コンピュータリテラシ発展 ～Pythonを学ぶ～

第２回：Pythonを始めよう

(shimizu@info.shonan-it.ac.jp)

今回の授業内容

今回の授業内容

- Python
- Google Colaboratory
- Python
-
-
-

Pythonの特徴

Pythonの特徴

Python

1.シ　プル で 変 みやすい　　Python

2.　　Python　　AI

3.　　なライブラ　　Python

4.イ　タ　プ　タ　　Python

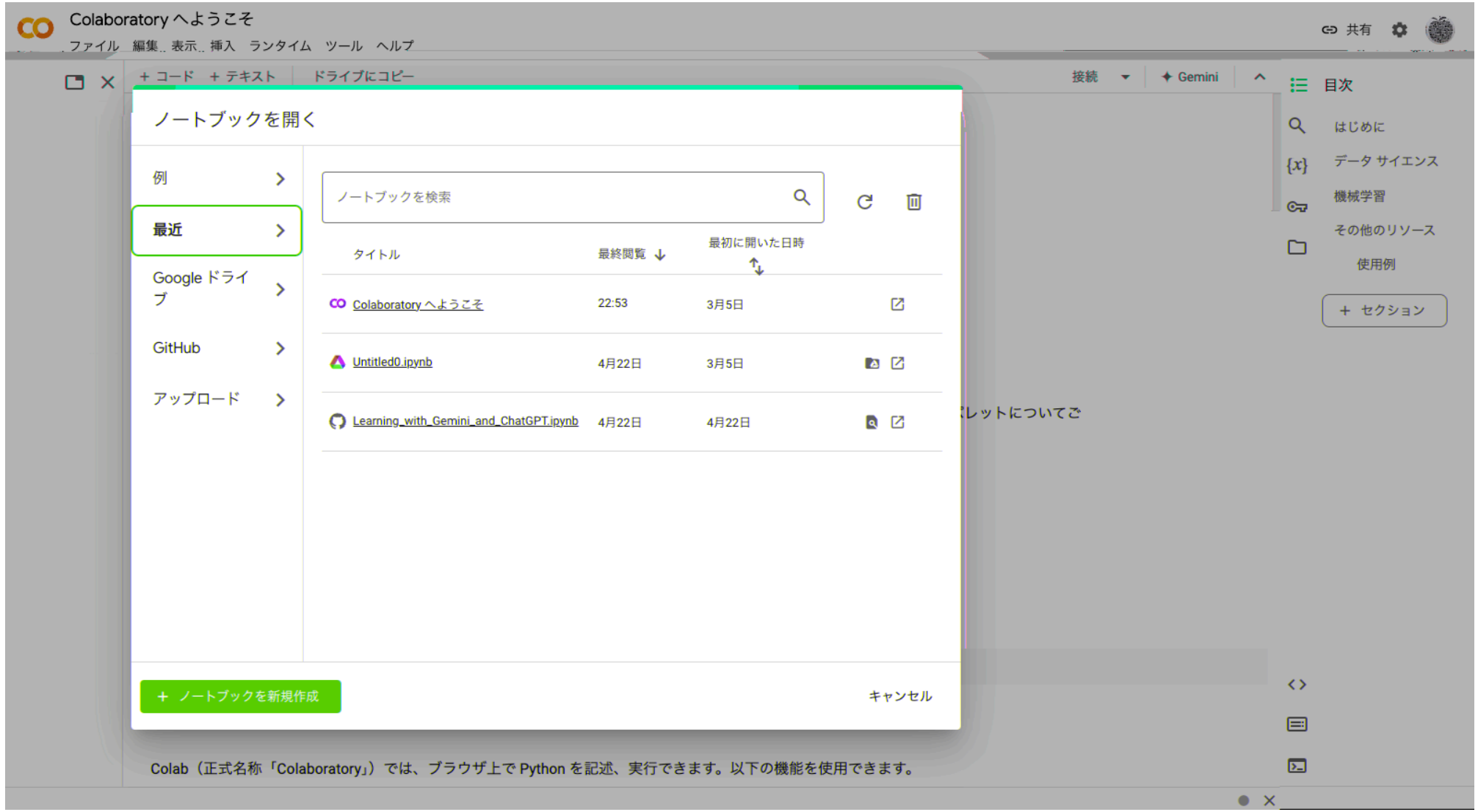
oogle Colaboratory の使い方

oogle Colaboratory の使い方

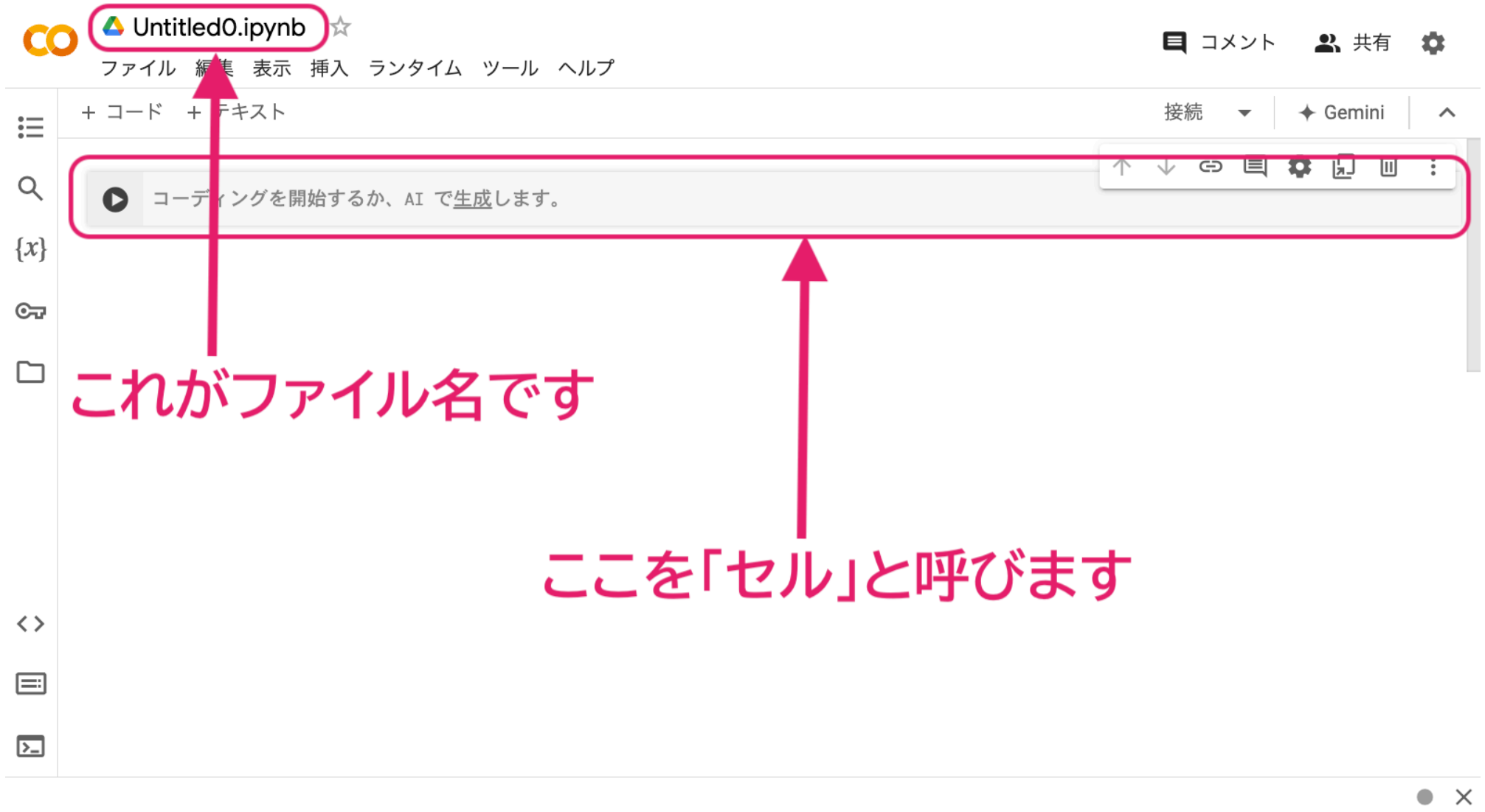
- Python　　Google Colaboratory
- Google Colab　　Jupyter　otebook

- 　　Google　　@sit.shonan-it.ac.jp
　　Google Drive

ノートブックの作成

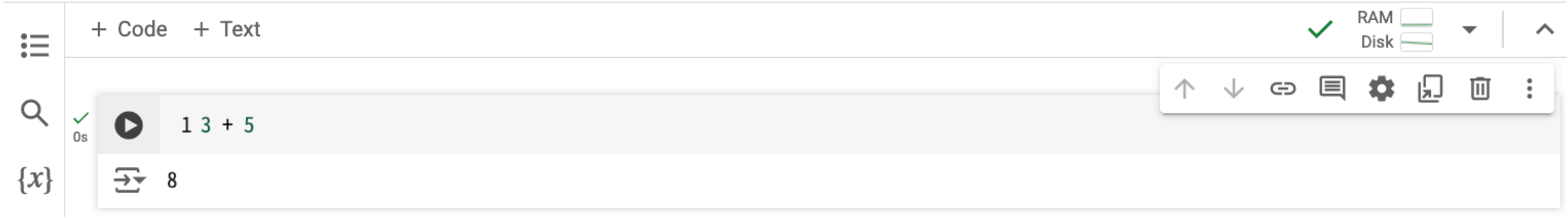

Colaboratory へようこそ
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ
共有
+ コード + テキスト ドライブにコピー
接続 Gemini
目次
はじめに
データ サイエンス
機械学習
その他のリソース
使用例
+ セクション
ノートブックを開く
例
最近
Google ドライブ
GitHub
アップロード
ノートブックを検索
タイトル
最終閲覧
最初に開いた日時
Colaboratory へようこそ 22:53 3月5日
Untitled0.ipynb 4月22日 3月5日
Learning_with_Gemini_and_ChatGPT.ipynb 4月22日 4月22日
+ ノートブックを新規作成
キャンセル
Colab（正式名称「Colaboratory」）では、ブラウザ上で Python を記述、実行できます。以下の機能を使用できます。

ノートブックの作成


Untitled0.ipynb
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ
コメント
共有
+ コード + テキスト
接続
Gemini
コーディングを開始するか、AI で生成します。
これがファイル名です
ここを「セル」と呼びます

Colabをつかってみる

1. `3 + 5`
2.
3. OK

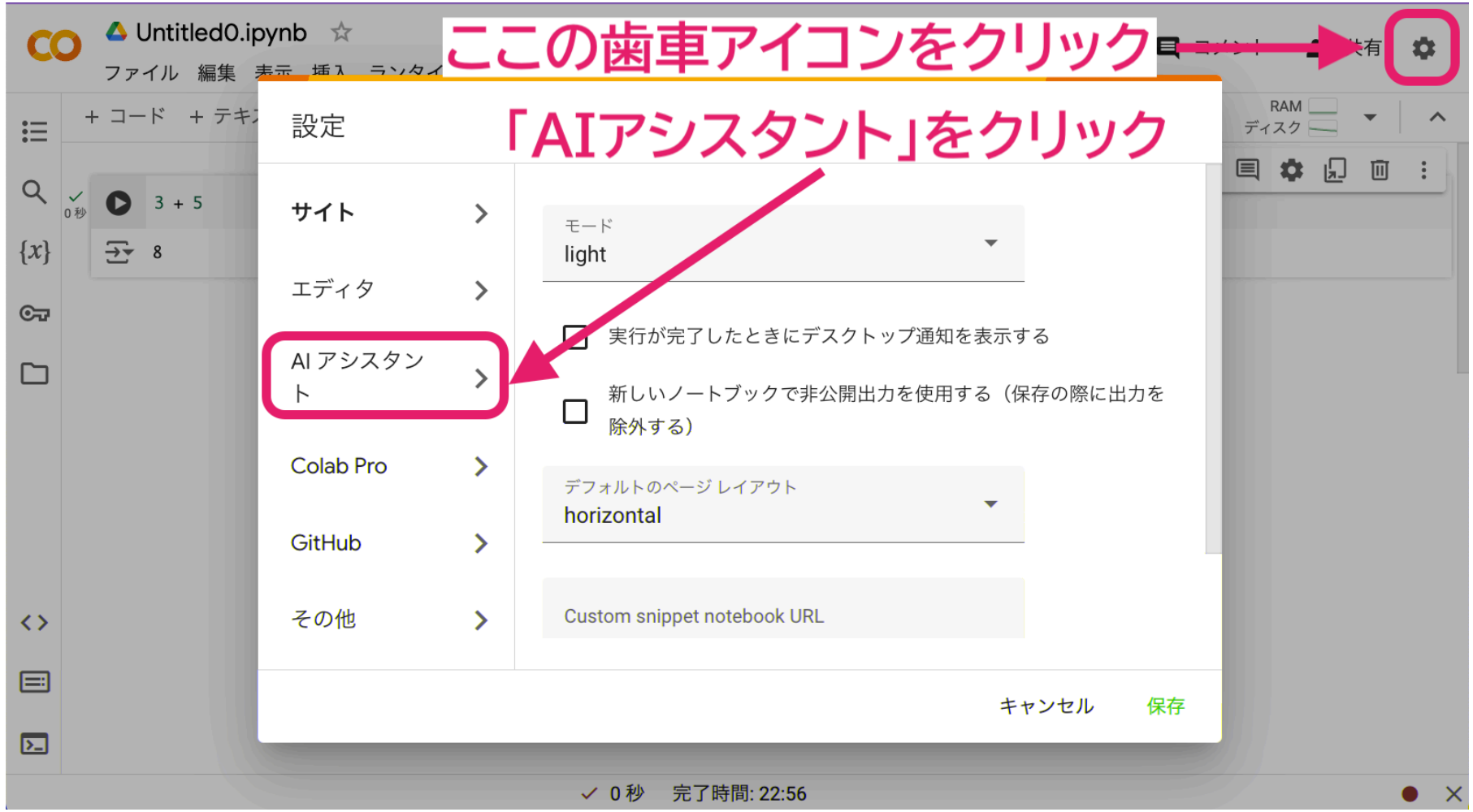


oogle Colabの使い方

-
-
-

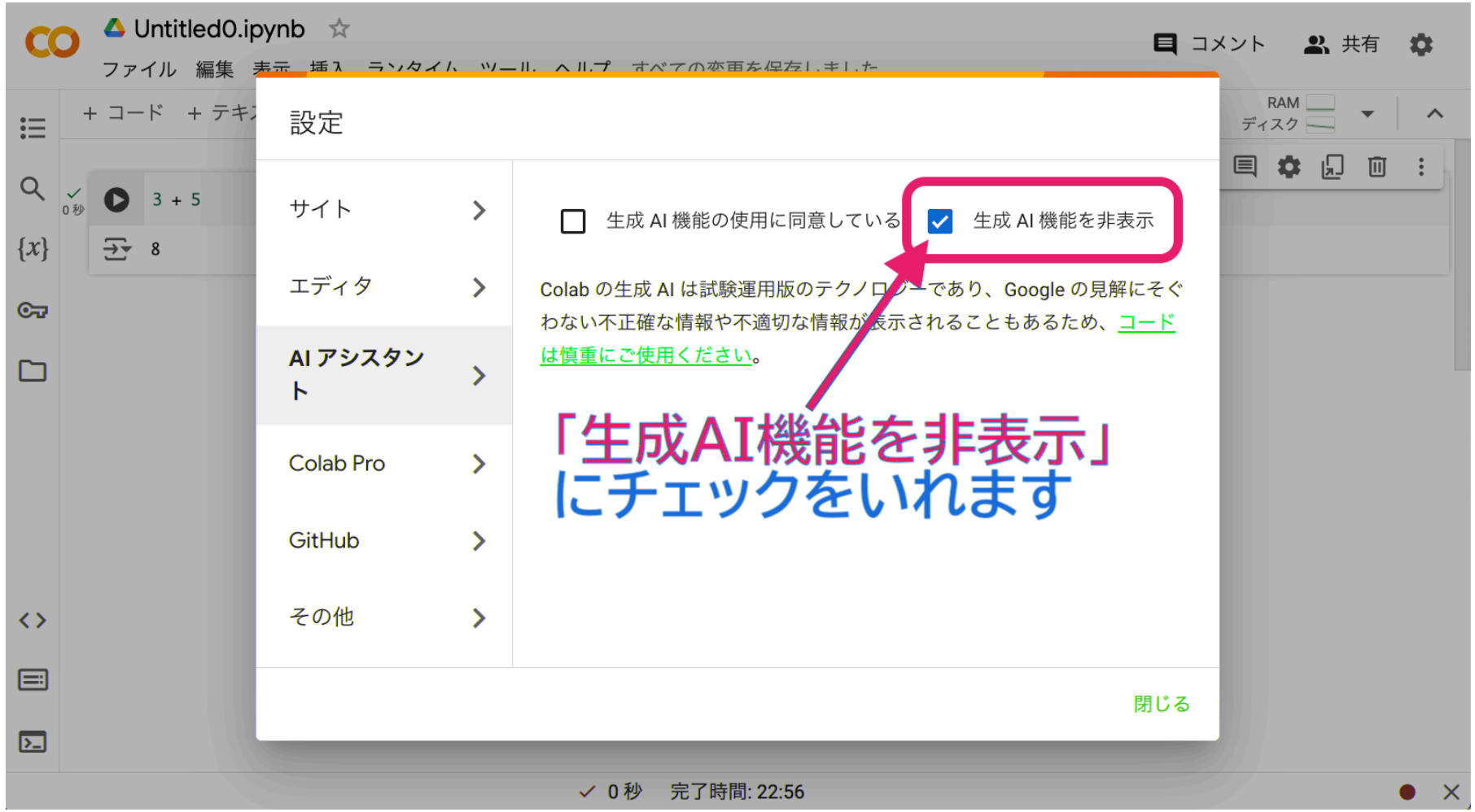
AIアシスタントの設定

Google Colab　　AI　　AI



AIアシスタントの設定

AI



Pythonのきほん

データの性質

-
- a Hello
- 1 -20
-

```
print(1)
```

```
print('hello world') or print("hello world")
```

データの性質

データ	概説	詳細
`str`		`'abc'`, `"hello world"`
`int`		`1`, `-500`
`float`		`1.23`
`bool`		`True` `False`
`list`		`[1, 2, 3]`
`tuple`		`(1,2,3)`, `1,2,3`
`dict`		`{'a':1, 'b':2}`

オブジェクトと関数

-
-
-
- `'hello'` str hello
- https://docs.python.org/ja/3/reference/datamodel.html
-
- `print()`

Pythonでの計算

: 2-2 Colab `print()` OK

+

`+`

```
print(10 + 30)
```

−

Pythonでの計算

`*`

`*` ×

```
print(10 * 30)
```

`**`

n a^n `**`

```
print(10 ** 3)
```

Pythonでの計算

`/`

`/` ÷

```
print(10 / 3)
```

`//`

`//`

```
print(10 // 3)
```

Pythonでの計算

`%`

`%`

```
print(10 % 3)
```

`divmod` **&**

Python

```
print(divmod(10, 3))
```

数値演算子の優先順位

```
print(10 + 5 * 2)
```

```
print((10 + 5) * 2)
```

異なるデータ型同士の計算

```
print('Hello' + 'World!')
```

```
print('Hello' * 5)
```

```
print('1' + 2)
```

異なるデータ型同士の計算

```
print('1' + 2)
```

```
TypeError: can only concatenate str (not "int") to str
```

("int"　　)str　　str

＝	意
`int()`	
`str()`	

オブジェクトを操作する

```
print('hello world!'.upper())
```

f ”　　hello world!

同じオブジェクトを使いまわす

-
-

```
hi = 'hello'
print(hi)
```

```
hi = 'world'
print(hi)
```

ある条件で処理を分ける

条件を判定する

	意
`<`	True
`<=`	True
`>`	True
`>=`	True
`!=`	True
`==`	True

条件を判定する

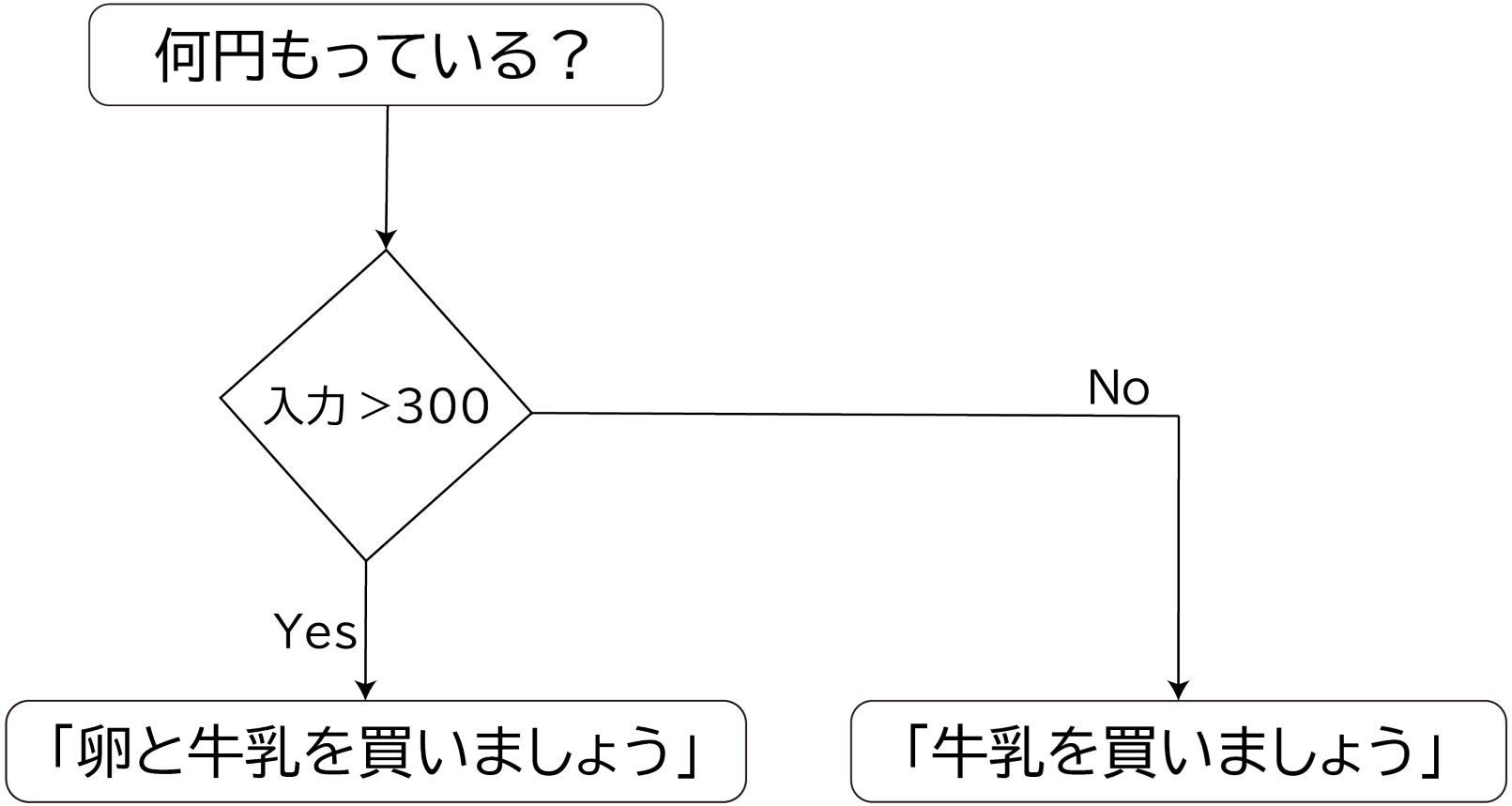
-

```
print(1 < 2)
```

```
print(1 > 2)
```

- True False
- True False

条件に応じて処理をする



条件に応じて処理をする

`if`

```
if 条件式:
    条件式がTrueのときの処理
else:
    条件式に当てはまらなかったときの処理
```

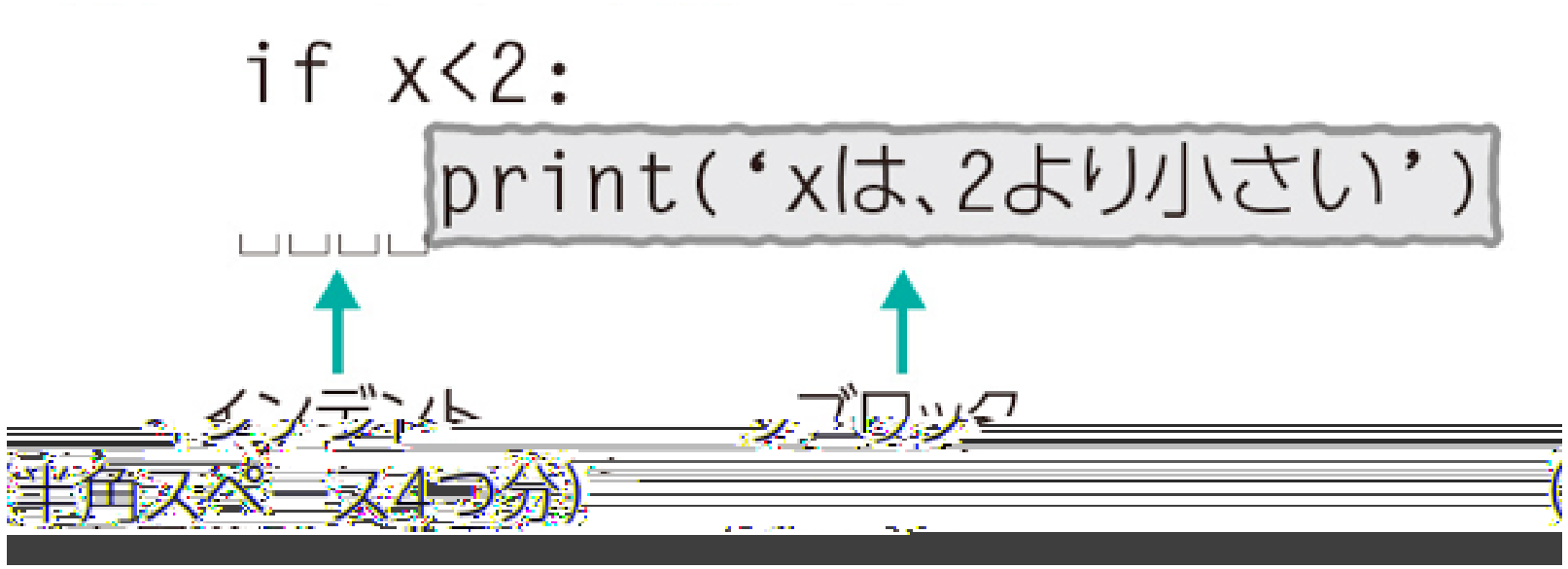
`else`

Python

条件に応じた処理をする

Colab

4

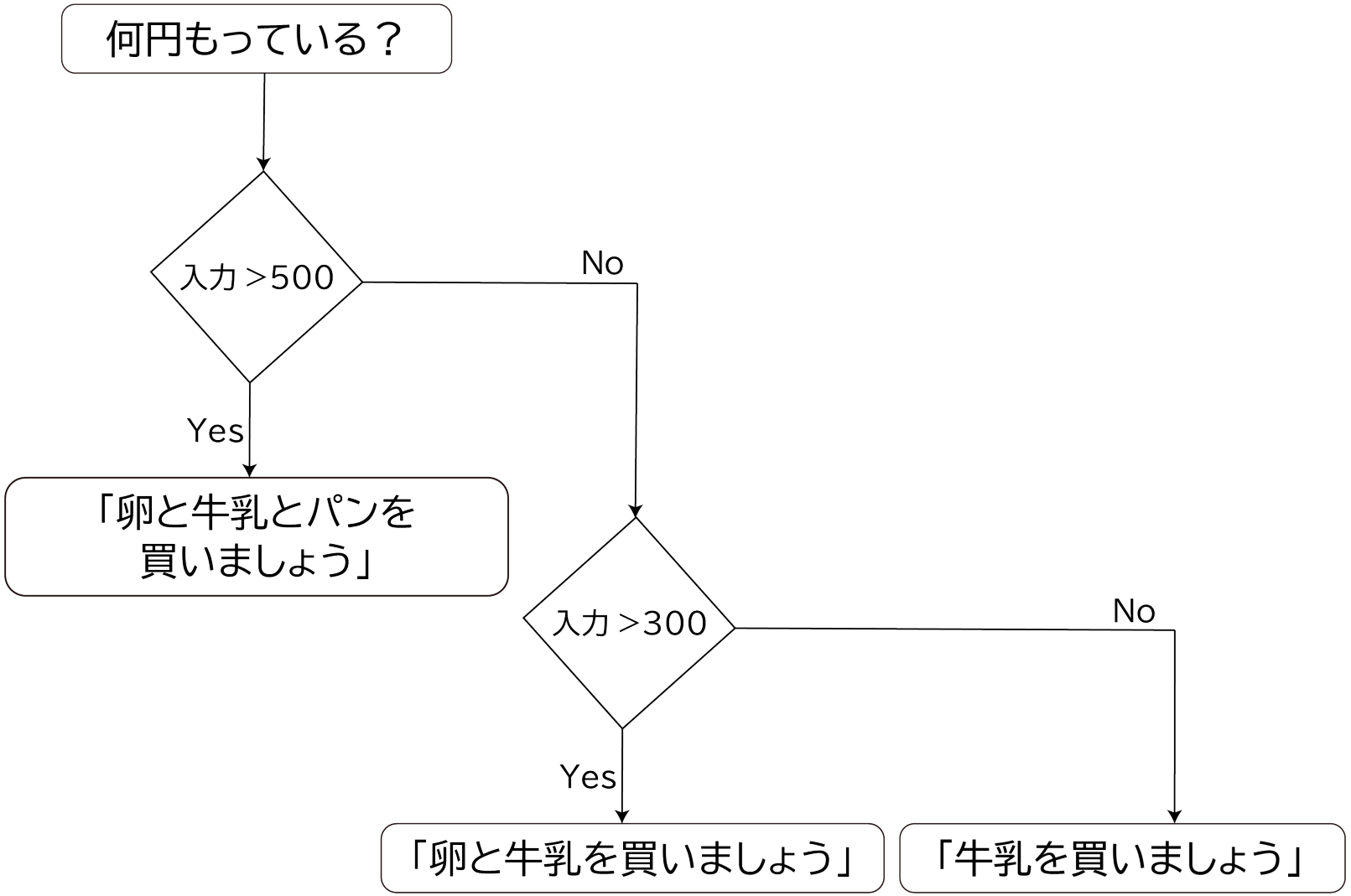


条件に応じた処理をする

P.28

```
money = 400
if money > 300:
    print('卵と牛乳を買いましょう')
else:
    print('牛乳を買いましょう')
```

条件に応じた処理をする



条件に応じた処理をする

`elif`　`else if`

```
if 条件式1
    処理1
elif 条件式2
    処理2
elif 条件式3
    処理3
 :
else:
    その他処理
```

P.32

オブジェクトの扱い

中身をあとから変更できるリスト型

- データ
- ス　ト(list)
- `[]`

```
[1]
[1,2,3]
['apple', 'orange', 'banana']
```

中身をあとから変更できるリスト型

- `[]`
- `,`( )
- 白
- OK
-

```
[1,2,3] # カンマで区切って複数のオブジェクトを扱う
['apple', 'orange', 'banana'] # 文字列も同じように扱える
[] # 要素が無い（空）のリストも作成できる
[[1, 2], [3, 4]] # リストの中にリストをいれることもできる
```

リスト型のインデックス

-
- `リスト型[インデックス]`
-

```
fruits = ['apple', 'orange', 'banana'] # リスト作成
print(fruits[0]) # 0番目の要素を表示
print(fruits[1]) # 1番目の要素を表示
fruits[1] = 'grape' # 1番目の要素を「grape」に変更
print(fruits) # リストの中身を確認
```

リスト型にオブジェクトを追加

-
- `append()`

append()

```
number = [1, 2, 3] # リスト作成
number.append(4) #  「number」リストに要素「4」を追加
print(number) # リストの中身を確認
```

リストからオブジェクトの取り出し

- `pop()`
-
-
-

pop()

```
alphabet = ['A', 'B', 'C', 'D'] # リスト作成
char_C = alphabet.pop(2) # アルファベット「C」のインデックスを指定して取り出す
print(char_C) # 中身を確認
print(alphabet) # リストの中身を確認
char_last = alphabet.pop() # インデックスを指定しない場合最後のオブジェクトを取り出す
print(char_last) # 中身を確認
print(alphabet) # リストの中身を確認
```

リストの中のデータを確認する

- `オブジェクト in リスト`
- `オブジェクト not in リスト`

```
Kanto = ['Tokyo', 'Kanagawa', 'Chiba', 'Saitama', 'Ibaraki', 'Tochigi', 'Gunma'] # リスト作成
if 'Gunma' in Kanto: # GunmaがKantoリスト内に存在するか判定
    print('Gunmaは関東地方です') # 処理
else: # それ以外
    print('Gunmaは関東地方ではありません') # 処理

if 'Yamanashi' not in Kanto: # YamanashiがKantoリスト内に存在しないか判定
    print('Yamanashiは関東地方ではありません') # 処理
else: # それ以外
    print('Yamanashiは関東地方です') # 処理
```

中身をあとから変更できないタプル型

-
-
- `()`

```
() # 空のタプル型の生成
(1,2) # 複数のオブジェクトで構成されたタプル
(1,) # オブジェクトが1つの場合でもカンマをつける
1,2 # ()がなくてもタプル型になる
1, # ()がない場合でオブジェクトが1つの場合もカンマをつけることでタプル型になる
```

中身をあとから変更できないタプル型

- ,
- 1 ,
- () ,

```
sample_tuple = (10, 20) # タプルの生成
x, y = sample_tuple # タプルの中身を展開してx, yに代入
print(x)
print(y)

sample_tuple2 = 'A', 'B', 'C' # タプルの生成
a, b, c = sample_tuple2 # タプルの中身を展開してx, yに代入
print(a)
print(b)
print(c)
```

キーと値をセットで扱う辞書型

- (dict)
- キー=
- (バリュー)=
-
- {}
- キー:バリュー
- ,

dict

```
{} # 空のdict型
{'tea' : 100} # キーとバリューを格納
{'tea' : 100, 'coffee' : 200} # 複数のキーとバリューを格納
```

辞書型でキーを指定する

- dict
-
-

辞書名[キー]

辞書名[追加キー]=追加バリュー

dict

```
my_dict = {'tea' : 100, 'coffee' : 200} # dict型の作成
print(my_dict['tea']) # キーを指定して対応するバリューを表示

my_dict['milk'] = 300 # 新たにキーとバリューを追加
print(my_dict) # dict型の中身を確認
```

キーと値をセットで扱う辞書型

-
- `期待するキー名 in 辞書名`
-

```
my_dict = {'tea' : 100, 'coffee' : 200, 'milk' : 300} # dict型の作成
'tea' in my_dict # my_dict内に'tea'があるか確認
```

課題

課題2-1

- Moodle SCfCL-2nd-prac.ipynb Colab
- File > Download > Download .ipynb .ipynb
- .ipynb Moodle
- **10 3 ( ) 20 まで**