

Google Colaboratoryを 使ったPythonプログラミング

Google Claboratoryを 使い始める手順

Googleにログイン

Gmail 画像



ログイン



Google 検索

I'm Feeling Lucky

登録したメールなどを入力



ログイン

お客様の Google アカウントを使用

メールアドレスまたは電話番号

██████████@gmail.com



[メールアドレスを忘れた場合](#)

ご自分のパソコンでない場合は、ゲストモードを使用して非公開でログインしてください。 [詳細](#)

[アカウントを作成](#)

次へ

日本語 ▼

[ヘルプ](#)

[プライバシー](#)

[規約](#)

ログインした状態で、 「Google colaboratory」を検索 「Colaboratoryへようこそ」をクリック



Google colaboratory



すべて

ニュース

画像

動画

ショッピング

地図

書籍

もっと見る

ツール



Google

<https://colab.research.google.com> > ...

Colaboratoryへようこそ - Colab - Google

Colab ノートブックは **Google** のクラウド サーバーでコードを実行します。そのため、パソコンの性能にかかわらず、GPU や TPU など **Google** ハードウェアの優れた機能を活用 ...

Python 入門

本チュートリアルでは基本的に Python というプログラミング言語を ...

最適な Colab のプランを選択する

料金がかからないバージョンの Colab では、GPU などの高コスト ...

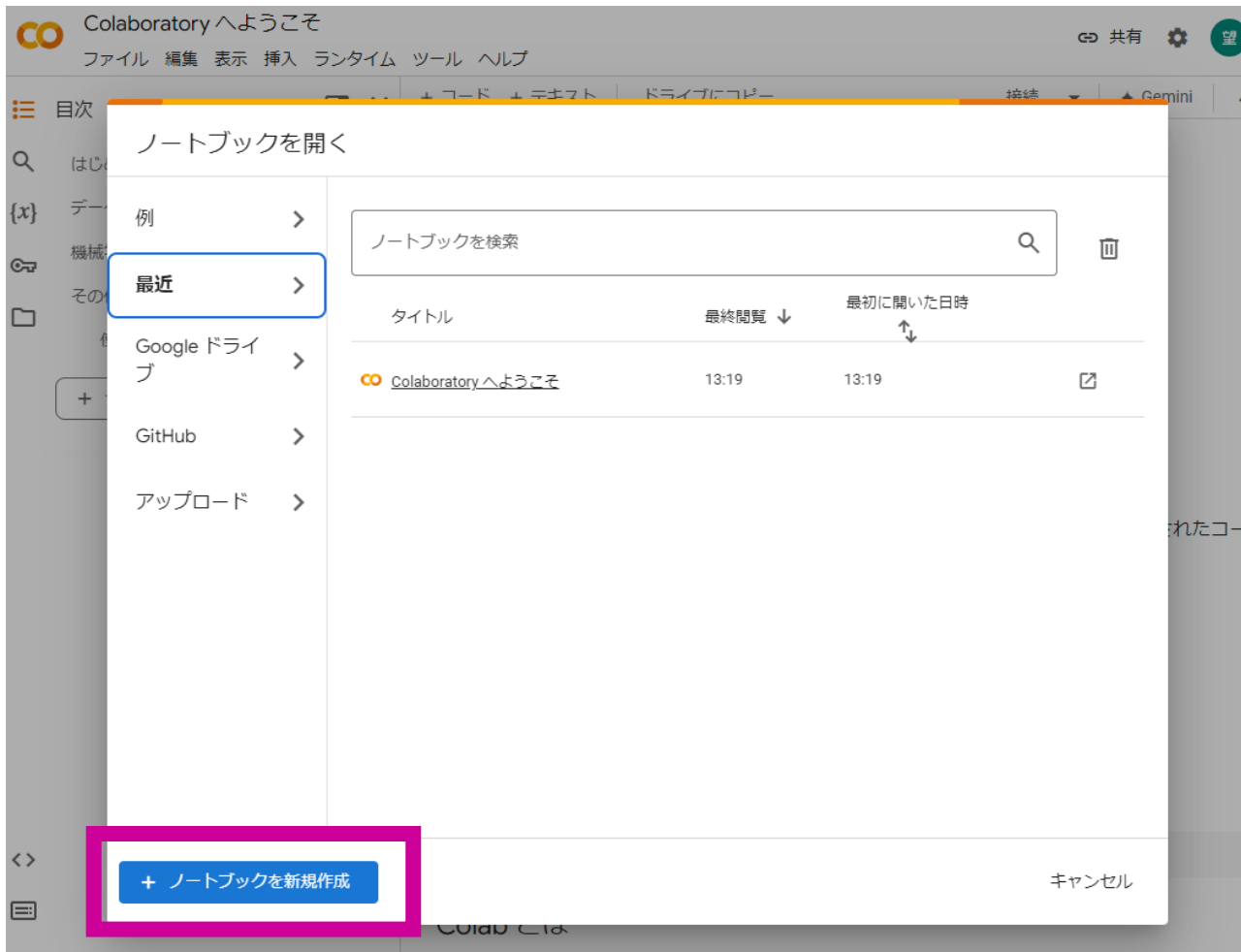
NumPy 入門

これが要素ごとの計算と同じように実行できる理由は、NumPy が自動的 ...

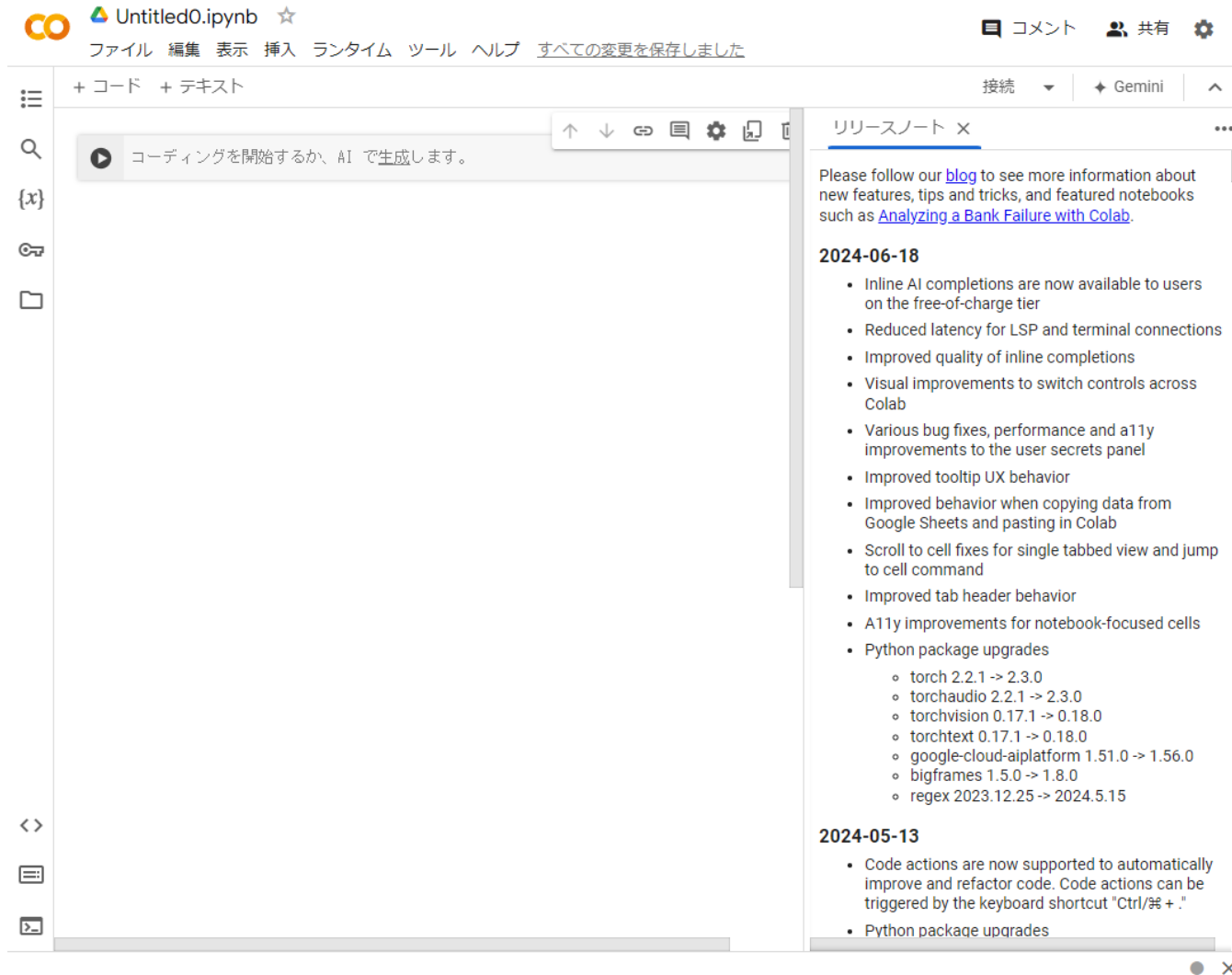
Pandas 入門

CSV ファイルの読み書き. データセットは Google Colaboratory で用意 ...

自分のColabotatoryスペースが表示されます。 「ノートブックを新規作成」をクリック



少し待つと、コード入力画面が表示されます。



※図のように、右側に「リリースノート」が表示されることがあるかも知れません。「×」印を押して閉じてしまってOKです。

左上の「Untitled0.ipynb」が、このノートブックの名前です。
カーソルを合わせると名前を変更できます。

※「.ipynb」という拡張子が表示されている場合があります。名前の変更で拡張子を消してしまっても問題ありません。



ノートブックとは、プロジェクト名みたいなものです

ノートブックの中には、これから作ってゆくプログラムだけでなく、使用するファイルや処理の流れ、得られた結果など、一連の作業内容全体が保存されます。

Googleドライブに戻って、ノートブックがどこに保存されたかを確認してみましょう。左上のCOアイコンをクリックすると、Googleドライブに戻ります。

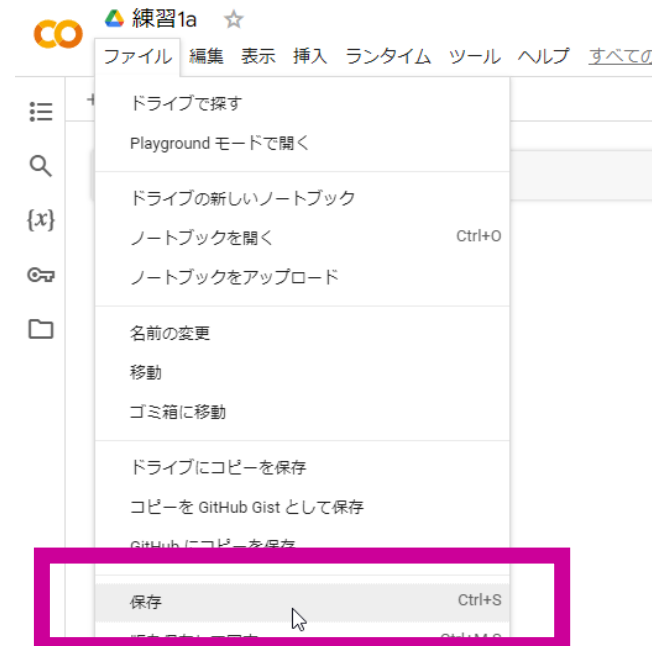


このとき、下記のような警告が出る場合があります。

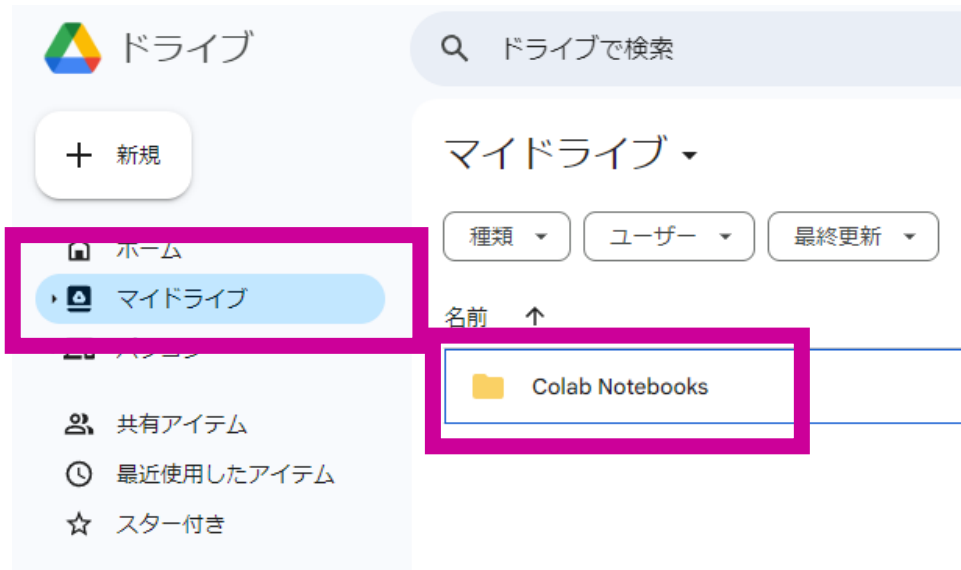


Colaboratoryの編集内容は自動保存されるのが基本ですが、保存しきれていない内容がある場合に、上記の警告が出ます。

警告が出た場合は、ファイルメニューから「保存」を選び、確実に保存しましょう。その後、ドライブに戻ってください。



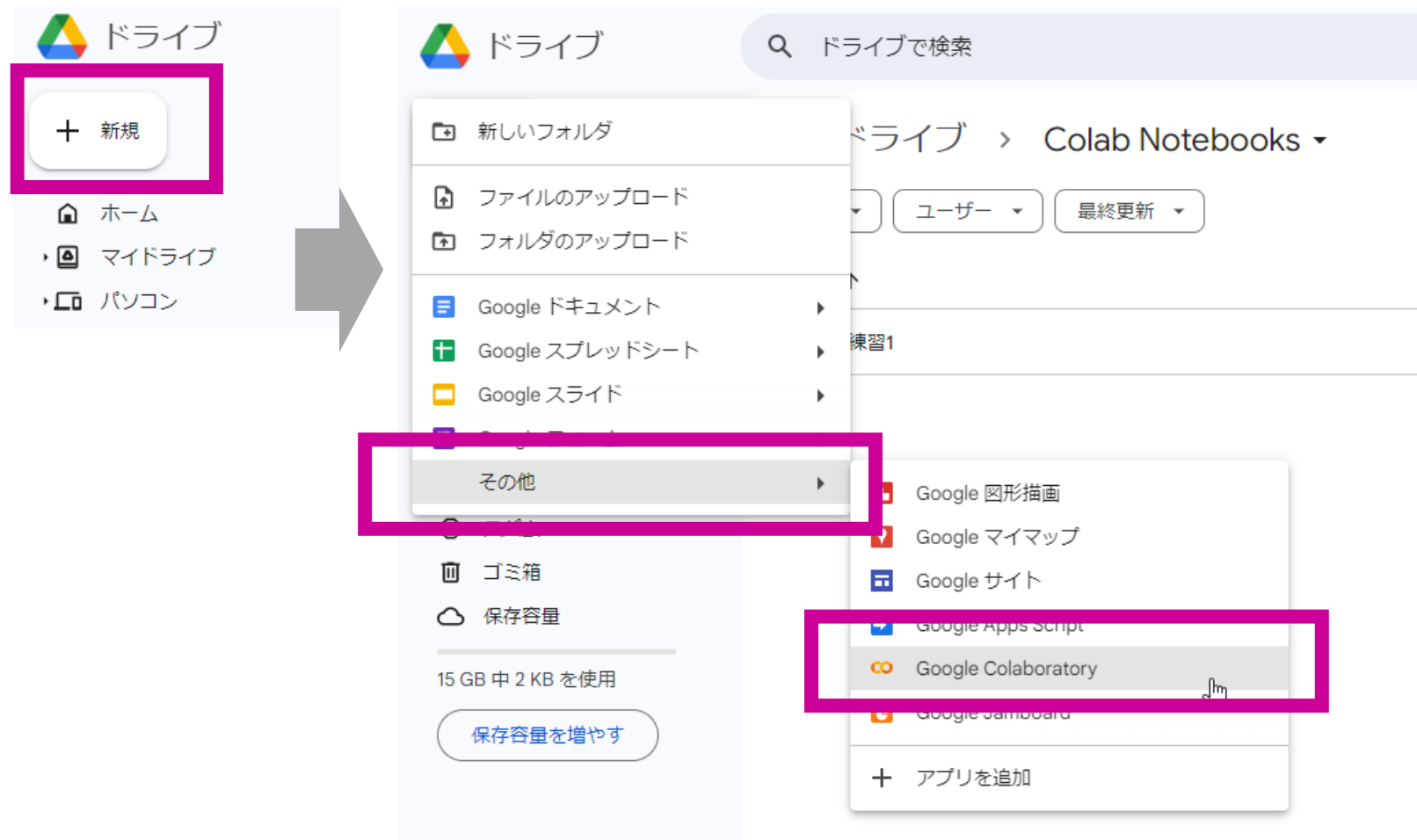
Googleドライブの「マイドライブ」に、「Colab Notebooks」というフォルダができており、その中にノートブックが保存されています。



ノートブックをダブルクリックすると、先の編集画面が開きます。



さらに新しいノートブックを作るには、ドライブの「新規」ボタンから、「その他」、「Google Colaboratory」を選択します。



※「Google Colaboratory」のメニューが見つからない時は、Google Colaboratoryを検索するところからの手順をもう一度行ってください。実は、Colaboratoryを利用するには、ブラウザにアプリを追加する必要がありますが、ここまでの手順を踏むことで、自動的にその追加が完了しています。別のアカウントでログインして、アプリの追加が行われていない場合などは、メニューが出てきません。

Colaboratoryでの プログラミング(1)

実際に動かして、使い方を簡単に確認してみる



この印がついた行に、プログラムを入力します。



練習1 ☆

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ すべて



+ コード + テキスト



{x}



コーディングを開始するか、AI で生成します。



```
print("はじめてのコラボ")
```



0 秒




```
print("はじめてのコラボ")
```

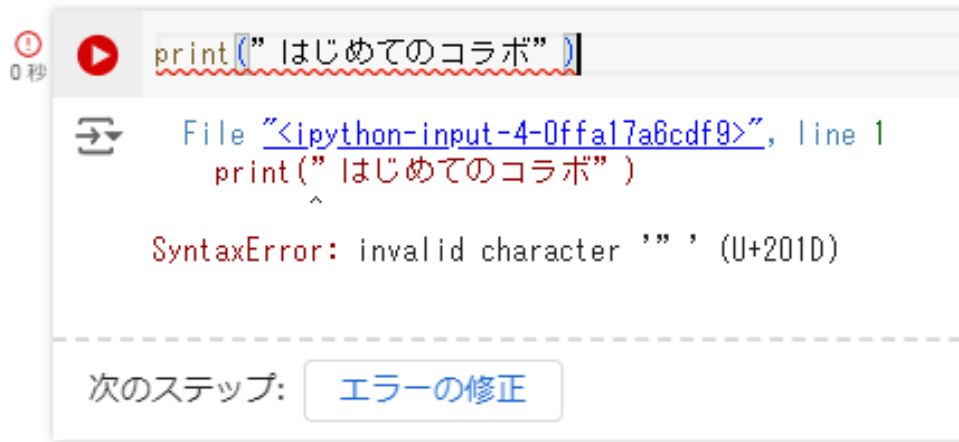


はじめてのコラボ

左の図のように入力してください。
※””で囲まれた部分以外は、全部
半角文字にしなければなりません。

実行ボタン  をクリックすると、
その結果がすぐ下に表示されます。


何かミスがある場合は、警告が出ます。



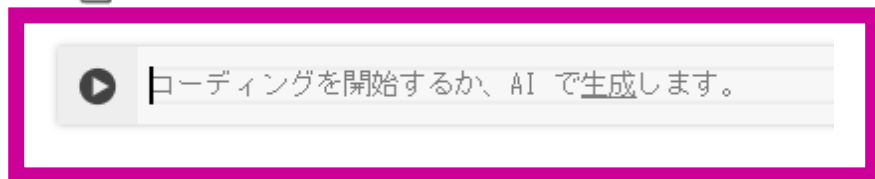
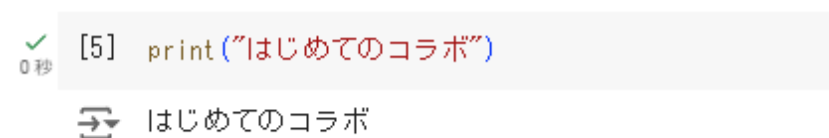
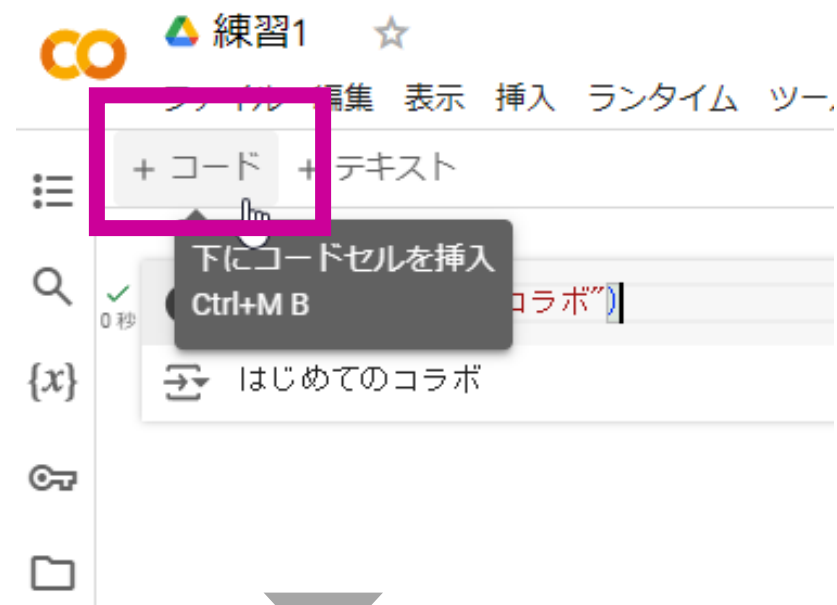
補助機能3
デバッグ

警告が出たら、表示に従って間違いを修正します

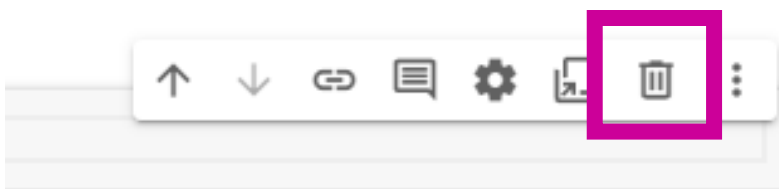
上記の例では、SyntaxError(文法の誤り)で、invalid character(不正な文字)「」が記載されている、と教えてくれています。

半角で書くべき「」を、全角の「」で記載していたので、警告が出ていたことが分かりました。”を半角に修正して、再度実行ボタン  を押して動作を確認してください。

「+コード」をクリックすると、コードを書く場所(「コードセル」と呼びます)が追加されます。




※不要なセルは、セルの右にあるごみ箱アイコンを押すと削除されます。





簡単なコードを書いてみる

数字を計算するコードを書いて、動作を確かめてみましょう。
コードセルに、以下のように書きます。




```
a = 2
b = 5
c = a + b
print(c)
```

実行ボタン  を押して実行すると、
下記のように、計算系結果が表示されます。



```
a = 2
b = 5
c = a + b
print(c)
```

 7

【解説】

コア機能2 変数

a、b、cは、値を覚えておく「変数」です。

「=」は、右辺の値を左辺の変数に代入する
「代入演算子」です。

コア機能3 演算子

「+」は、数値同士の場合は加算する「算術演算子」です。
(文字列に対して使うと、文字列同士を連結します)

コア機能1 入出力

printは、画面に値を表示する命令です。変数を与えると、変数が持つ値を表示します。

【チャレンジ】

aとbで他の四則演算を行い、結果を確認してみましょう。演算子はそれぞれ以下です。

足し算: + 掛け算: *
引き算: - 割り算: /

Colaboratoryでの プログラミング(2)

プログラミングの例として、データの平均値を計算し、結果をファイルとして得るまでの基本的な流れを確認します。

ここまでできれば、今後、入力データとプログラムを様々なアレンジするだけで、いろいろな処理を自在に行えるようになります。

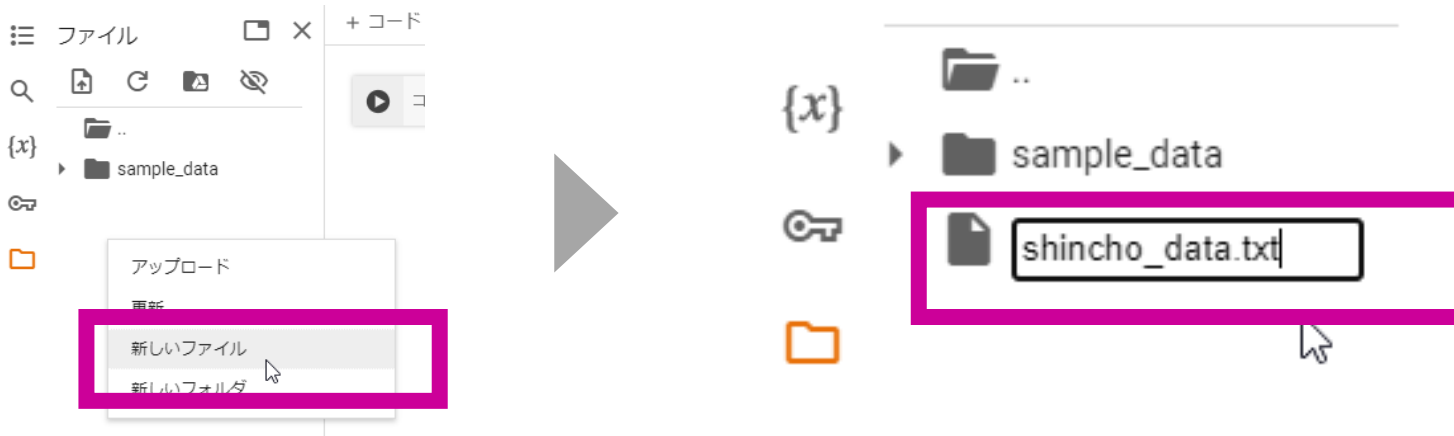
処理するデータの準備

ノートブックの中に、処理するデータをファイルとして準備します

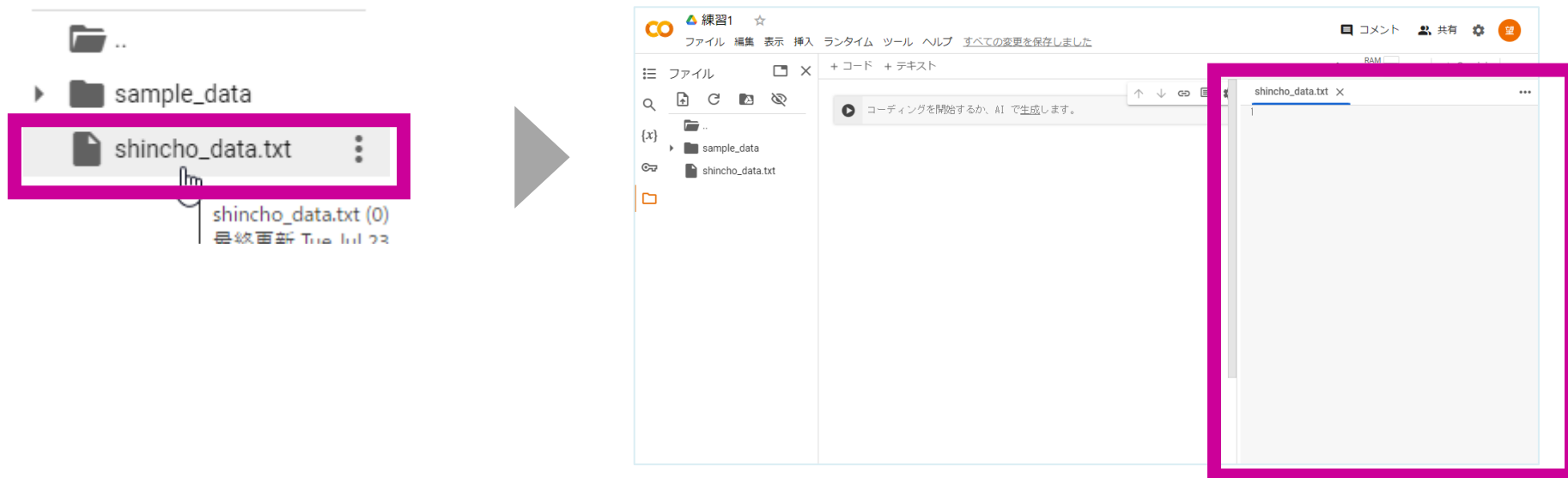
左のサイドバーから、フォルダアイコンをクリックして情報を展開させます。



展開された部分で右クリックし、「新しいファイル」を選択します。新しいファイルアイコンが表示されるので、適当な名前(shincho_data.txt)を付けます。



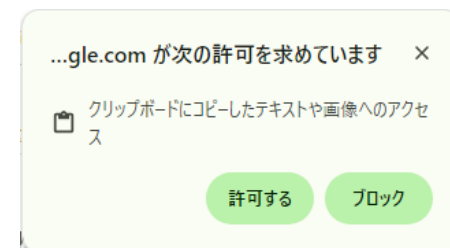
できたファイル(shincho_data.txt)のアイコンをダブルクリックすると、画面右側にファイル編集画面が表示されます。



Excelなどから数値データをコピーして、編集画面上で右クリックし、Pasteを選択します。(Ctrlキー + Vの貼り付けショートカットは効かないようです)




※以下のような警告が出る場合は、「許可する」を押してください。



ファイルからデータを読み込む処理

作ったファイルshincho_data.txtからデータを読み込んで、処理して
いく流れを作ります

コードセルに、次のコードを書きます。

```
 data = open("shincho_data.txt", "r")  
  
for line in data:  
    print(line)  
  
data.close()
```

【解説】

openはファイルを開く命令(関数)で、“r”は読み込み(read)モードで開くことを意味します。

dataと名付けたものは、開いたファイルのことをプログラムで扱うための変数のようなものです。

専門的には、ファイルハンドラと呼ばれます。

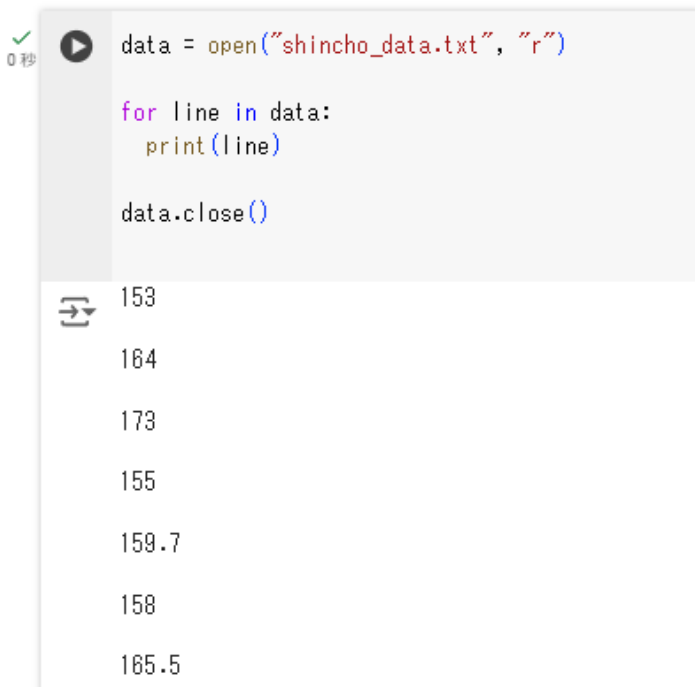
開いたファイルを示すdataは、最後にcloseで閉じて終了。

openでファイルを開いて、dataという名で扱えるようにし、for～で内容を一行ずつline変数に読み込み、それをprintで画面上に表示するプログラムです。

forで繰り返される内容(printの行)は、字下げ(インデント)して書きます。
pythonは、この字下げに大きな意味があり、同じ字下げ幅が続く行を、一つのブロックとしてとらえます(ここでは、forによる繰り返しが行われる行を示す)。**字下げしないとエラー**になります。

実行ボタンを押して、書いたコードを実行してみます

下記のように、ファイルの中身が無事に表示されました。



```
data = open("shincho_data.txt", "r")

for line in data:
    print(line)

data.close()
```

153
164
173
155
159.7
158
165.5

※データによって図の数値とは異なる場合があります。

しかし、数値が1行おきに出力されているようです。
次は、ここを修正してみましょう。

一行おきに出力されているのを直す

以下のようにコードを修正します。

```
data = open("shincho_data.txt", "r")

for line in data:
    # 以下の二行を修正
    real_line = line.strip()
    print(real_line)

data.close()
```

【解説】

に続くその行の文字は、コードではなくコメントとみなされ、何も処理されません。
書かなくてもプログラム上は問題ありません。

lineに読み込まれた行のデータについて、strip命令により、行の末尾についた改行文字を除去しています。

実はprint命令は、与えた文字に勝手に改行文字をつけて出力します。このため前回までは、もともとついていた改行と重複して出力されていたため、1行おきになっていたのです。

strip命令で、行の末尾についていた改行文字を取り払い、real_line変数に入れました。

実行すると、期待通りに、1行ずつデータが表示されるようになりました。

```
→ 152
   167
   160.5
   153
   164
   173
   155
   150.7
```

コア機能と補助機能の確認

プログラムの基本機能に分けてとらえてみます。

コア機能2
変数

補助機能2
コメント

コア機能1
入出力
(画面へ出力)
(コア機能3
命令でもある)

```
data = open("shincho_data.txt", "r")
```

```
for line in data:
```

```
# 以下の二行を修正
```

```
real_line = line.strip()
```

```
print(real_line)
```

```
data.close()
```

コア機能1
入出力

(ファイルから入力)
(補助機能3
関数でもある)

コア機能5
制御構造

(繰り返し)

コア機能3
命令

(参考)データファイルのアップロード

ここではノートブック上でファイルを作成する方法を紹介しましたが、自分のPC上に、すでにデータファイルを作成済みである場合は、そのファイルをアップロードしてそのまま使用することも可能です。

アップロードするには、PC上のファイルを、ノートブックのファイル画面以上にドラッグ&ドロップします。



あるいは、アップロードボタンからファイルを選択します。

計算の処理を作ってゆく1: 合計を出す

ファイルからデータを読み込む準備ができたので、数値の処理を書いてゆきます。まずは目標として、読み込んだ数値の合計を出してみましょう。

合計を出すには、読み込んだ数値を全部足し合わせればできそうです。合計値を覚えておく変数を準備すればよさそうですね。

以下のようにコードを書きます。

【解説】

合計を覚えておく変数totalを準備し、初期値0を覚えさせておきます。

改行文字を取り去ったreal_lineは、実はデータを「文字」として保持しています。数値として計算できるようにするため、float関数で文字を数値に直し、value変数に保持し直しています。

最後に、totalに足しこみます。それまでのtotalに、新しいvalueを足したものを、新しいtotalに代入することで、足しこみを実現しています。

最後に、足した結果として、totalの値を表示しています。

※コメントは、この例ように行の途中から書くこともできます。

```
data = open("shincho_data.txt", "r")
```

```
total = 0      # 追加
```

```
for line in data:
```

```
    real_line = line.strip()
```

```
    print(real_line)
```

```
    value = float(real_line) # 追加
```

```
    total = total + value    # 追加
```

```
data.close()
```

```
print(total)    # 追加
```

155.0

155

181

172

5192.2



実行すると、末尾に合計が出ました。Excelでの計算と合っていましたか？

エラーが起きるかも？

float関数は、「数値に見える文字列を数値に変換する」処理をします。このため、元のファイルの中に数値に置き換えられない文字が含まれていると、エラーとなります。

たとえば、余分な空行や全角で書かれた数字などがある場合にエラーになります。

```
shincho_data.txt X
1 152
2 167
3 160.5
4
5 153
6 164
7 173
8 155
9 159.7
10 158
11 165.5
12 176
```

←たとえば空行
があると、

```
data = open("shincho_data.txt", "r")

total = 0      # 追加
for line in data:
    real_line = line.strip()
    print(real_line)
    value = float(real_line) # 追加
    total = total + value    # 追加

data.close()

print(total)    # 追加
```

```
152
167
160.5

-----
ValueError                                Traceback
<ipython-input-30-68af67a49c66> in <cell line: 4>()
      5 real_line = line.strip()
      6 print(real_line)
----> 7 value = float(real_line) # 追加
      8 total = total + value    # 追加
      9
```

ValueError: could not convert string to float: ''

補助機能3 デバッグ

- それまでに正常に処理された結果
- エラーを起こしたコードの場所
- エラーの内容

などから、何が原因かを考えることができます。親切！

コア機能と補助機能の確認

コア機能2
変数

```
data = open("shincho_data.txt", "r")

total = 0      # 追加
for line in data:
    real_line = line.strip()
    print(real_line)
    value = float(real_line) # 追加
    total = total + value    # 追加

data.close()

print(total)   # 追加
```

補助機能1
関数

(float関数は、あらかじめ用意されている関数)

コア機能3
演算子

(参考)コードをすっきりと書く

value変数は、forループの中で一時的に使われ、しかも作られた後に1度しか参照されないの
で、いちいち別の変数に保持し直す意味合いは、あまりありません。

処理結果を、変数に保持させずにfloat関数に渡すことで、コードをすっきり書くことができます。

これまでの書き方

意味合いを理解する上ではわかりやすい

```
data = open("shincho_data.txt", "r")

total = 0      # 追加
for line in data:
    real_line = line.strip()
    print(real_line)
    value = float(real_line) # 追加
    total = total + value    # 追加

data.close()

print(total)    # 追加
```

すっきりした書き方

慣れてくるとこれでも十分理解できる。

```
data = open("shincho_data.txt", "r")

total = 0
for line in data:
    real_line = line.strip()
    print(real_line)
    total += float(real_line) # 修正

data.close()

print(total)
```

totalをtotal自身に足しこむ処理は、「+=」という特別な演算子ですっきり書くことができます。

計算の処理を作ってゆく2: 平均を出す

次の目標として、平均値を計算してみましょう

平均値を出すには、これまでで計算した合計値を、データの個数で割ればよいです。データの個数を数えるには、どうしたらよいのでしょうか？

forループが回る回数を数えればよさそうですね。以下のようにコードを修正します。

```
data = open("shincho_data.txt", "r")

total = 0
num = 0      # 追加

for line in data:
    real_line = line.strip()
    print(real_line)
    value = float(real_line)
    total = total + value
    num = num + 1      # 追加

data.close()

print(total)
print(num)            # 追加
```



実行すると、末尾に合計値と個数が出ました。
個数は入力データと一致していましたか？

【解説】

個数を覚えておく変数numを準備し、初期値0を覚えさせておきます。

forで一行ずつ処理されるたびに、numを1ずつ増やしています。

最後に結果(個数であるnumの値)を出力しています。

【注意】

この例では、空行があると、float関数でエラーになるため気付けますが、floatの計算をせずにnumだけ数える処理をした場合は、空行や、数字以外のデータがあっても、numは行数分だけ加算されてしまい、予期しない結果となってしまいます。

しっかりしたプログラムを作るには、こうしたありがちなミスへの対処も考える必要があります。

150
161
172
5192.2
32

平均値の計算

合計と個数が出たので、割り算をすればよいですね。
コードを以下のように修正します。

```
▶ data = open("shincho_data.txt", "r")

total = 0
num = 0

for line in data:
    real_line = line.strip()
    print(real_line)
    value = float(real_line)
    total = total + value
    num = num + 1

data.close()

print(total)
print(num)

average = total / num    # 追加
print(average)          # 追加
```

【解説】

割り算をする演算子は「/」(スラッシュ)記号です。
totalをnumで割り、average変数に格納しています。



実行すると、末尾に平均値が出ました。
Excelでの計算と合っていましたか？

```
100
161
172
5192.2
32
162.25625
```

(参考)ノートブックの利点

合計totalと個数numは、先ほどまでで計算済みです。平均値を出すための修正では、この二つの変数を使って、最後の2行の処理を追加しただけでした。

ノートブックは、ノートブック内で一度計算された変数の値を、ずっと保持し続けています。このため、最後の2行の処理は、コードセルを新たに追加して、以下のように書くだけでも計算できます。

```
average = total / num
print(average)
```

162.25625

ファイルからの読み込みやfor~による繰り返し処理がないので、すぐに結果が得られます(体感的にはあまりわからないレベルですが)。

複雑な計算処理も、ステップバイステップで、結果を確認しながら、目的とする処理を達成してゆけるのが、ノートブックの良いところです。

もちろん、全体のステップを一気に通して実行させることも可能です。

※ブラウザを閉じた後は、変数の値や計算結果は消えてしまいますので、同じ結果を得るには再度実行する必要があります。

現在のノートブックで、どんな変数がどんな値を持っているかは、左のサイドバーの「{x}」アイコンをクリックすると表示されます。

変数名にカーソルを合わせると、持っている値が表示されるほか、各変数の「型」もわかります。

変数

変数インスペクタは、開いている間、ランタイムのパフォーマンスに影響を与える可能性があります。

名前	型	形状
average	float	
data	TextIOWrapper	
line	str	4 chars
num	int	
real_line	str	3 chars
total	float	
value	float	

型:float
値:172.0
172.0

結果をファイルとして得る

Colaboratoryは、いったんブラウザを閉じてしまうと、実行した計算結果が消えてしまいます。もう一度結果を得るには、プログラムを再度実行しなければなりません。

計算結果をファイルに出力することで、実施した成果を確実に残すことができます。データをファイルに書き出すには、以下のようにします。

※ここでは、すでにaverageが計算済みであることを前提としています。一度ブラウザを閉じてしまった場合は、これらを計算するセルをもう一度実行してください。

```
out = open("out_file.txt", "w")

out.write(str(average))

out.close()
```



実行すると、"out_file.txt"というファイルがColaboratoryの中にでき、ダブルクリックで中身を確認すると、平均値が書きだされているのが分かります。

【解説】

open命令で"w"を指定すると、書き出し(write)モードでファイルを開き、「out」という名の変数(ハンドラ)で扱えるようになります。

write命令で、averageを書きだします。このとき、write命令は文字列データしか受け付けないため、str関数で、数値データを文字列に変換しています(以前のfloat関数と逆)。

close命令で、ファイルを閉じます。"out_file.txt"が存在しない場合は新規に作成され、存在する場合は上書きされます。

【発展】

write命令は、書きだした後、勝手に改行をしてくれません。改行するには、改行文字(¥n)を明示的に付け足してあげる必要があります。

文字列データを連結するには、「+」演算子を使います。例えば下記のようにすると、分かりやすい結果を得ることができます。

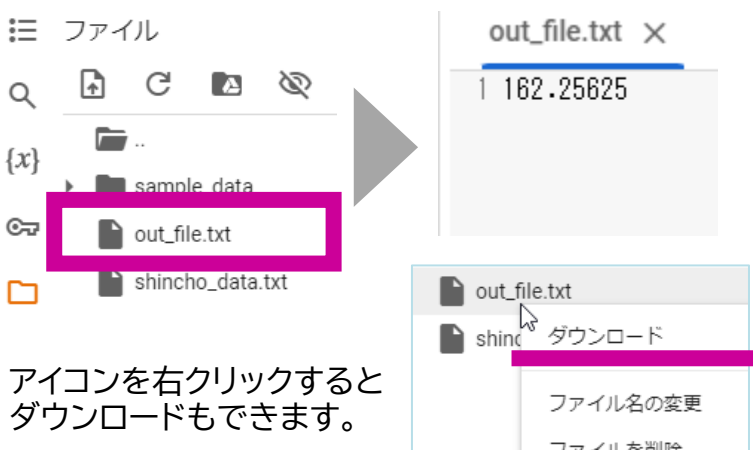
```
out = open("out_file.txt", "w")

out.write("合計: " + str(total) + "¥n")
out.write("個数: " + str(num) + "¥n")
out.write("平均: " + str(average) + "¥n")

out.close()
```

out_file.txt X

```
1 合計: 5192.2
2 個数: 32
3 平均: 162.25625
4
```



アイコンを右クリックするとダウンロードもできます。

【発展】

Colaboratoryの 生成AIで遊ぶ

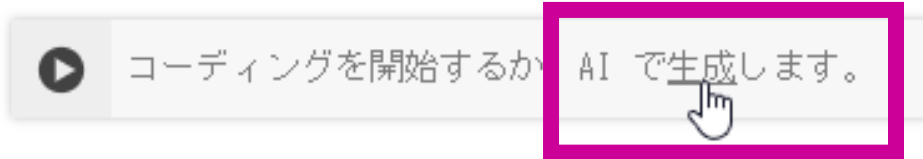


コーディングを開始するか、AI で生成します。

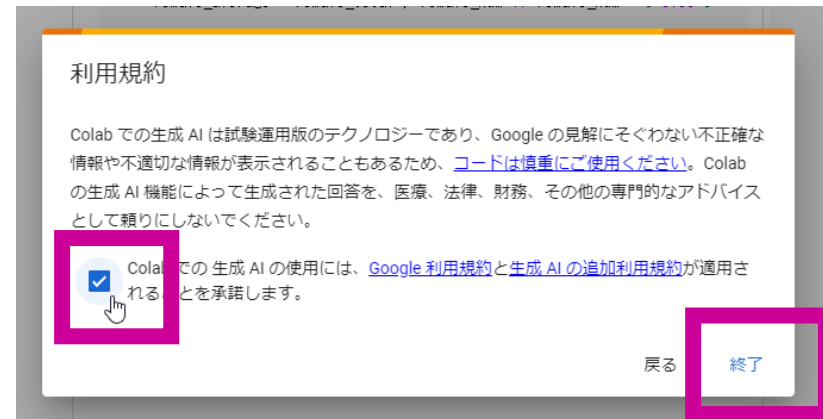
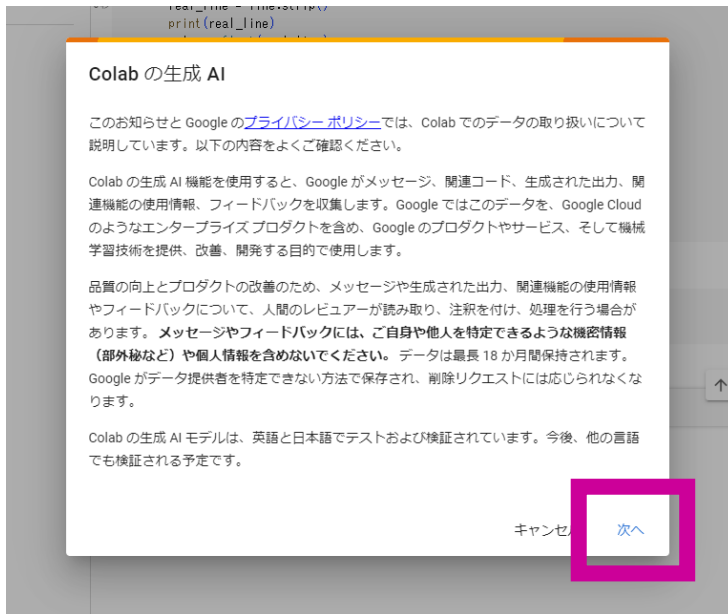
先人たちの英知に学べるすごい技術です

生成AIを使う流れ

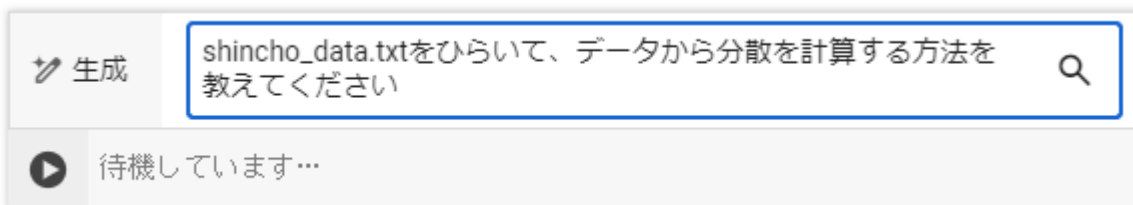
「生成」をクリックし、



利用規約を承諾します。




プロンプトに指示を書き、リターンキーを押すと、見本となるコードが生成されます。



例1： 分散を計算するプログラム

🔗 生成 🔍

< 1 of 1 > 👍 💬 [注意してコードをご使用ください](#)

 # prompt: shincho_data.txtのデータから、数値の分散を計算するには？

data = open("shincho_data.txt", "r")

total = 0
num = 0
sum_squares = 0 # 平方の合計を計算するための変数

for line in data:
 real_line = line.strip()
 value = float(real_line)
 total += value
 num += 1
 sum_squares += value ** 2 # 各値の平方を計算して加算

data.close()

average = total / num
variance = sum_squares / num - average ** 2 # 分散の計算

print("平均値:", average)
print("分散:", variance)

🔄 平均値: 162.25625
分散: 52.09183593749913

自分がそれまでに書いたプログラムも参考にしながら、目的のプログラムの例を示してくれます。

【気付き】

分散を計算するには、一度平均値を出した後、各要素と平均値の差を計算してゆくの、forループは2度回るはずだが、1度しか回っていない。

実行結果はExcelと合っているが、本当にこの簡単な方法で正しく計算できているのか？

分散の計算方法の確認

$(a + b)^2 = a^2 + 2ab + b^2$ なので、展開すると、

$$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \rightarrow \frac{\{ (x_1 - \text{ave})^2 + (x_2 - \text{ave})^2 + (x_3 - \text{ave})^2 + \vdots + (x_n - \text{ave})^2 \}}{n} \rightarrow \frac{\{ x_1^2 - 2 * x_1 * \text{ave} + \text{ave}^2 + x_2^2 - 2 * x_2 * \text{ave} + \text{ave}^2 + x_3^2 - 2 * x_3 * \text{ave} + \text{ave}^2 + \vdots + x_n^2 - 2 * x_n * \text{ave} + \text{ave}^2 \}}{n}$$

縦方向をまとめると、

$$\rightarrow \frac{\{ (x_1^2 + x_2^2 + x_3^2 \cdots + x_n^2) - 2 * (x_1 + x_2 + x_3 \cdots + x_n) * \text{ave} + n * \text{ave}^2 \}}{n} = n * \text{ave}$$

$$\rightarrow \frac{\{ (x_1^2 + x_2^2 + x_3^2 \cdots + x_n^2) - 2 * n * \text{ave} * \text{ave} + n * \text{ave}^2 \}}{n} \rightarrow \frac{\{ (x_1^2 + x_2^2 + x_3^2 \cdots + x_n^2) - 2 * n * \text{ave}^2 + n * \text{ave}^2 \}}{n}$$

合ってた！

$$\rightarrow \frac{\{ (x_1^2 + x_2^2 + x_3^2 \cdots + x_n^2) - n * \text{ave}^2 \}}{n} \rightarrow \boxed{(x_1^2 + x_2^2 + x_3^2 \cdots + x_n^2)/n - \text{ave}^2}$$

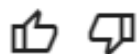
例2: ヒストグラムを描く

生成

shincho_data.txtのデータのヒストグラムを描いてください



< 1 of 1 >



[注意してコードをご使用ください](#)

prompt: shincho_data.txtのデータのヒストグラムを描いてください

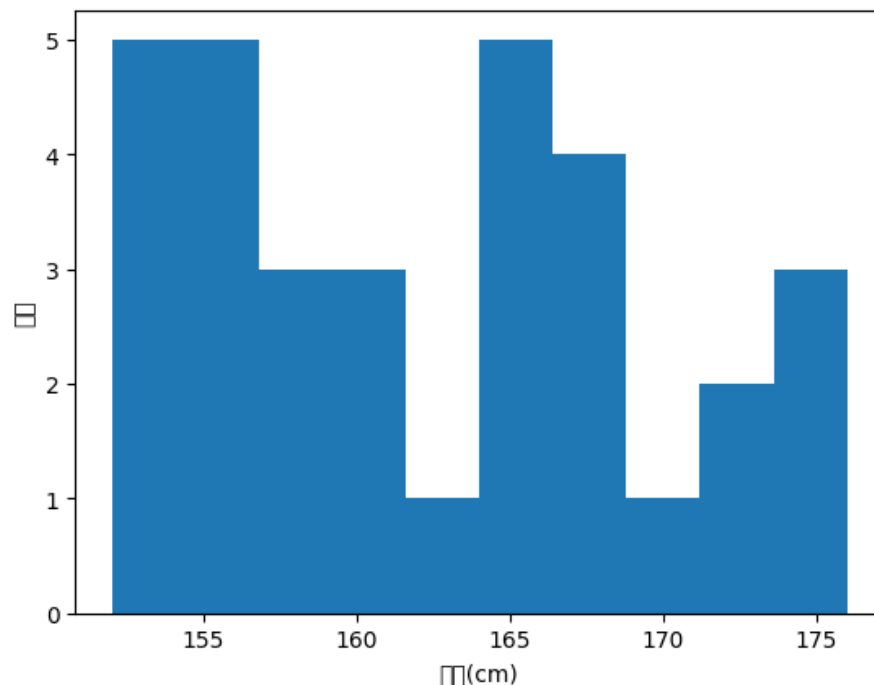
```
import matplotlib.pyplot as plt

data = open("shincho_data.txt", "r")

values = []
for line in data:
    real_line = line.strip()
    value = float(real_line)
    values.append(value)

data.close()

plt.hist(values)
plt.xlabel("身長(cm)")
plt.ylabel("人数")
plt.show()
```



使うライブラリや、その使い方が分かる。
日本語はうまく表示されないので、若干
修正が必要。

例3: タブ区切りのテキストを処理する

生成

shincho_data2.txtのデータはタブ区切りデータで、1列目に男性と女性の区別、2列目に身長データが記載されています。男性女性それぞれのデータの平均を計算してください。

閉

< 1 of 1 > [注意してコードをご使用ください](#)

prompt: shincho_data2.txtのデータはタブ区切りデータで、1列目に男性と女性の区別

```
male_total = 0
male_num = 0
female_total = 0
female_num = 0

with open("shincho_data2.txt", "r") as data:
    for line in data:
        gender, value_str = line.strip().split("\t")
        value = float(value_str)
        if gender == "male":
            male_total += value
            male_num += 1
        elif gender == "female":
            female_total += value
            female_num += 1

male_average = male_total / male_num if male_num > 0 else 0 # ゼロ除算回避
female_average = female_total / female_num if female_num > 0 else 0

print("男性の平均身長:", male_average)
print("女性の平均身長:", female_average)
```

「こんな感じに書けば処理できるのか…」などがわかり、実際にプログラムを作るときの大きなヒントになります。

→ 豊富な経験はもはや不要

入力データや結果は省略

【発展】 今後のために

そのほかの
Pythonプログラミング

Google Colaboratoryでも、かなり実用的な処理を行うことができますが、セルごとにコードを実行したり、実行の順番が結果に影響したりなど、少し使い方にくせがあります。

また、オンラインでないと使えないことや、処理するデータを一度アップロードしなければならないことなど、若干の使いづらさもあります。

Pythonは、自分のPCにインストールして使うこともできます(これまではそれが主流でした)。そして、プログラムを書くための専用ソフトも存在します。

ここでは、自分のPCでPythonプログラムを行ういくつかの方法を、簡単に紹介します。

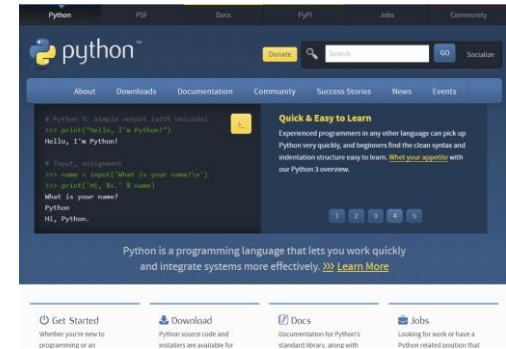
自分のPCにPythonをインストールする

とりあえずは

方法1: 本家Pythonのサイトからインストーラをダウンロードする

<https://www.python.org/>

Downloadsのメニューから、自分のPCに合ったインストーラをダウンロードして使います。



慣れてきたら or 最初からチャレンジしたい

方法2: Minicondaをインストールする

実際のやり方は、「miniconda インストール windows」などで検索してみてください。

Pythonを使ってゆくと、便利なプラグイン(ライブラリー)を、追加でたくさんインストールしてゆくことになります。中には、特定のバージョンのPythonでしか動作しないライブラリーなどもあります。Minicondaは、異なる複数のPython環境(インストールしたライブラリーを含む)を、切り替えて使うことができるシステムです。なので、ワンチャンしかない方法1に比べて、より柔軟にpythonを使いこなすことができます。また、ライブラリーのインストールも簡単にできるので、バイオ系では実践現場で多用されています。

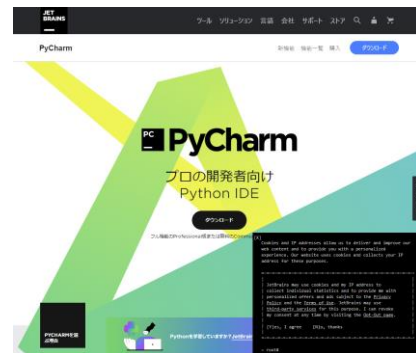
Colaboratoryの元となっているjupyter notebookも、比較的簡単にインストールできるので、自分のPCだけでColaboratory的なプログラミング環境も構築できます。

プログラミング環境

Pythonプログラミングを実践で使用するゆく場合は、高度なコード管理機能などがついた「開発環境」ソフトを使うと、効率よい開発が行えます。

1: PyCharm <https://www.jetbrains.com/ja-jp/pycharm/>

プロ向けの機能までたくさんついているため、いきなり使いこなすのは難しいですが、関係するコード間を行き来したり、変数名を一気に変えたり、エラーを事前に教えてくれたり、コードを補完してくれたりなど、実践的な開発には欠かせない機能が揃っています。動作が軽く安定しており、おすすめ。



2: Eclipse (Pleiades) <https://willbrains.jp/>

豊富なプラグインを組み合わせることで、Pythonだけでなく、Java、PHP、Rubyなど、様々な言語に対応できます。あらかじめ各種言語のプラグインを組み込み、日本語化されたパッケージ、Pleiades All in Oneが、上記リンクで配布されています。コードの構造を一覧する「アウトライン機能」が使いやすく、個人的には一番のお気に入り。



3: VSCode <https://code.visualstudio.com/>

様々なプログラミング言語に対応した、コードを書くためのエディタです。コードを書くだけなら、上記二つよりも学習コストが低く、気軽に使えると思います。Microsoft製。広く人気の高いソフトです。

