Python基礎研修
Python Basic Training

1.基礎知識
2.実行環境の準備

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

はじめに
Introduction

はじめに

はじめに

- Pythonは「シンプルで学びやすいプログラミング言語」として紹介されることが多い
- しかし、一度にすべてを理解し、スムーズにプログラムを書けるようになるわけではない
- 本研修は、Pythonをこれから学び始める方を対象とした入門研修
- 研修を通じて、Pythonの基本概念を理解し、プログラミングの第一歩を踏み出すことを目指す

中級編へのステップ

- Pythonの基礎を習得後、データ分析やAIなどの応用分野にも挑戦できる
- 研修の最後に、次のステップとして役立つ学習サイトや参考資料を紹介

Pythonとライブラリ

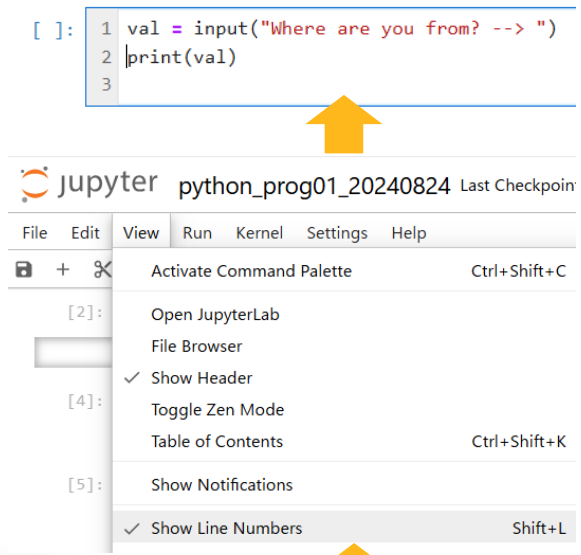
- Pythonは他のプログラミング言語と比べて使いやすいが、ゼロからすべてを作るのは大変
- 多くの便利なライブラリ（既存の関数やツールの集合）が提供されており、活用することで効率的にプログラムを作成できる
- 本研修では、基本的なライブラリの活用方法についても紹介し、実践的なスキルを身につけられるようサポート
- Pythonの学習を楽しみながら、着実にスキルを伸ばしていきましょう！

Jupyter Notebookの立ち上げ方

マッチスル

Jupyter Notebookの立ち上げ方法

- Anacondaのインストール
- Jupyter Notebookの起動方法
  - フォルダをクリックしてJupyter Notebookを開く
  - 左上の「NEW」→「Notebook」を選択
  - 「Untitled0」をクリックし、適切な名前に変更（Rename）
  - セルに 1+2 を入力し、Shift+Enter を押すと、結果 3 が表示される



1.基礎知識

Basic Knowledge

章の目次

マッチスル

基礎知識

- ✓ プログラミング言語の概要
- ✓ Pythonの起源と歴史
- ✓ Pythonの特徴
- ✓ Pythonの課題と短所
- ✓ Pythonの活用分野
- ✓ コラム：データサイエンス分野での活用

プログラミング言語の概要

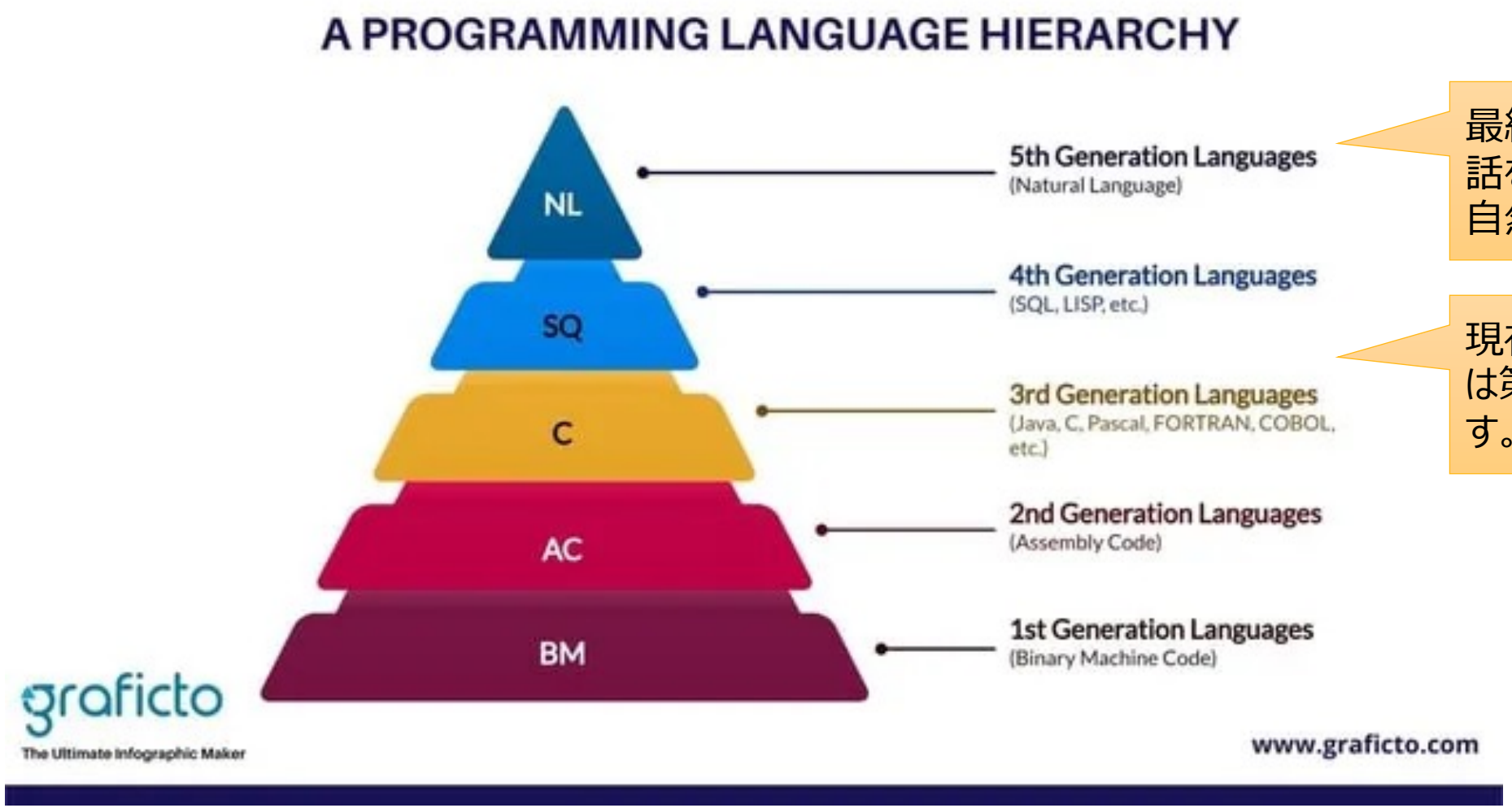
マッチスル

- **1. プログラミング言語**
  - コンピュータに指示を与えるための「言葉」として機能します。
- プログラミング言語の目的
  - コンピュータに作業を指示することです
  - コンピュータに特定の問題を解決する処理をさせる

マッチスル

- **2. プログラミング言語の分類**
- **低水準言語：**
  - 機械語やアセンブリ言語など、コンピュータのハードウェアに近い言語
  - 効率的な処理が可能であるが、プログラムが難解
- **高水準言語：**
  - 人間にとって理解しやすい言語。例：C、Python、Javaなど。
  - 抽象度が高く、開発が効率的
- **ドメイン特化型言語（DSL）：**
  - 特定の分野や用途に特化した言語
  - 例：SQL（データベース操作用）、HTML（ウェブページ作成用）

プログラミング言語の歴史・世代



最終的には、機械が人間と会話をすることになります。技術：自然言語処理

現在の多くのプログラミング言語は第3世代～第4世代言語です。

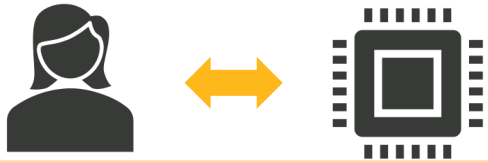
マッチスル

- **3. プログラミング言語の特徴**
- **Python：**
  - シンプルで学びやすい構文が特徴
  - データ分析や機械学習、ウェブ開発などに広く利用
- **C：**
  - システムプログラミングや組み込みシステムに強い
  - 高速で効率的な処理が可能
- **Java：**
  - オブジェクト指向プログラミングに適している
  - 大規模な企業向けシステムやモバイルアプリケーションに使われる

プログラミング言語の概要　機械語とアセンブリ語

マッチスル

プログラミングとは



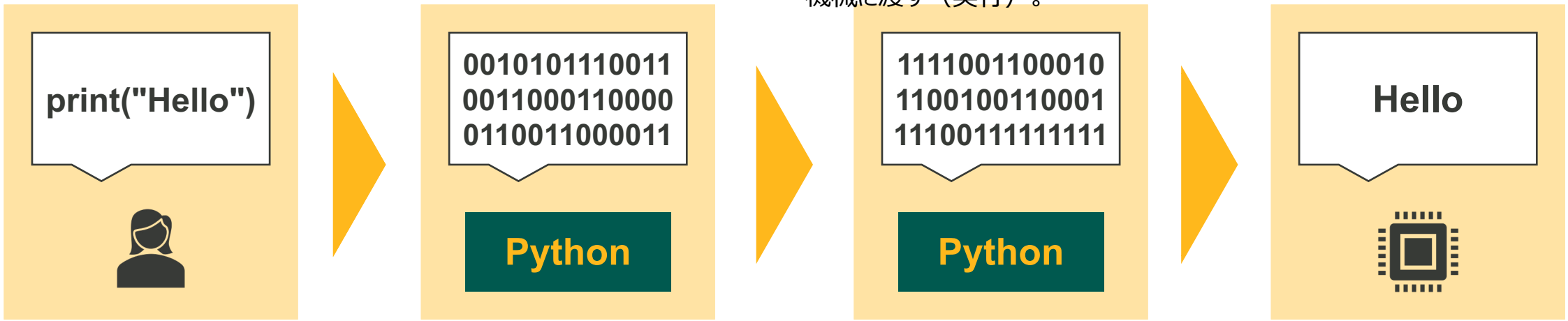
人間とコンピュータの関係

- 人間は、理解できる言語（高級言語）を使用してプログラムを作成し、コンピュータが理解できる命令に変換します。
- コンピュータは、機械語（0と1）で命令を実行しますが、人間の言葉をそのまま理解することはできません。

プログラミングはコラボレーション

- プログラミングは、人間とコンピュータのコラボレーションと捉えることができます。
- 人間はコンピュータが理解できる命令（コード）を作成し、その命令をコンピュータに実行させることで、目的を達成します。

プログラミングはコラボレーション　コンパイル：機械語に翻訳　リンク：機械語の他のライブラリとリンクし、機械に渡す（実行）。　ラン（実行）：機械の処理



マッチスル

プログラミング言語

- ✓ コンピュータに指示するときに用いられる特殊な言語
- ✓ 用途に応じて多種多様なプログラミング言語が存在

Python	**Java**	**C#**	**PHP**
Ruby	**JavaScript**	**C言語**	**など、多数存在**

Pythonは数あるプログラミング言語の一つで、
Webアプリケーション開発 や データサイエンス分野 に強い言語

Pythonの起源

マッチスル

ABC言語

数値

```
>>> WRITE 2**1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542167660429831652624386837205668069376

>>> PUT 1/(2**1000) IN x
>>> WRITE 1 + 1/x
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542
```

テキスト

Python

```python
def subtraction():
    num_1 = randint(0,9)
    num_2 = randint(0,9)

    print(f"What is {num_1} - {num_2} ?\n")

    choice = input("> ")

    if int(choice) == num_1 - num_2:
        print("Correct! Nice job! Keep on playing!\n")
        start_game()
    else:
        print(f"Incorrect...the answer is {num_1-num_2}!\n")
        start_game()

def multiplication():
    num_1 = randint(0,9)
    num_2 = randint(0,9)

    print(f"What is {num_1} x {num_2} ?\n")
```

Pythonの起源

- ✓ **1991年に登場。**
- ✓ ABC言語（教育向けに開発されていたプログラミング言語）の後継として考案。
- ✓ オランダ出身で、数学や計算機科学の研究機関に勤める**＜グイド・ヴァンロッサム＞**氏により開発。
- ✓ 言語の名称は、1970年代に流行したイギリスのコメディ番組**＜空飛ぶモンティ・パイソン＞**が由来。
- ✓ ヴァンロッサム氏はこのコメディ番組を好んで見ており、軽い気持ちで拝借したことが由来とされている。

http://www2.airnet.ne.jp/sardine/complang/abc_intro.html

プログラミングの起源と歴史

不要

1940年以前
- 微分方程式の機械設計が進行、1912年にはアナログコンピュータ開発。
- チャールズ・バベッジが機械式コンピュータを開発。
- 1946年、世界初の電子コンピュータENIACが完成。

1950年代～1960年代
- 手続き型言語が開発され、FORTRANが登場（ジョン・バッカス）。
- 日本でCOBOLが登場、BASICやPL/Iも発展。GOTO文や行番号を使用したプログラムスタイル。

1970年代
- ALGOL58が登場、構造化プログラミングが推進。
- PASCAL、FORTRAN77、C言語などが登場。
- オブジェクト指向や関数型プログラミングの概念が広がる。

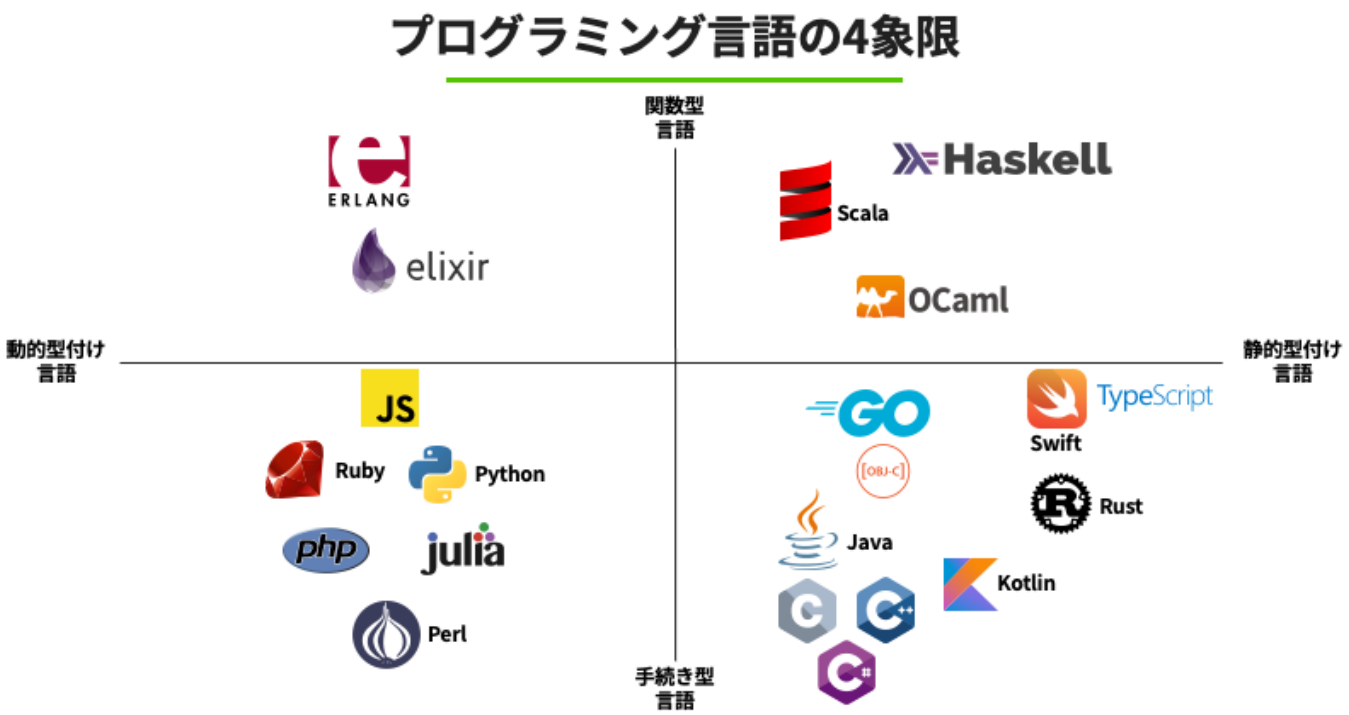
1980年代
- C++、Object型C、PostScript、Perlなどが登場。
- オブジェクト指向プログラミングが進展。

1990年代
- Java、JavaScript、Python、R言語、Rubyなどが登場。
- 大量データ、高速化、並列処理が求められるようになる。

2000年代
- C#、D言語、Scala、Go、Hadoop、Sparkなどが登場。
- スクリプト型言語が普及、開発効率と高速化が進む。

2010年代
- Julia、Rust、Kotlin、Swiftなどが登場。
- Rustはシステムプログラミング、KotlinはAndroid開発で注目。

Pythonの特徴



https://qiita.com/danwatanabe/items/b2dae4b73f6c18056f3d

静的型付け言語：変数宣言が必要
動的型付け言語：変数宣言が不要
手続き型言語・オブジェクト指向言語：コンピュータに命令したい処理を順番通りに記述する
関数型言語：処理に関数を用いた記述。

Pythonの特徴

インタプリタ言語・スクリプト言語

- ✓ **コンパイル不要**
- ✓ ソースコードを**一行ずつ翻訳しながら実行**する形式の言語。
- ✓ 確認したい箇所を、手軽に実行して確認することが可能。

オブジェクト指向言語・関数型言語・動的型付け言語

- ✓ Java言語のような**オブジェクト指向言語**
- ✓ C言語のような**関数型言語でもある**
- ✓ **変数の宣言が不要**

オープンソースソフトウェア

- ✓ ソースコードが公開されており、**誰もが自由に利用できる。**
- ✓ 世界中の支援者が日々ブラッシュアップに参加している。

豊富なライブラリ

- ✓ **標準ライブラリ**に加え、様々な分野の**外部ライブラリ**が充実。計算や統計処理、画像処理、音声処理、自然言語処理などのライブラリが多数存在。

Pythonの短所と対策

マッチスル

Pythonの短所

実行速度の遅さ

✓ 手軽に記述できる一方、C言語やJavaに比べ、実行速度は遅め。

バージョン互換性の低さ

✓ Pythonにはバージョンが存在し、現在も日々更新されている。

✓ Pythonはバージョン間の互換性があまりなく、バージョンについて認識しておくことは大切。

シンプルで簡単

✓ 多くの書籍や記事で **＜Pythonは簡単なプログラミング言語＞** と紹介されている。

✓ 他のプログラミング言語に比べ、ソースコードの記述が簡単で分かりやすいのは事実。

✓ その一方、活用できる分野の幅が広く、全ての知識を網羅的に習得することは容易ではない。

✓ Pythonを用いて大規模なシステム開発を行う場合、Javaと同程度の知識や学習量が必要。

対策

Pythonは使い方次第で高速：

- 並列処理
- 分散処理
- 遅いライブラリーは使わない
- 高速の処理を使う
- 適切なGPUやCPUを選択する
- インフラも用途で使い分ける
- 仮想マシンやVertualBox、Vmwareを使い、運用先で開発時のバージョン環境を使う

マッチスル

Pythonの活用分野

Webアプリケーション開発	データサイエンス、AI分野	教育分野
ゲーム開発	データ収集・加工	ブロックチェーン開発

様々な分野に特化した フレームワーク や ライブラリ が提供され、
強力なプログラムやサービスの作成を支援している。

Pythonで開発されたサービス例

YouTube	Instagram	Dropbox	など、多数存在

コラム：データサイエンス分野での活用

需要予測、在庫管理・仕入計画、価格予測

- ✓ カレンダー情報、気象データ、過去の実績情報、分野情報、地域情報などのデータを活用して、未来の需要を予測する
- ✓ これにより、仕入の週間計画、月間計画、年間計画を実施したり、在庫を管理する
- ✓ 需要と仕入の難易度により、価格を予測する。市場全体価格、ローカル価格を予測する
- ✓ Matlabで行われていたプログラムをpythonに変更している
- ✓ 事例：株価予測、電力量の需要予測、スーパーマーケットの価格設定の自動化

異常検知

- ✓ 時系列多変量位置データを活用して、異常の予兆の早期発見を行う
- ✓ 取引や業務のログデータ、画像、振動、音声、自然言語などのデータやBIのレポーティング結果など様々な時系列データを扱うことができる
- ✓ リアルタイムのモニタリングや予防保全を行う
- ✓ PythonやBIツールを使う
- ✓ 事例：船の監視、運用設備の監視、WEBアクセスログの監視など

章のまとめ

マッチスル

基礎知識

Pythonの起源

- ✓ 1991年に登場。
- ✓ オランダ出身のグイド・ヴァンロッサム（Guido van Rossum）氏によって開発された。

Pythonの特徴

- ✓ インタプリタ言語の一種であり、プログラムの中で確認したい箇所だけを実行して確認することができる。
- ✓ オープンソースソフトウェアの一種でソースコードが公開されており、誰もが自由に利用することができる。

Pythonの活用分野

- ✓ Webアプリケーション開発やデータサイエンス分野に特化したフレームワークやライブラリが提供されている。
- ✓ Python によって作られた代表的なサービスとして、YouTube / Instagram / Dropbox などがある。

インフラ分野での活用、データサイエンス分野での活用

- ✓ 障害発生時のログの収集や解析を代行するプログラムを、Python を用いて記述することができる。
- ✓ 近年流行りの Infrastructure as Code を実現するサービスやツールに Python が用いられている。
- ✓ 需要予測、異常量の早期発見

2.Python実行環境
Python Execution Environment

章の目次

WindowsでのPython実行環境

- ✓ 実行環境
- ✓ 環境構築
- ✓ スクリプトの実行

本研修の環境：

Anaconda3-2024.06-1

AnacondaではPythonのバージョンが指定できるが、デフォルトの場合はPython3.12.4となる

Mac OS について

- ✓ 本研修は、基本的に **Windows での受講**を想定しています。
- ✓ Mac OS の場合、Python（バージョンは若干異なる）がプリインストールされているため、本章はスキップして構いません。
- ✓ なお、Mac OS の場合は使用するアプリケーションは若干異なるため、詳しい使い方はインターネット等で検索してください。

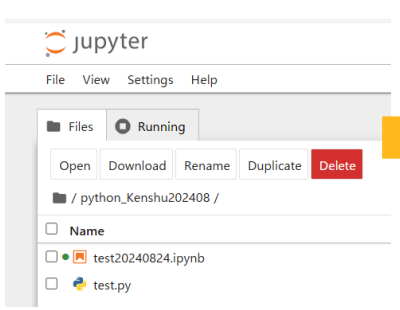
Python実行環境

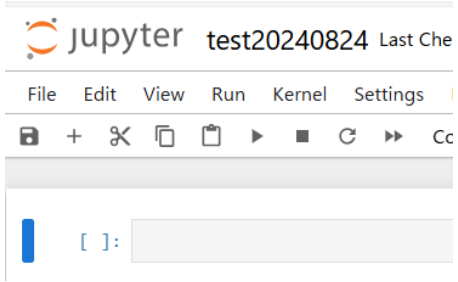
```
[2]: !python --version

Python 3.12.4
```

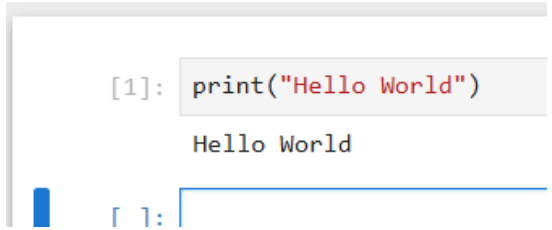
作業場所にフォルダー
python_Kenshu2024
08を作成する。

Jupyter Notebook

作業場所：C:¥Users¥社員番号¥python_Kenshu202408



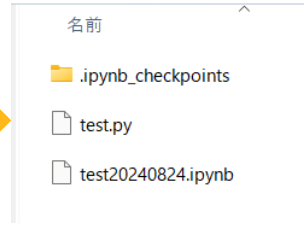




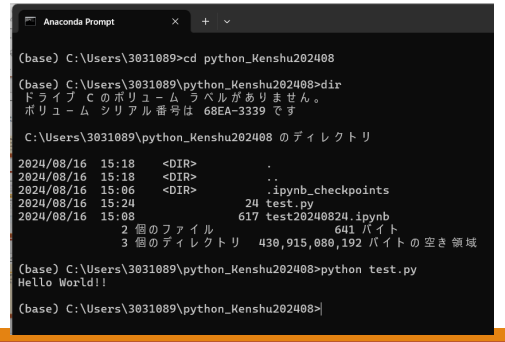
Shift Enterで実行する。

Anaconda prompt

pythonコードを作業場所に保存する



Windowsボタン→Anaconda promptを立ち上げる
作業場所に移動する
>cd python_Kenshu202408
Pythonコードを実行する
>python test.py



本研修で使用する実行環境

Python

- ✓ **Python 3.12.4**
  - ✓ 研修資源として配布したインストーラを使用

Pythonフォルダ

- ✓ C:¥Users¥社員番号¥直下にフォルダーを作成し、その中で作業する

アプリケーション（2つのやり方、今回はJupyter Notebookを使う）

- ✓ **Jupyter Notebookで編集したコードを実行**
- ✓ **テキストファイルで編集したpythonコードをanaconda promptで実行**

章のまとめ

マッチスル

実行環境

Pythonのインストール

- ✓ AnacondaでPythonをインストールすることで、自身の端末でもPythonを使えるようになる。
- ✓ Pythonのインストーラは、Anacondaの公式サイトからダウンロードすることが可能。

Pythonの実行(Jupyter Notebook)

- ✓ Jupyter Notebook。
- ✓ Print(“Hello World”)
- ✓ Shift Enterで実行

Pythonの実行コマンド（コマンドプロンプトからの実行）

- ✓ **python +（半角スペース）+（Pythonのファイル名、例：test,py）**
- ✓ 必ず、Pythonファイルのある場所で実行すること、ファイル名は拡張子まで含めて指定すること。