

# Python公式チュートリアル of 学習環境構築手順

本手順書ではPython公式チュートリアル of 学習環境構築手順について記述する。

Pythonは演習の中でも使用している為、本学習を通して事前に内容を把握することを目的としている。

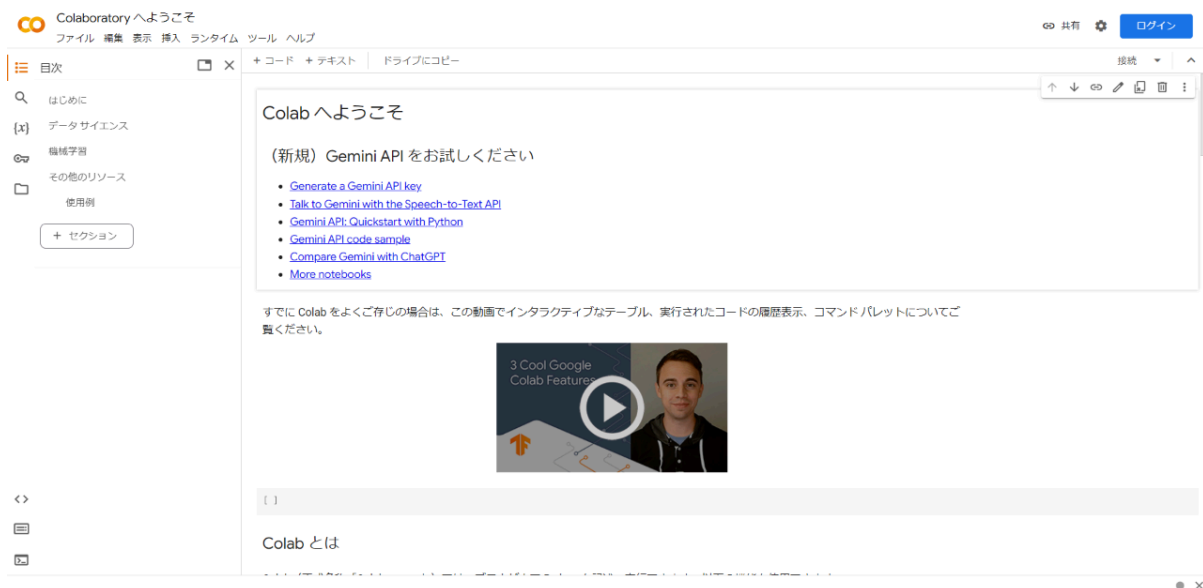
また、公式チュートリアル内のプログラムはGoogleColaboratoryにて進める為、その実行方法を説明する。

## 目次

- [1 GoogleColaboratoryログイン](#)
- [2 ノートブックの作成](#)
- [3 Python公式チュートリアルの実施](#)
- [4 他参考ドキュメント](#)

## 1 GoogleColaboratoryログイン

1. [GoogleColaboratory](#) のページへ移動



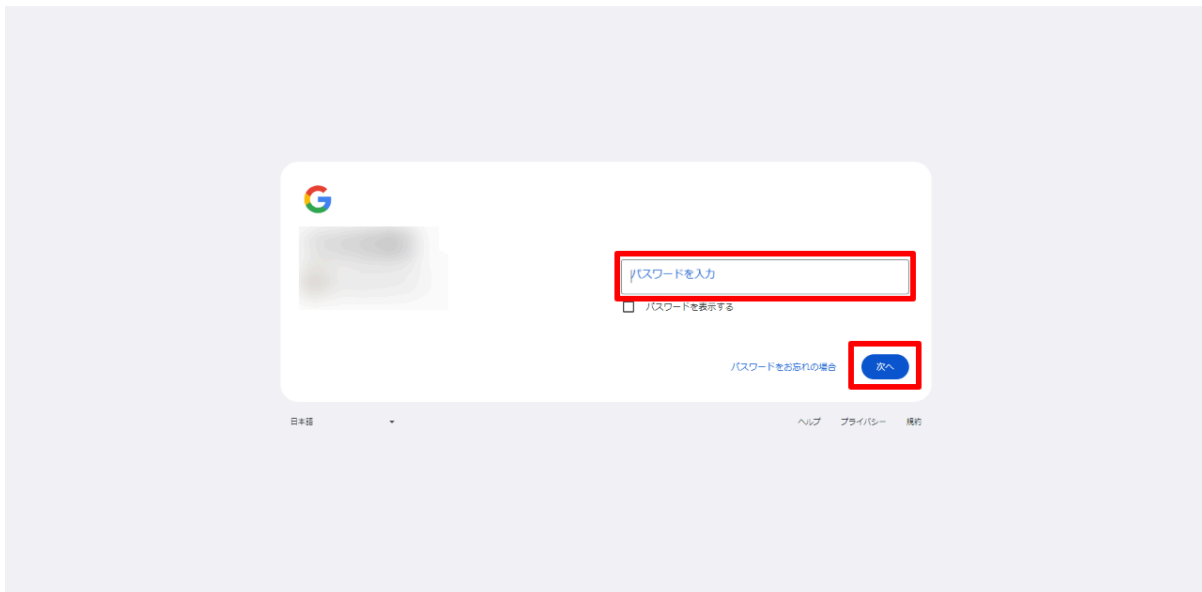
## 2. 右上の「ログイン」をクリック



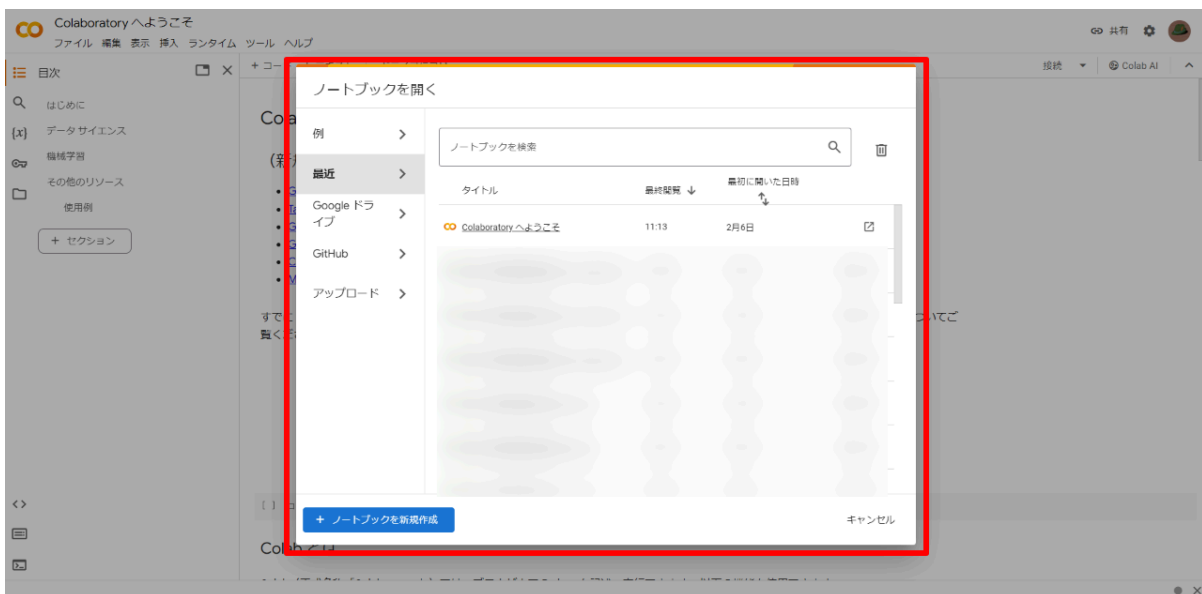
## 3. 対象のGoogleアカウントを選択



#### 4. パスワードを入力して「次へ」をクリック



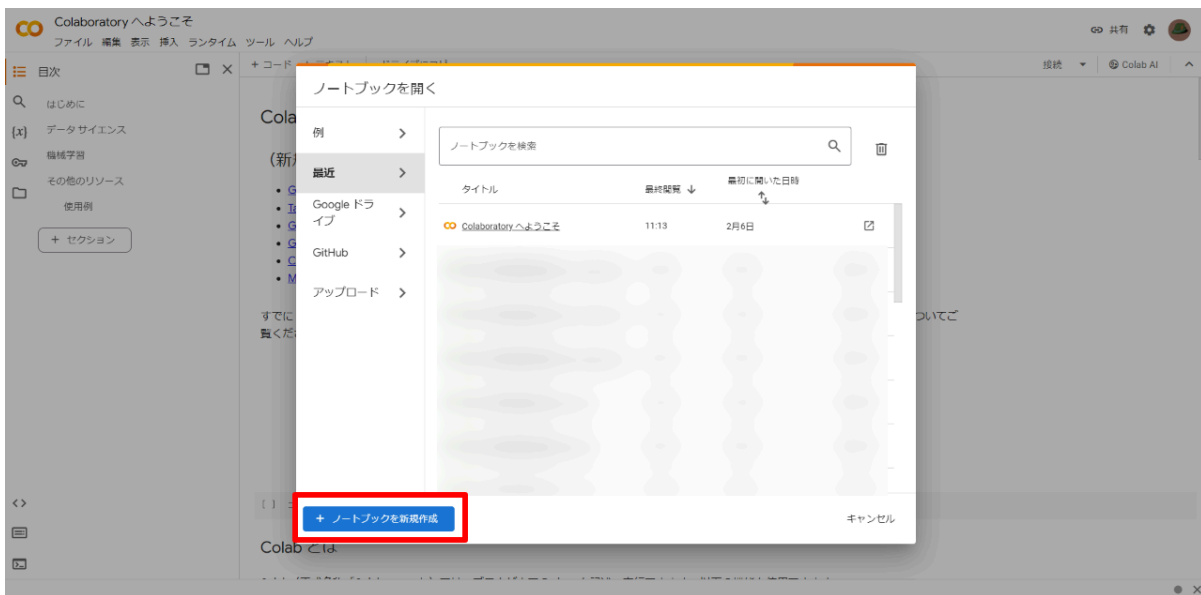
#### 5. ログインが成功するとGoogleColaboratoryの「ノートブックを開く」画面が表示される



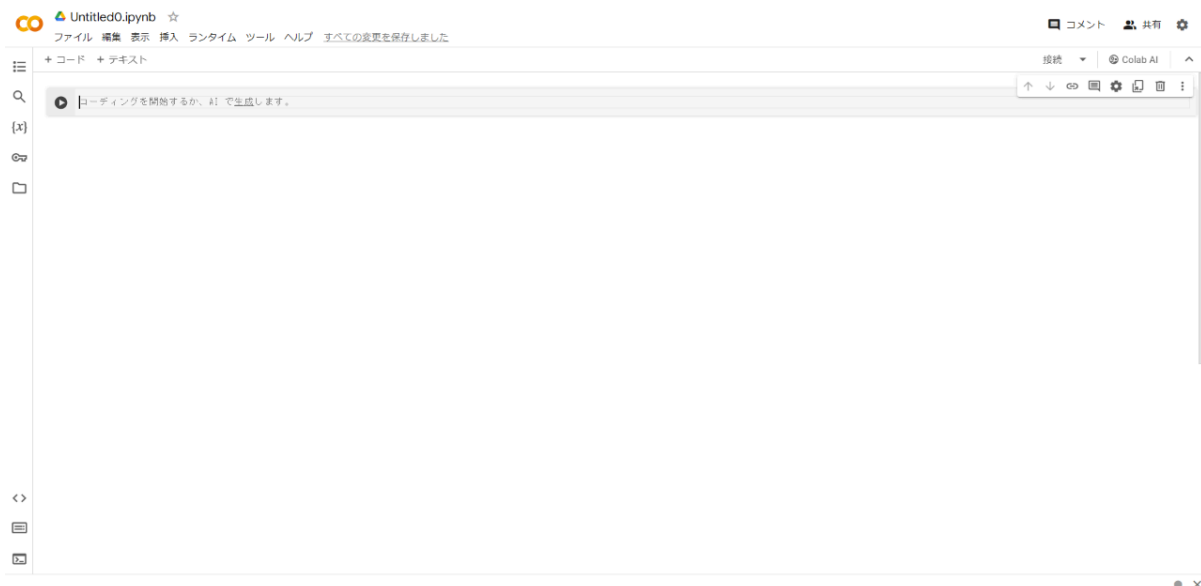
## 2 ノートブックの作成

チュートリアルに記載されているプログラムを実行する為に、ノートブックを作成する。

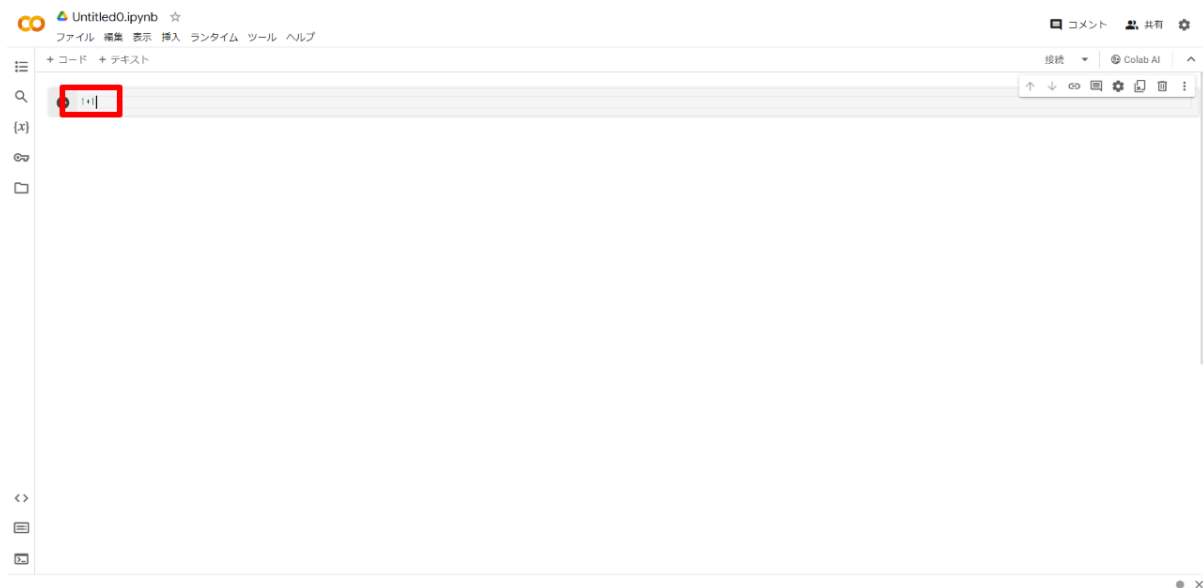
### 1. 「ノートブックを新規作成」をクリック



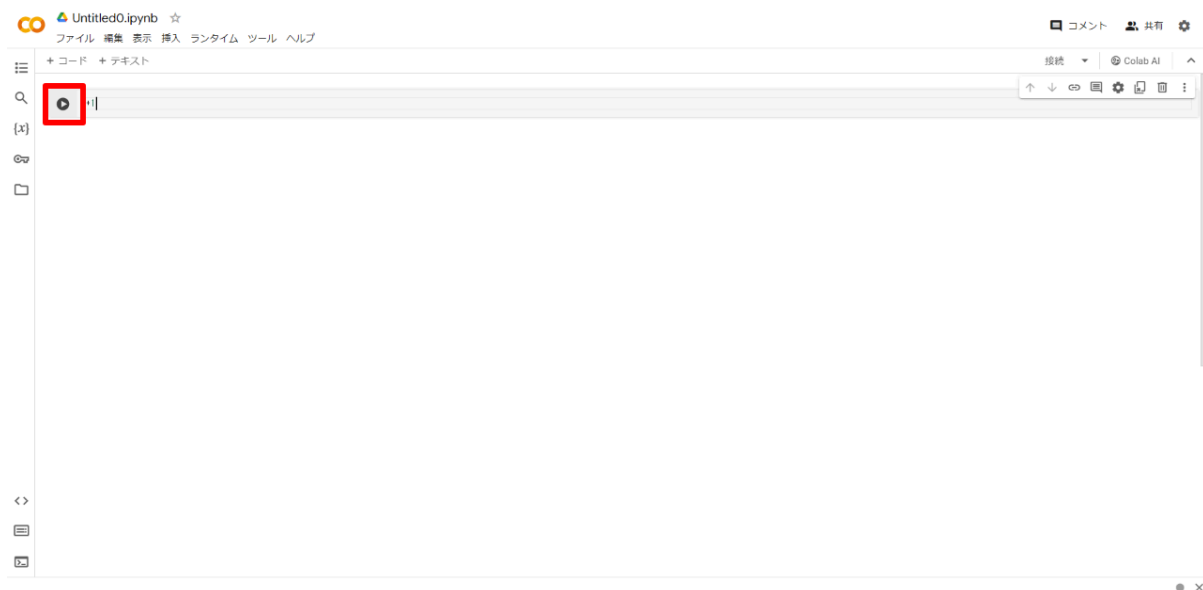
### 2. ノートブックが開く



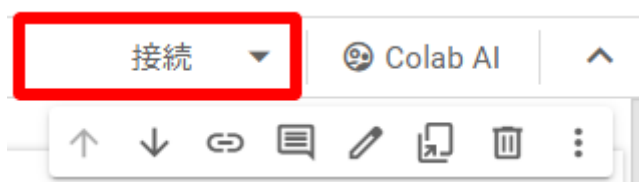
### 3. プログラムの動作を確認する為、セル内に `1+1` を入力



### 4. 実行ボタンをクリック



※初回実行時は少し時間がかかるので、画面右上のステータスが「RAMディスク」になるまで待機

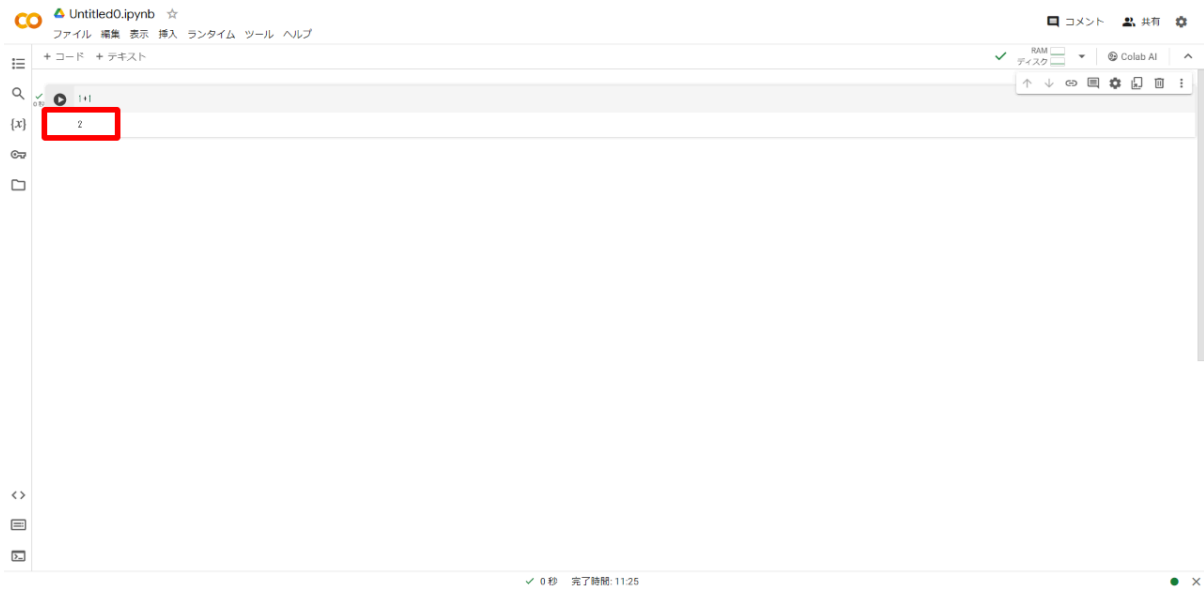


1. 「接続」→ プログラム実行前
2. 「接続中」→ プログラム実行ボタンをクリック
3. 「接続済」→ プログラム実行準備完了

4. 「RAMディスク」 → プログラム実行開始

5. プログラムの実行結果として、 2 という結果が表示されることを確認

※公式チュートリアルではこの手順 3、4、5 を実施してプログラムの確認を行う



## 3 Python公式チュートリアルの実施

[Python公式チュートリアル](#)に記載されている内容を実施する。

実施内容は1章から16章まであり、特に重要な項目は以下の通りだが、可能であれば全ての章を実施すること。

また、学習の際は、必ず記載されているプログラムを実行して結果を確認しながら進めること。

- 3.形式ばらない Python の紹介
  - 3.1. Python を電卓として使う
    - 3.1.1. 数
    - 3.1.2. テキスト
    - 3.1.3. リスト型 (list)
  - 3.2. プログラミングへの第一歩
- 4.その他の制御フローツール
  - 4.1. if 文
  - 4.2. for 文
- 5.データ構造
  - 5.1. リスト型についてもう少し
  - 5.5. 辞書型 (dictionary)
  - 5.6. ループのテクニック
  - 5.7. 条件についてもう少し
- 6.モジュール
  - 6.1. モジュールについてもう少し
- 7.入力と出力
  - 7.1. 出力を見やすくフォーマットする
    - 7.1.1. フォーマット済み文字列リテラル
    - 7.1.2. 文字列の format() メソッド
    - 7.1.3. 文字列の手作業でのフォーマット
    - 7.1.4. 古い文字列書式設定方法

# プログラム実行手順について

チュートリアルに記載されているプログラムの実行手順を以下に記載する。

## 1. プログラムが記載されているチュートリアルを開く



## 2. プログラム記載エリア右上の「>>>」をクリック





### 3. 記載されているプログラムから「>>>」や「...」の不要な文字が除外される

Python » Japanese » 3.12.2 » 3.12.2 Documentation » Python チュートリアル » 4. その他の制御フローツール

Theme Auto | クイック検索 | 検索 | 前へ | 次へ | モジュール | 索引

#### 4. その他の制御フローツール

前章で紹介した [while](#) 文の他にも、Python にはいくつか制御フローツールがあり、本章で説明します。

##### 4.1. if 文

おそらく最もおなじみの文型は [if](#) 文でしょう。例えば：

```
x = int(input("Please enter an integer: "))

if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

ゼロ個以上の [elif](#) 部を使うことができ、[else](#) 部を付けることもできます。キーワード '[elif](#)' は '[else if](#)' を短くしたもので、過剰なインデントを避けるのに役立ちます。一連の [if ... elif ... elif ...](#) は、他の言語における [switch](#) 文や [case](#) 文の代用となります。

いくつかの定数と同じ値かを比較する場合や、特定の型や属性を確認する場合には、[match](#) 文が便利です。詳細は [match 文](#) を参照してください。

... ..

### 4. 手順7にて、ここに記載されているプログラムを書き写す為、この画面は閉じないでおく

Python » Japanese » 3.12.2 » 3.12.2 Documentation » Python チュートリアル » 4. その他の制御フローツール

Theme Auto | クイック検索 | 検索 | 前へ | 次へ | モジュール | 索引

#### 4. その他の制御フローツール

前章で紹介した [while](#) 文の他にも、Python にはいくつか制御フローツールがあり、本章で説明します。

##### 4.1. if 文

おそらく最もおなじみの文型は [if](#) 文でしょう。例えば：

```
x = int(input("Please enter an integer: "))

if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

ゼロ個以上の [elif](#) 部を使うことができ、[else](#) 部を付けることもできます。キーワード '[elif](#)' は '[else if](#)' を短くしたもので、過剰なインデントを避けるのに役立ちます。一連の [if ... elif ... elif ...](#) は、他の言語における [switch](#) 文や [case](#) 文の代用となります。

いくつかの定数と同じ値かを比較する場合や、特定の型や属性を確認する場合には、[match](#) 文が便利です。詳細は [match 文](#) を参照してください。

... ..

## 5. 前の手順で開いたGoogleColaboratoryのノートブックを開く



## 6. 左上の「+ コード」をクリック



## 7. 追加されたセルに手順4のプログラムを書き写す



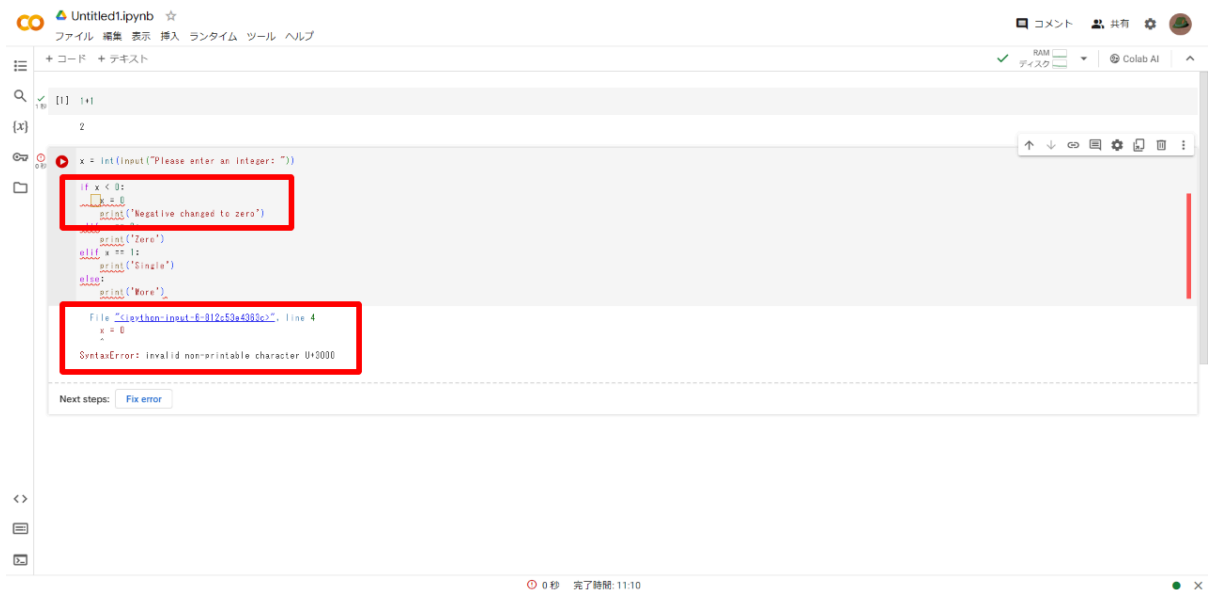
A screenshot of a Jupyter Notebook interface. The top bar shows 'Untitled1.ipynb' and various menu options. The left sidebar has icons for file explorer, search, and other tools. The main area contains a code cell with the following Python code:

```
[1] In:
2
x = int(input("Please enter an integer: "))
if x < 0:
    x = 0
    print("Negative changed to zero")
elif x == 0:
    print("Zero")
elif x == 1:
    print("Single")
else:
    print("More")
|
```

The code cell is highlighted with a red box. The output area below the code cell shows the number '2'. The status bar at the bottom indicates '0秒 完了時間: 16:09'.

※プログラムを書き写す際、インデント(行頭のスペース)に全角スペースを用いたり、異なる数の半角スペースを入力するとエラーとなる為、正確に書き写すこと

- 全角スペースを入力した場合



A screenshot of a Jupyter Notebook interface showing a syntax error. The code cell contains the same Python code as the previous screenshot, but with a full-width space (全角スペース) used for indentation on line 4:

```
[1] In:
2
x = int(input("Please enter an integer: "))
if x < 0:
    x = 0
    print("Negative changed to zero")
elif x == 0:
    print("Zero")
elif x == 1:
    print("Single")
else:
    print("More")
|
```

The code cell is highlighted with a red box. The output area below the code cell shows a syntax error message:

```
File "C:\Python\ipynb-0-312c53a4383c2", line 4
x = 0
^
SyntaxError: invalid non-printable character U+3000
```

The error message is highlighted with a red box. The status bar at the bottom indicates '0秒 完了時間: 11:10'.

- 異なる数の半角スペースを入力した場合

```
def func(x):  
    x = int(input("Please enter an integer: "))  
    if x < 0:  
        print('negative changed to zero')  
    elif x == 0:  
        print('Zero')  
    elif x == 1:  
        print('Single')  
    else:  
        print('More')  
    print('done')
```

File ~/env/lib/python3.9.7/site-packages/line\_profiler/hooks.py, line 5:  
print('negative changed to zero')  
^  
IndentationError: unexpected indent

Next steps: [Explain error](#)

## 8. 実行ボタンをクリック

```
def func(x):  
    x = int(input("Please enter an integer: "))  
    if x < 0:  
        print('negative changed to zero')  
    elif x == 0:  
        print('Zero')  
    elif x == 1:  
        print('Single')  
    else:  
        print('More')  
    print('done')
```

## 9. 実行結果が表示されるので確認

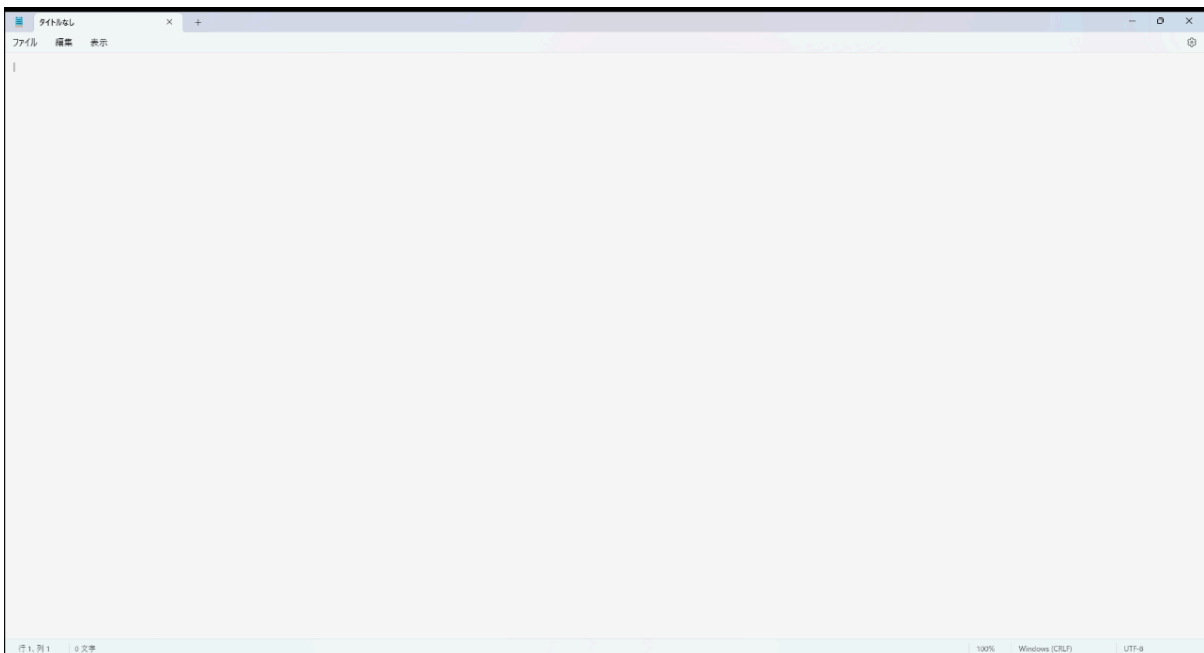


## モジュールの作成について

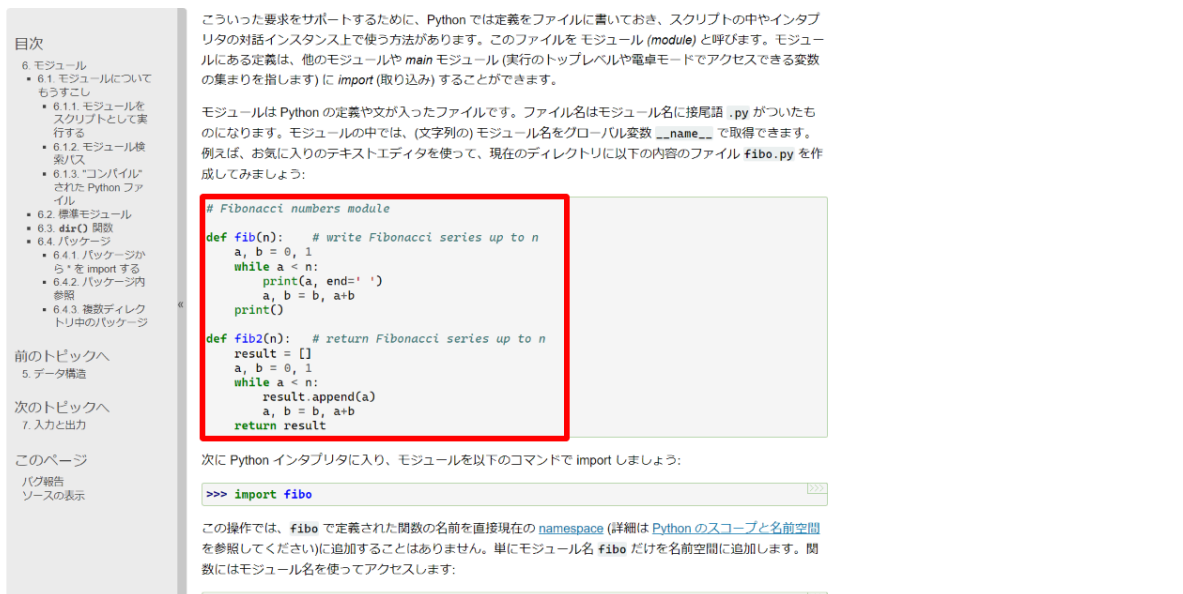
6章のモジュールの説明の中で、簡単なモジュールを作成する必要がある。  
GoogleColaboratoryで実行する場合は、特別な手順の実施が必要な為、その手順を記載する。

### モジュールの作成

#### 1. テキストエディタ（メモ帳など）を開く



## 2. 「[6.モジュール](#)」を開き、以下のプログラムを確認



目次

- 6. モジュール
  - 6.1. モジュールについて
    - 6.1.1. モジュールをスクリプトとして実行する
    - 6.1.2. モジュール検索パス
    - 6.1.3. "コンパイル"された Python ファイル
  - 6.2. 標準モジュール
  - 6.3. `dir()` 関数
  - 6.4. パッケージ
    - 6.4.1. パッケージから `*` を import する
    - 6.4.2. パッケージ内参照
    - 6.4.3. 複数ディレクトリ中のパッケージ

前のトピックへ  
5. データ構造

次のトピックへ  
7. 入力と出力

このページ  
バグ報告  
ソースの表示

こういった要求をサポートするために、Python では定義をファイルに書いておき、スクリプトの中やインタプリタの対話インスタンス上で使う方法があります。このファイルをモジュール (*module*) と呼びます。モジュールにある定義は、他のモジュールや *main* モジュール (実行のトップレベルや電卓モードでアクセスできる変数の集まりを指します) に `import` (取り込み) することができます。

モジュールは Python の定義や文が入ったファイルです。ファイル名はモジュール名に接尾語 `.py` がついたものになります。モジュールの中では、(文字列の) モジュール名をグローバル変数 `__name__` で取得できます。例えば、お気に入りのテキストエディタを使って、現在のディレクトリに以下の内容のファイル `fibonacci.py` を作成してみましょう:

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

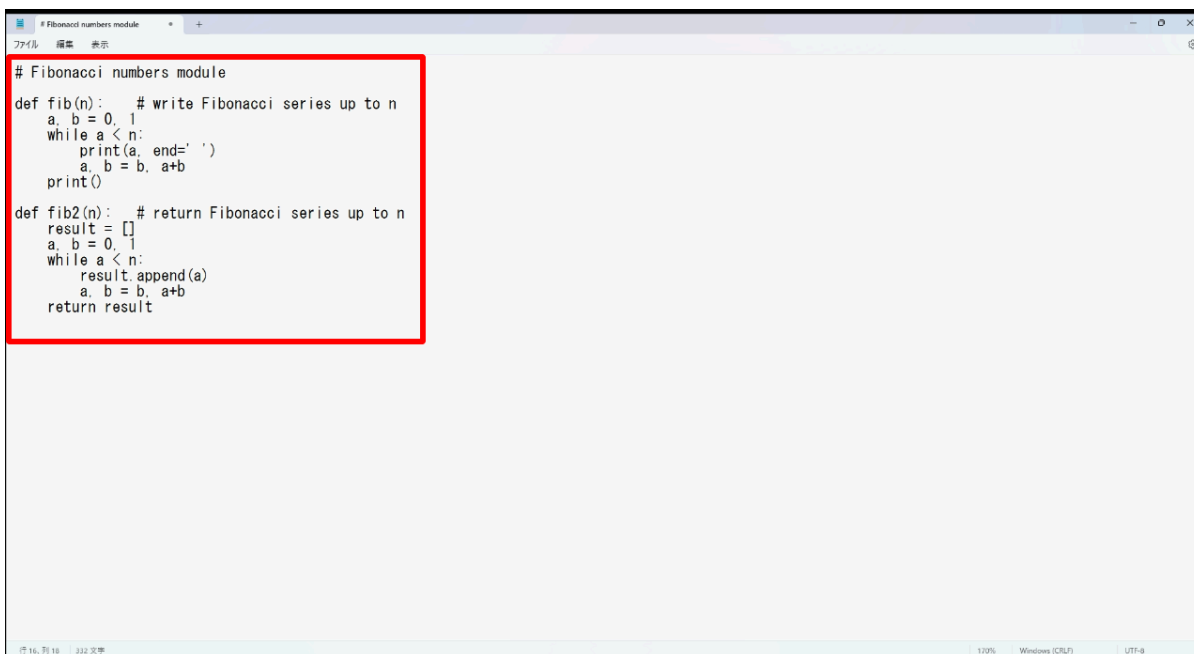
def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

次に Python インタプリタに入り、モジュールを以下のコマンドで import しましょう:

```
>>> import fibonacci
```

この操作では、`fibonacci` で定義された関数の名前を直接現在の `namespace` (詳細は [Python のスコープと名前空間](#) を参照してください) に追加することはありません。単にモジュール名 `fibonacci` だけを名前空間に追加します。関数にはモジュール名を使ってアクセスします:

## 3. 手順1で開いたテキストエディタに手順2のプログラムを書き写す



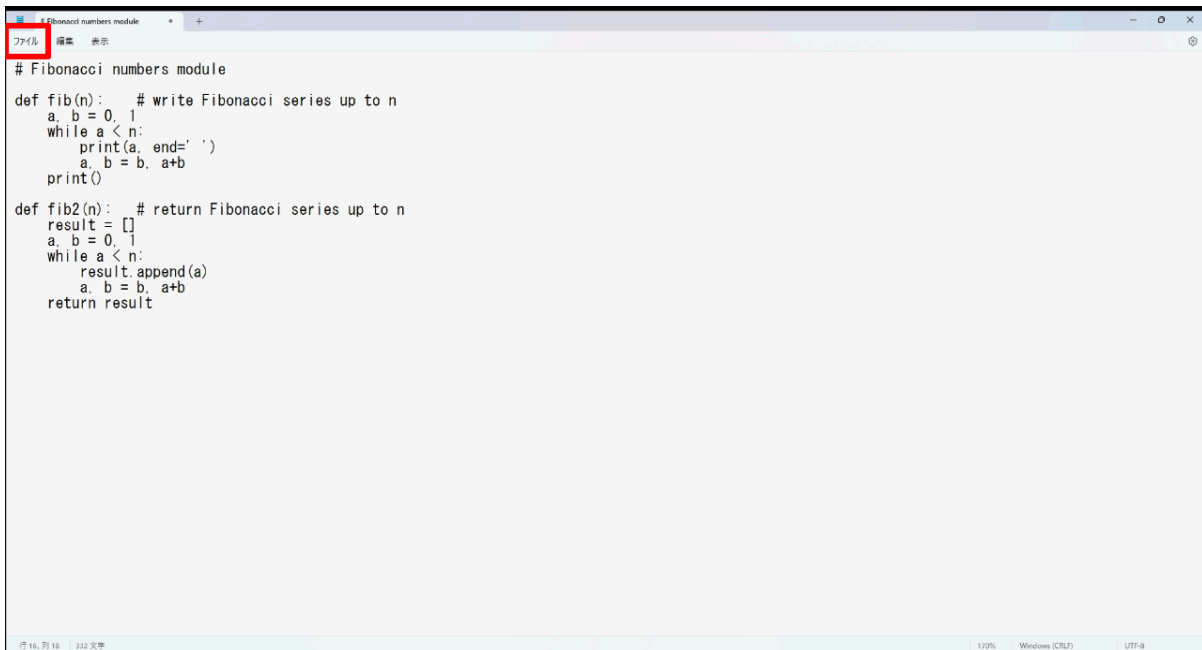
```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

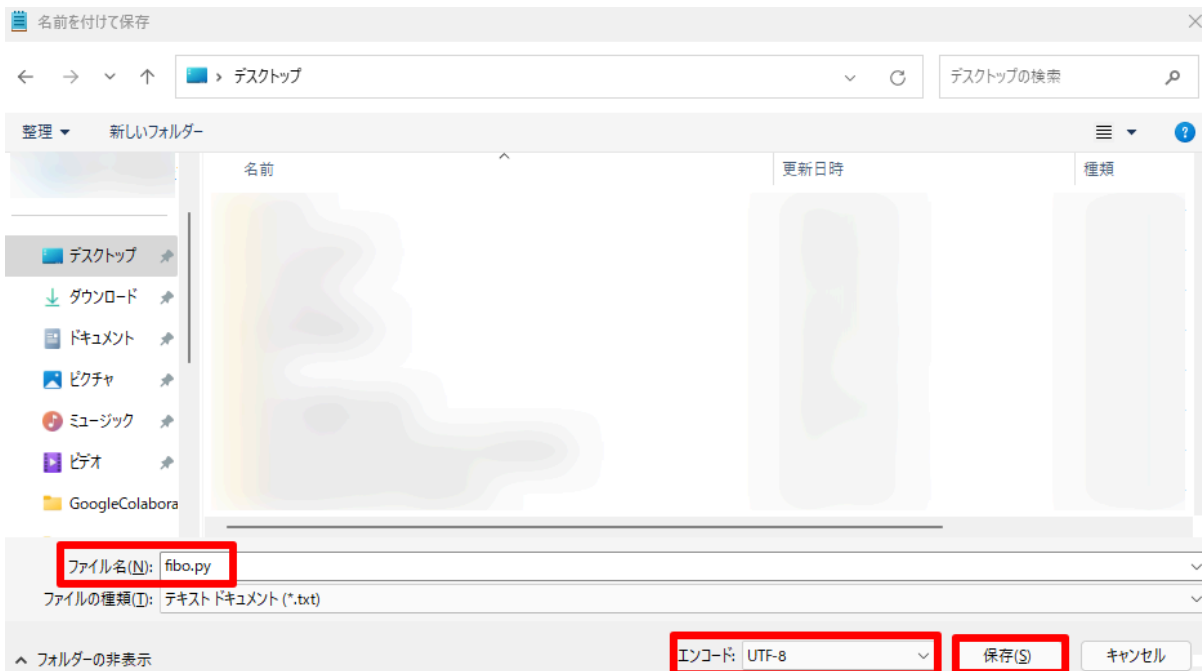
※ 「[プログラム実行手順について](#)」の手順7に記載の通り、プログラムは正確に書き写すこと

#### 4. テキストエディタ左上の「ファイル」から「名前を付けて保存」をクリック



#### 5. 任意の場所（画像ではデスクトップを設定）に以下の内容を設定して「保存」ボタンをクリック

- ファイル名 : fibo.py
- エンコード : UTF-8



# モジュールのアップロード

## 1. GoogleColaboratoryを開き、左側のファイルマークをクリック

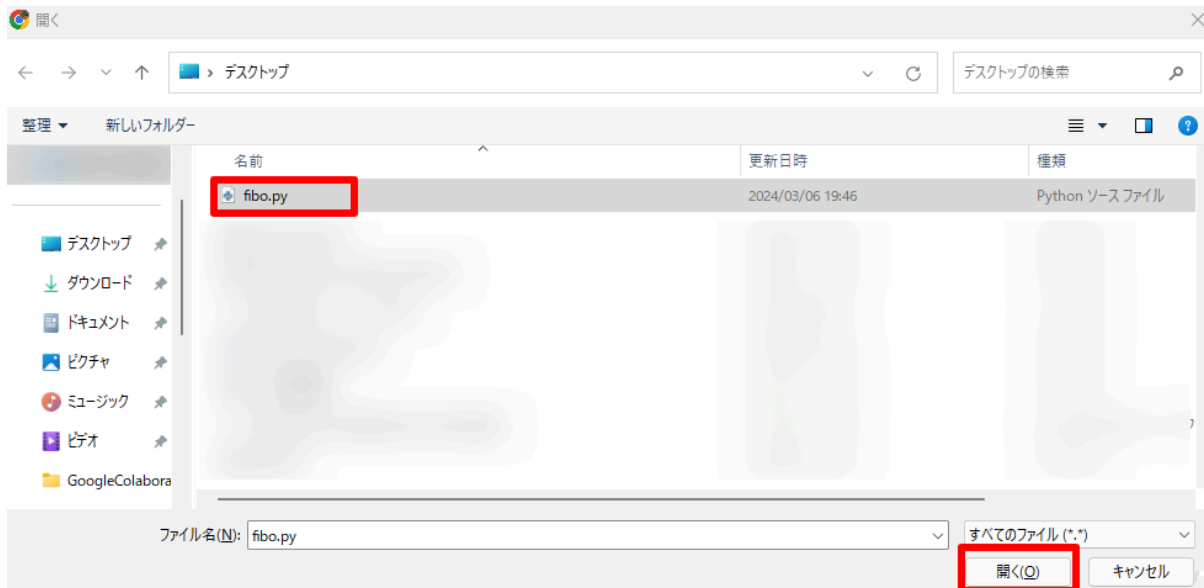


## 2. ファイル一覧の上のアップロードボタンをクリック

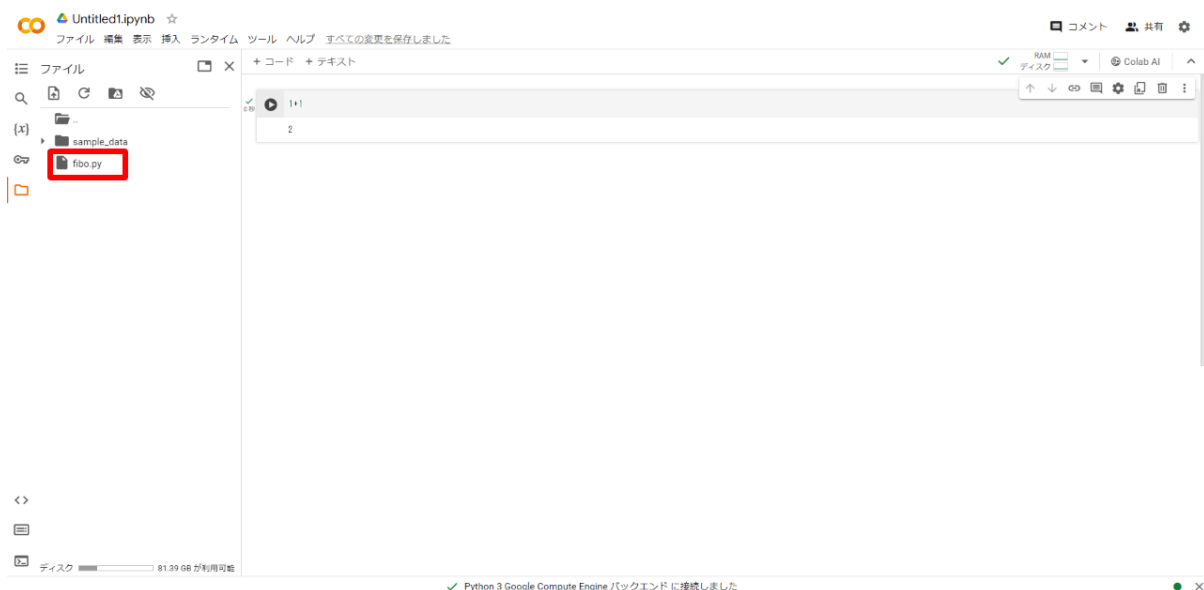




3. ファイルの選択画面が表示されるので、「[モジュールの作成](#)」の手順5で作成したfibo.pyを選択して「開く」ボタンをクリック



4. ファイル一覧にfibo.pyがアップロードされていることを確認



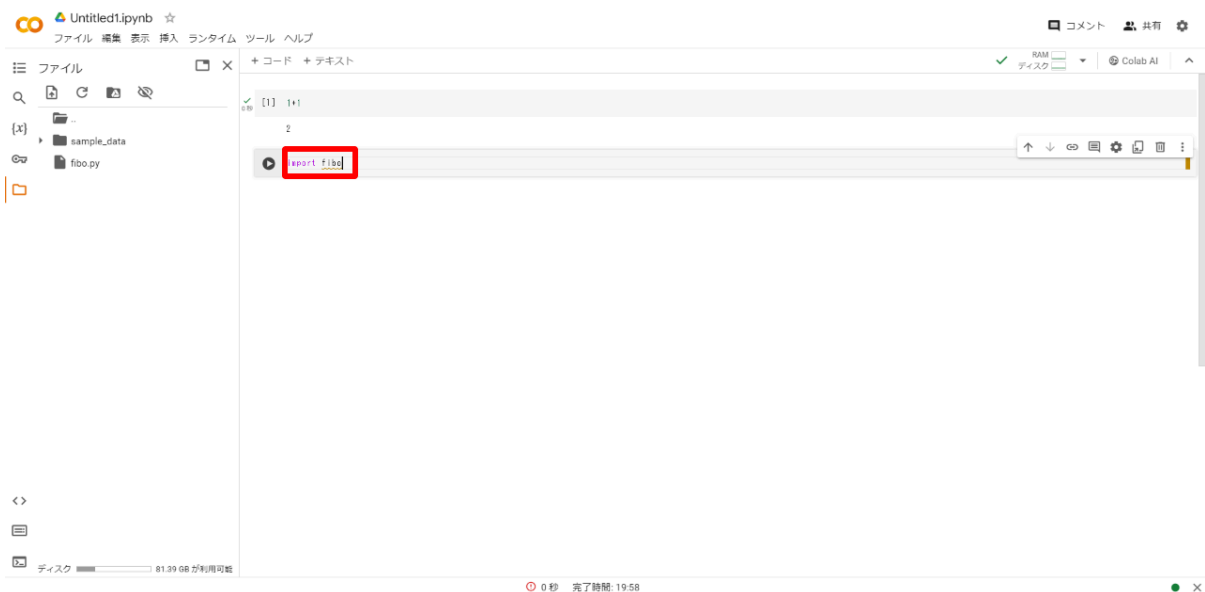
## モジュールのインポート

別ファイルで定義した関数を使えるようにする為に、モジュールのインポートが必要となる。

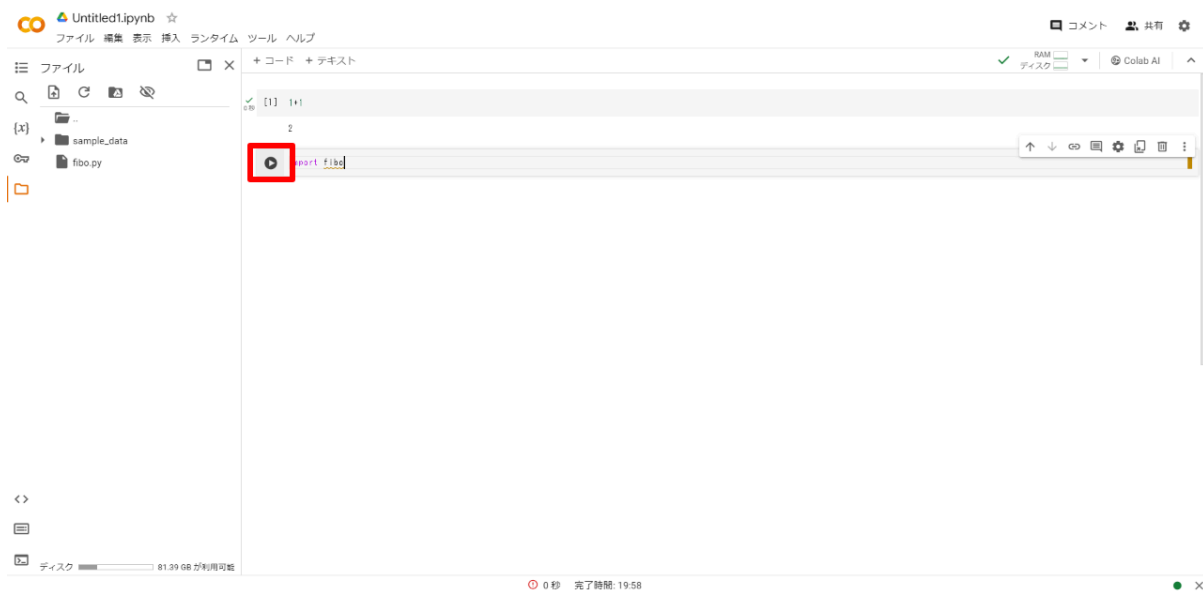
### 1. 左上の「+ コード」をクリック



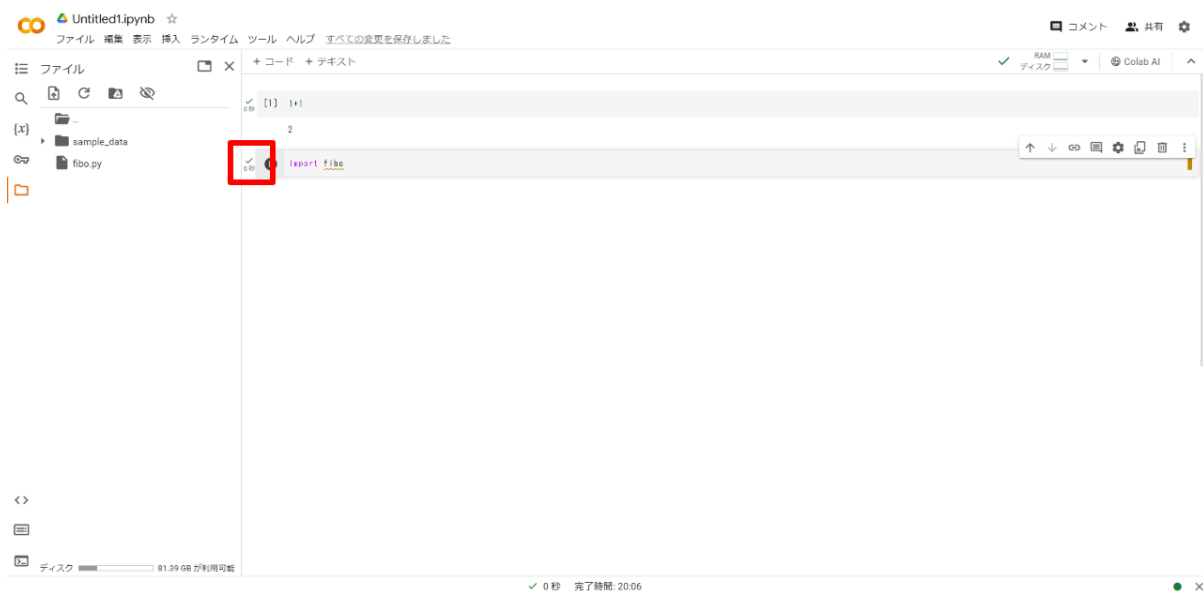
### 2. 追加されたセルに `import fibo` と入力



### 3. 実行ボタンをクリック



### 4. モジュールのインポートに成功するとセルの左側にチェックマークがつく



# チュートリアル実行手順

本手順書では、例として「4.1. if 文」と「4.2. for 文」の進め方についてのみ記載するが、その他のチュートリアル項目も同様の進め方で実施すること。

## 4.1. if 文

### 1. Python公式チュートリアル「[4.1. if 文](#)」へ移動

Python » Japanese » 3.12.2 » 3.12.2 Documentation » Python チュートリアル » 4. その他の制御フローツール

Theme Auto | クイック検索 | 検索 | 前へ | 次へ | モジュール | 索引

目次

4. その他の制御フローツール

- 4.1. if 文
- 4.2. for 文
- 4.3. range() 関数
- 4.4. break 文と continue 文とループの else 節
- 4.5. pass 文
- 4.6. match 文
- 4.7. 関数を定義する
- 4.8. 関数定義についてもう少し
  - 4.8.1. デフォルトの引数値
  - 4.8.2. キーワード引数
  - 4.8.3. 特殊なパラメータ
    - 4.8.3.1. 位置またはキーワード引数
    - 4.8.3.2. 位置専用引数
    - 4.8.3.3. キーワード専用引数
  - 4.8.3.4. 関数の例
  - 4.8.3.5. 変数
- 4.8.4. 任意引数リスト
- 4.8.5. 引数リストのアンパック
- 4.8.6. ラムダ式
- 4.8.7. ドキュメント

## 4. その他の制御フローツール

前章で紹介した [while](#) 文の他にも、Python にはいくつか制御フローツールがあり、本章で説明します。

### 4.1. if 文

おそらく最もおなじみの文型は [if](#) 文でしょう。例えば:

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

ゼロ個以上の [elif](#) 部を使うことができ、[else](#) 部を付けることもできます。キーワード '[elif](#)' は '[else if](#)' を短くしたもので、過剰なインデントを避けるのに役立ちます。一連の [if ... elif ... elif ...](#) は、他の言語における [switch](#) 文や [case](#) 文の代用となります。

いくつかの定数と同じ値かを比較する場合や、特定の型や属性を確認する場合には、[match](#) 文が便利です。詳細は [match 文](#) を参照してください。

### 2. 説明文を読んだ上でプログラムの内容を確認

Python » Japanese » 3.12.2 » 3.12.2 Documentation » Python チュートリアル » 4. その他の制御フローツール

Theme Auto | クイック検索 | 検索 | 前へ | 次へ | モジュール | 索引

目次

4. その他の制御フローツール

- 4.1. if 文
- 4.2. for 文
- 4.3. range() 関数
- 4.4. break 文と continue 文とループの else 節
- 4.5. pass 文
- 4.6. match 文
- 4.7. 関数を定義する
- 4.8. 関数定義についてもう少し
  - 4.8.1. デフォルトの引数値
  - 4.8.2. キーワード引数
  - 4.8.3. 特殊なパラメータ
    - 4.8.3.1. 位置またはキーワード引数
    - 4.8.3.2. 位置専用引数
    - 4.8.3.3. キーワード専用引数
  - 4.8.3.4. 関数の例
  - 4.8.3.5. 変数
- 4.8.4. 任意引数リスト
- 4.8.5. 引数リストのアンパック
- 4.8.6. ラムダ式
- 4.8.7. ドキュメント

## 4. その他の制御フローツール

前章で紹介した [while](#) 文の他にも、Python にはいくつか制御フローツールがあり、本章で説明します。

### 4.1. if 文

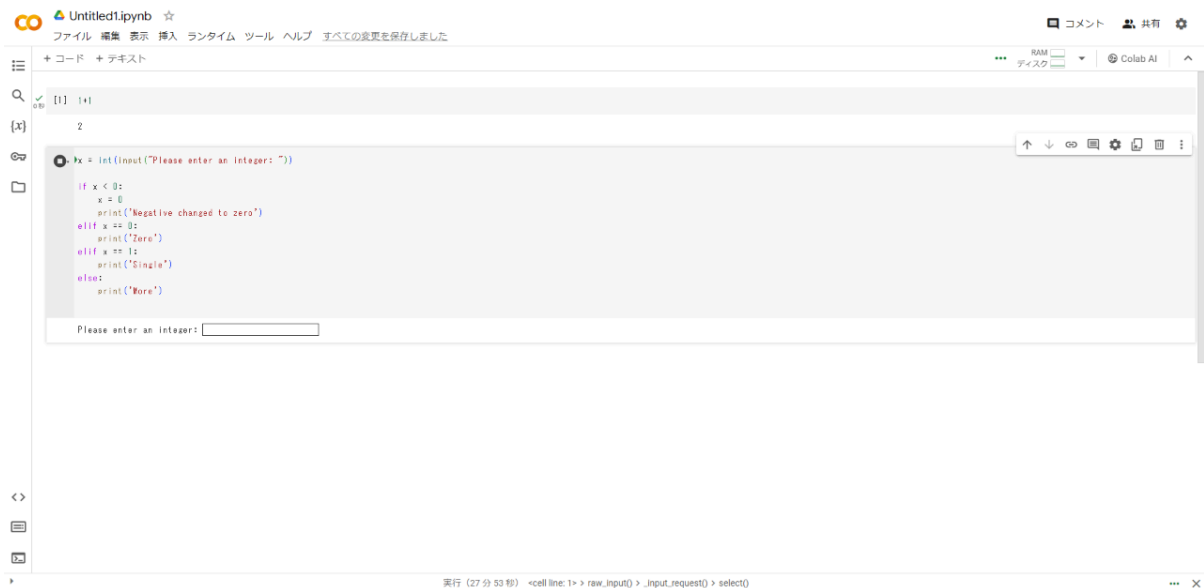
おそらく最もおなじみの文型は [if](#) 文でしょう。例えば:

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

ゼロ個以上の [elif](#) 部を使うことができ、[else](#) 部を付けることもできます。キーワード '[elif](#)' は '[else if](#)' を短くしたもので、過剰なインデントを避けるのに役立ちます。一連の [if ... elif ... elif ...](#) は、他の言語における [switch](#) 文や [case](#) 文の代用となります。

いくつかの定数と同じ値かを比較する場合や、特定の型や属性を確認する場合には、[match](#) 文が便利です。詳細は [match 文](#) を参照してください。

### 3. 「[プログラム実行手順について](#)」の手順に従い、プログラムを実行



```
Untitled1.ipynb ☆
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ すべての変更を保存しました
+ コード + テキスト
[1] 1+1
2
In [ ]: x = int(input("Please enter an integer: "))
if x < 0:
    x = 0
    print("Negative changed to zero")
elif x == 0:
    print("Zero")
elif x == 1:
    print("Single")
else:
    print("More")
Please enter an integer: 
```

### 4. 表示されたテキストボックスに任意の数字を入力してEnterキーを押す



```
Untitled1.ipynb ☆
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ すべての変更を保存しました
+ コード + テキスト
[1] 1+1
2
In [ ]: x = int(input("Please enter an integer: "))
if x < 0:
    x = 0
    print("Negative changed to zero")
elif x == 0:
    print("Zero")
elif x == 1:
    print("Single")
else:
    print("More")
Please enter an integer: 1
```

### 5. 入力された数字によって以下の結果が出力されることを確認

- 入力された数字が0より小さい場合 : Negative changed to zero
- 入力された数字が0の場合 : Zero
- 入力された数字が1の場合 : Single
- 入力された数字が1より大きい場合 : More



```
Untitled1.ipynb ☆
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

+ コード + テキスト

[1] 1+1
2

x = int(input("Please enter an integer: "))

if x < 0:
    x = 0
    print("Negative changed to zero")
elif x == 0:
    print("Zero")
elif x == 1:
    print("Single")
else:
    print("Word")

Please enter an integer: 1
Single
```

## 4.2. for 文

### 1. Python公式チュートリアルの「[4.2. for 文](#)」へ移動

目次

- 4. その他の制御フローツール
  - 4.1. if 文
  - 4.2. for 文
  - 4.3. range() 関数
  - 4.4. break 文と continue 文とループの else 節
  - 4.5. pass 文
  - 4.6. match 文
  - 4.7. 関数を定義する
  - 4.8. 関数定義についてもう少し
    - 4.8.1. デフォルトの引数値
    - 4.8.2. キーワード引数
    - 4.8.3. 特殊なパラメータ
      - 4.8.3.1. 位置またはキーワード引数
      - 4.8.3.2. 位置専用引数
      - 4.8.3.3. キーワード専用引数
      - 4.8.3.4. 関数の例
      - 4.8.3.5. 変数
    - 4.8.4. 任意引数リスト
    - 4.8.5. 引数リストのアンパシクル
    - 4.8.6. ラムダ式
    - 4.8.7. ドキュメンテーション文字列
    - 4.8.8. 関数のアンデーション
  - 4.9. 関数: コーディ

#### 4.2. for 文

Python の `for` 文は、読者が C 言語や Pascal 言語で使われているかもしれない `for` 文とは少し違います。(Pascal のように) 常に算術型の数値にわたる反復を行ったり、(C のように) 繰返しステップと停止条件を両方ともユーザが定義できるようにするのは違い、Python の `for` 文は、任意のシーケンス型 (リストまたは文字列) にわたって反復を行います。反復の順番はシーケンス中に要素が現れる順番です。例えば:

```
>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrate']
... for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

コレクションオブジェクトの値を反復処理をしているときに、そのコレクションオブジェクトを変更するコードは理解するのが面倒になり得ます。そうするよりも、コレクションオブジェクトのコピーに対して反復処理をするか、新しいコレクションオブジェクトを作成する方が通常は理解しやすいです:

```
# Create a sample collection
users = {'Hans': 'active', 'Éléonore': 'inactive', '景太郎': 'active'}

# Strategy: Iterate over a copy
for user, status in users.copy().items():
    if status == 'inactive':
        del users[user]

# Strategy: Create a new collection
active_users = {}
for user, status in users.items():
    if status == 'active':
        active_users[user] = status
```

#### 4.3 range() 関数

## 2. 説明文を読んだ上でプログラムの内容を確認

目次

- 4. その他の制御フローツール
- 4.1. if 文
- 4.2. for 文
- 4.3. range() 関数
- 4.4. break 文と continue 文とループの else 節
- 4.5. pass 文
- 4.6. match 文
- 4.7. 関数を定義する
- 4.8. 関数定義についてもう少し
  - 4.8.1. デフォルトの引数値
  - 4.8.2. キーワード引数
  - 4.8.3. 特殊なパラメータ
    - 4.8.3.1. 位置またはキーワード引数
    - 4.8.3.2. 位置専用引数
    - 4.8.3.3. キーワード専用引数
    - 4.8.3.4. 関数の引
    - 4.8.3.5. 変約
  - 4.8.4. 任意引数リスト
  - 4.8.5. 引数リストのアンパック
  - 4.8.6. ラムダ式
  - 4.8.7. ドキュメントーション文字列
  - 4.8.8. 関数のアンダースコア
- 4.9. 関数: コーディ

### 4.2. for 文

Python の `for` 文は、読者が C 言語や Pascal 言語で使っているかもしれない `for` 文とは少し違います。(Pascal のように) 常に算術型の数列にわたる反復を行ったり、(C のように) 繰返しステップと停止条件を両方ともユーザが定義できるようにするのは違い、Python の `for` 文は、任意のシーケンス型 (リストまたは文字列) にわたって反復を行います。反復の順番はシーケンス中に要素が現れる順番です。例えば:

```
>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

コレクションオブジェクトの値を反復処理をしているときに、そのコレクションオブジェクトを変更するコードは理解するのが面倒になり得ます。そうするよりも、コレクションオブジェクトのコピーに対して反復処理をするか、新しいコレクションオブジェクトを作成する方が通常は理解しやすいです:

```
# Create a sample collection
users = {'Hans': 'active', 'Éléonore': 'inactive', '景太郎': 'active'}

# Strategy: Iterate over a copy
for user, status in users.copy().items():
    if status == 'inactive':
        del users[user]

# Strategy: Create a new collection
active_users = {}
for user, status in users.items():
    if status == 'active':
        active_users[user] = status
```

### 4.3. range() 関数

## 3. 「[プログラム実行手順について](#)」の手順に従い、プログラムを実行

Untitled1.ipynb ☆

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

RAM  
ディスク

コメント 共有

コード テキスト

```
[1] 1+1
2
```

```
[2] x = int(input("Please enter an integer: "))

if x < 0:
    x = 0
    print("Negative changed to zero")
elif x == 0:
    print("Zero")
elif x == 1:
    print("Single")
else:
    print("More")

Please enter an integer: 1
Single
```

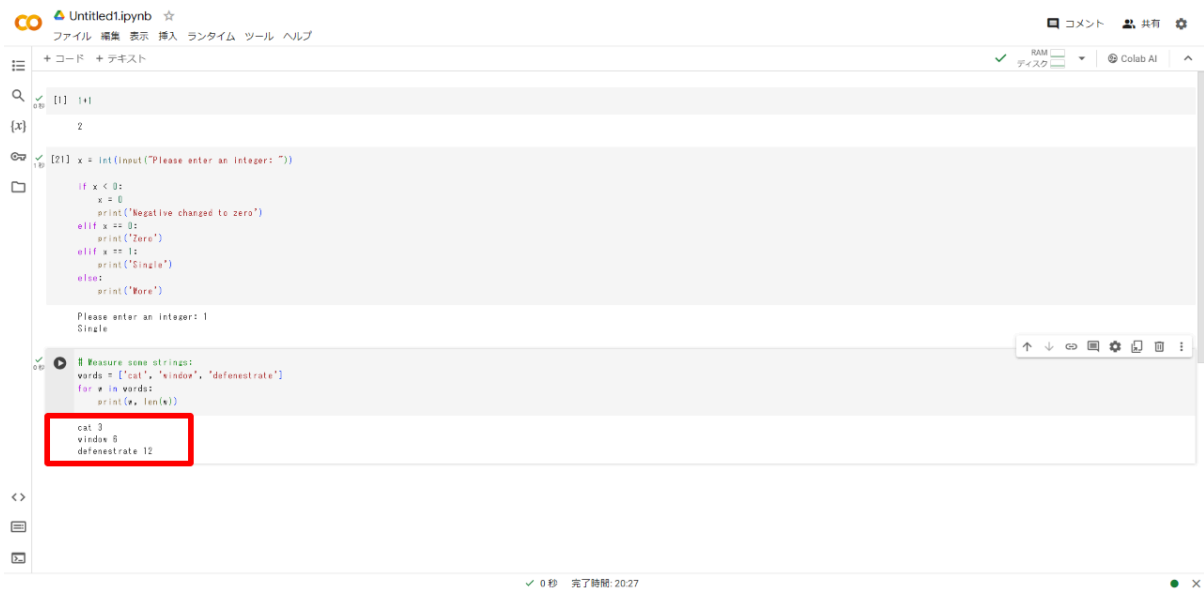
```
# Measure some strings:
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))

cat 3
window 6
defenestrate 12
```

↑ ↓ ↺ ⚙️ 📄 🗑️ ⋮

0 秒 完了時間: 20:27

4. 'cat', 'window', 'defenestrate'の文字と、各文字数が結果として出力されていることを確認



The screenshot shows a Jupyter Notebook with the following code and output:

```
[1] 1+1
```

```
[2] x = int(input("Please enter an integer: "))
```

```
if x < 0:
    x = 0
    print("Negative changed to zero")
elif x == 0:
    print("Zero")
elif x == 1:
    print("Single")
else:
    print("More")
```

Please enter an integer: 1  
Single

```
Measure some strings:
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(len(w))
```

```
cat 3
window 6
defenestrate 12
```

The output for the string lengths is highlighted with a red box.

## 4 他参考ドキュメント

演習内では Pandas や scikit-learn といったPythonライブラリも利用している。  
上記のライブラリについては以下のリンクを参照。

- [Pandas](#) : Pythonのデータ解析ライブラリ
- [scikit-learn](#) : Pythonの機械学習ライブラリ