コンソール] にコードを記述します。

[エディター] に記述するほとんどのコードは、[Python コンソール] にも同様に記述が可能です。しかし、エディターに記述したコードはファイルに保存したコードであり、[Pythonコンソール] はコマンドラインに対して記述するコードという違いがあるので、ファイル内で指定する文字コードの指定等は [Pythonコンソール] では記述することができません。

まず始めに、math モジュールをインポートします。

① PyCharm 下部にある [Python コンソール] ボタン  をクリックします。

> [Windows セキュリティの重要な警告] ダイアログが表示された場合は、[許可] をクリックしてください。



② 「In[2]:」が表示されていることを確認します。表示されていない場合はしばらくお待ちください。

③ 「In[2]:」に続いて以下のコードを記述します。

```
In[2]: import math
```

④ Enter キーを押します。

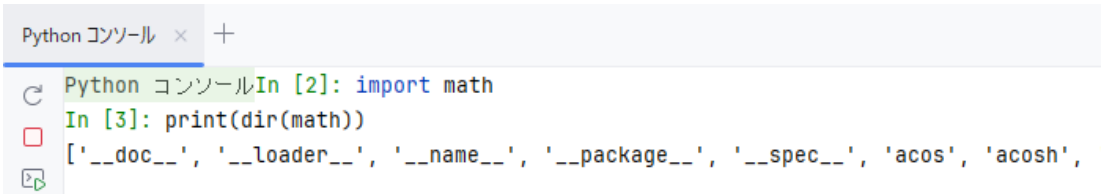
この文により、math モジュール内に含まれるすべての関数へアクセスできるようになりますが、関数のリストは表示されません。math モジュールに含まれる関数のリストを調べるために、組み込み関数の 1 つである dir 関数を利用します。

⑤ [Python コンソール] に以下のコードを記述し、Enter キーを押します。

```
In[3]: print(dir(math))
```

In[ ] 内に表示される数値はコードを実行するたびに増えていくので、テキストと異なっても問題ありません。

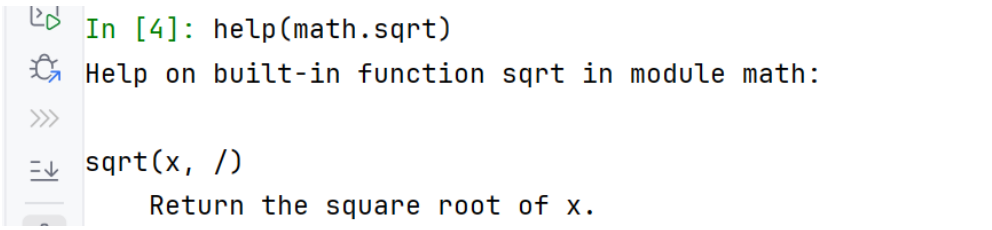
[Python コンソール] では、print 関数を使用しなくても出力することはできますが、print 関数を使用して出力すると、結果が読みやすくなります。



表示されているすべての関数は math モジュールから使用可能な関数です。math、または、Python にあらかじめ用意されている組み込み関数やモジュール、モジュール内の関数には、各関数の定義や使用方法などの情報を確認するための機能が用意されています。

⑥ [Python コンソール] に以下のコードを記述し、Enter キーを押します。

```
In[4]: help(math.sqrt)
```



help 関数を使用することで、英語とはなりますが、sqrt 関数の使用方法と説明が表示されます。適切に sqrt 関数を使用するために、これらの情報は大いに役立ちます。

⑦ [Python コンソール] に以下のコードを記述し、Enter キーを押します。

```
In[5]: math.sqrt(64)
```

sqrt 関数の結果として、「8.0」が返されます。モジュール内の関数を実行するためには、関数名の前に、モジュール名を記述することに注意してください。

また、その他の代表的なモジュールとして、random モジュールがあります。

⑧ [Python コンソール] に、random モジュールをインポートする以下のコードを記述し、Enter キーを押します。

```
In[6]: import random
```

⑨ random モジュールに格納されている関数のリストを表示する以下のコードを記述し、Enter キーを押します。

```
In[7]: print(dir(random))
```

```
In [6]: import random
In [7]: print(dir(random))
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST',
```

⑩ [Python コンソール] に以下のコードを記述し、Enter キーを押します。

```
In[8]: help(random.choice)
```

```
In [8]: help(random.choice)
Help on method choice in module random:

choice(seq) method of random.Random instance
    Choose a random element from a non-empty sequence.
```

choice 関数の説明や使い方が表示されます。choice 関数は引数として入力したリストや文字列から、ランダムに要素を 1 つ選び、出力結果として値を返すことが分かります。

⑪ [Python コンソール] に以下のコードを記述し、Enter キーを押します。

```
In[9]: random.choice(["a", "b", "c", "d"])
```

引数として設定したリストに格納されている 4 つの要素からランダムに 1 つ選ばれ、出力されます。

⑫ [Python コンソール] でキーボードの ↑ キーを押し、再度コマンドを呼び出し、Enter キーを押します。

⑬ 何回か繰り返し、ランダムに結果が選ばれていることを確認します。

math モジュールや random モジュールは、インポートによって使用できるモジュールのほんの一部であり、標準モジュールだけでも数百のモジュールが存在します。

ステップ 5: 条件分岐の操作

スクリプトでよく使われるプログラムの構造として、条件分岐があります。Python でよく使用される条件分岐文は、if – elif – else です。

① [エディター] に戻り、スクリプト内の実行可能なコードをすべて選択し、ハイライト状態にします。

② Ctrl キー + / キーを押下し、コメント化します。

③ [エディター] 上で最終行の次に、以下のコードを記述します。

```
x = 5
if x < 5:
    print("x は 5 よりも小さい")
elif x > 5:
    print("x は 5 よりも大きい")
else:
    print("x は 5")
```

Python では、インデントはプログラムのブロックを構成する役目を果たすので、同じスペース数 (半角スペース 4 つ)、または、タブでインデントを付けてください。タブとインデントの混合は避けてください。なお、PyCharm の場合、自動でインデントが入ります。

BasicOfPython.py

```
25    # print(len(k))
26    x = 5
27    if x < 5:
28        print("x は 5 より小さい")
29    elif x > 5:
30        print("x は 5 より大きい")
31    else:
32        print("x は 5")
```

④ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

現在、変数「x」の値は「5」であるため (x = 5)、else 文内のコードが実行され、「x は 5」という文字列が [実行ツールウィンドウ] に表示されます。

⑤ 「x」の値を「5」より大きい値や、小さい値に変更し、再度スクリプトを実行してみてください。

Python の条件分岐文を使用する際の注意点は以下のとおりです。

- if、elif、else は必ず小文字で記述します。
- 条件文の行頭位置を揃えます。
- 各条件文の最後にコロン (:) を記述します。
- ブロックを構成するインデントは必ず揃えます。

Python には、if ブロックの終わりを示す end if 文や括弧がなく、各条件文はインデントに基づいて認識され、実行されます。すなわち、Python では同じインデントの条件文が終了した時点で、条件文のブロックが終了したと判断します。

次のステップでは、ループの使用方法について学習します。

ステップ 6: ループの操作

スクリプトでよく利用するもう 1 つのプログラムの構造として、一連のコードを何度も繰り返して実行する処理が挙げられます。これは「ループ」と呼ばれます。Python には、while、for の 2 種類のループがあります。このステップでは、2 つのループと、さらに range 関数と組み合わせた利用方法について学習します。

① [エディター] 上のスクリプト内の実行可能なコードをすべて選択し、ハイライト状態にします。

② Ctrl キー + / キーを押下し、コメント化します。

③ スクリプトの最終行の次に、以下のコードを記述します。

```
y = 1
while y < 10:
    print(y)
    y = y + 1
```

while ループは、条件式が True の間、ループ内のコードを実行します。コードの各行については、下記の説明をご覧ください。

- 1 行目: 変数「y」には、まず「1」という値が割り当てられます。
- 2 行目:「y」が「10」未満である限り、ループを実行します。
- 3 行目: [実行ツールウィンドウ] に「y」の値を表示します。
- 4 行目:「y」に「1」を足し、ループの最初 (2 行目) の条件式に戻ります。

while ループは、「y」の値が「10」より小さい間、処理を続けます。「y」の値が「10」に達して条件式が成立しなくなった時点で、ループが終了します。

質問 1: [実行ツールウィンドウ] には、何個の数字が表示されると思いますか？

---

④ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

2 番目の ループ タイプとして、for ループを学習します。for ループは、リストなどのシーケンス型のオブジェクトから要素を 1 つずつ取り出して処理を行います。

⑤ スクリプトの最終行の次に、以下のコードを記述します。

```
fcs = ["City.shp", "Roads.shp", "Railroads.shp"]
for fc in fcs:
    print(fc)
```

この例では、for ループはリスト内にある各文字列の要素に対して実行され、計 3 回出力されます。

⑥ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

[実行ツールウィンドウ] に、各シェープファイルの名前が表示されます。

次に値の範囲を利用してループを実行してみましょう。

⑦ スクリプトの最後行の次に、以下のコードを記述します。

```
for num in range(3, 7):
    print(num)
```

range 関数は開始値と終了値を指定して連番のリストを作成します。終了値に指定した最後の値は含まれませんので、range(3, 7) は 3 から 6 までの連番のリストになります。

⑧ [BasicOfPython.py] タブを右クリック → ['BasicOfPython' の実行] を選択し、スクリプトを実行します。

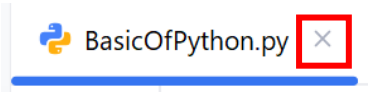
[実行ツールウィンドウ] に、「3」、「4」、「5」、「6」という値が表示されます。

Python のループ文を使用する際の注意点は以下のとおりです。

- while、for、in、range は小文字で記述します。
- while、または、for 文の最後にコロン (:) を記述します。
- 各ループ文は、インデントに基づいて実行されます。

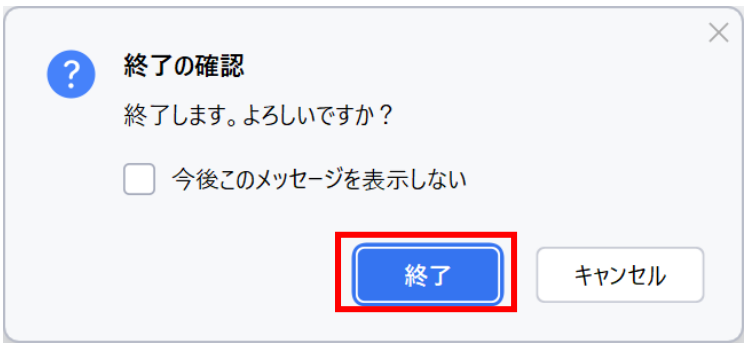
ループ文も条件分岐文と同様に、インデントに基づいてコードブロックが認識され実行されます。すなわち、Python では同じインデントのループ文が終了した時点で、ループのブロックが終了したと判断します。

⑨ [BasicOfPython.py] タブの ✕ をクリックし、スクリプトを閉じます。



⑩ PyCharm を閉じます。

⑪ [終了の確認] ダイアログが表示されたら、[終了] をクリックします。



その他の一般的な注意点:

- 実行したくないコードは、コメント化します。
- 変数名、文、関数名、クラス名は大文字 - 小文字が区別されます。
- パス名は大文字 - 小文字が区別されません。
- パス名の指定の仕方は以下のいずれかです。
  例:「D:¥Student¥PYTS」フォルダーを指定する場合

```
r"D:¥Student¥PYTS"
"D:¥¥Student¥¥PYTS"
"D:/Student/PYTS"
```

質問の解答

解答 1: [実行ツールウィンドウ] には、何個の数字が表示されると思いますか？

9 個

書名　：Python の基本構文
発行日　：2025 年 8 月 20 日 第 1 版
著作　：ESRIジャパン株式会社
発行　：ESRIジャパン株式会社
〒102-0093 東京都千代田区平河町2-7-1 塩崎ビル
電話　：03-3222-3941
FAX　：03-3222-3946
URL　：https://www.esrij.com/
PythonBasic_352_01_20250820