

# 情報リテラシー演習

## 第11回（講義）

2025年6月19日（木）／6月20日（金）

# 授業計画

- 第1回：ノートPC環境構築①（学内ITの利用設定など）
- 第2回：ノートPC環境構築②（ITツールのインストール確認など）
- 第3回：電子メールの使い方、Googleドライブ等のサービスの使い方、MS PowerPointによるプレゼンテーション資料作成方法
- 第4回：MS Excelによる関数利用、表・グラフ作成方法
- 第5回：MS Wordによる文書作成方法
- 第6回：MS Officeを用いたレポート作成の演習
- 第7回：Linuxの操作方法①（基本コマンド）
- 第8回：Linuxの操作方法②（テキストエディタviなど）
- 第9回：Linuxの操作方法③（応用コマンド）
- 第10回：開発基礎①（VSCode）
- 第11回：開発基礎②（Git、GitHub）
- 第12回：開発基礎③（Docker）
- 第13回：Linuxの操作方法及び開発基礎の総合演習
- 第14回：理解度チェック、デバッガーの使い方



# 本日のテーマ

- VSCodeでのGitの使い方
- GitHubの使い方

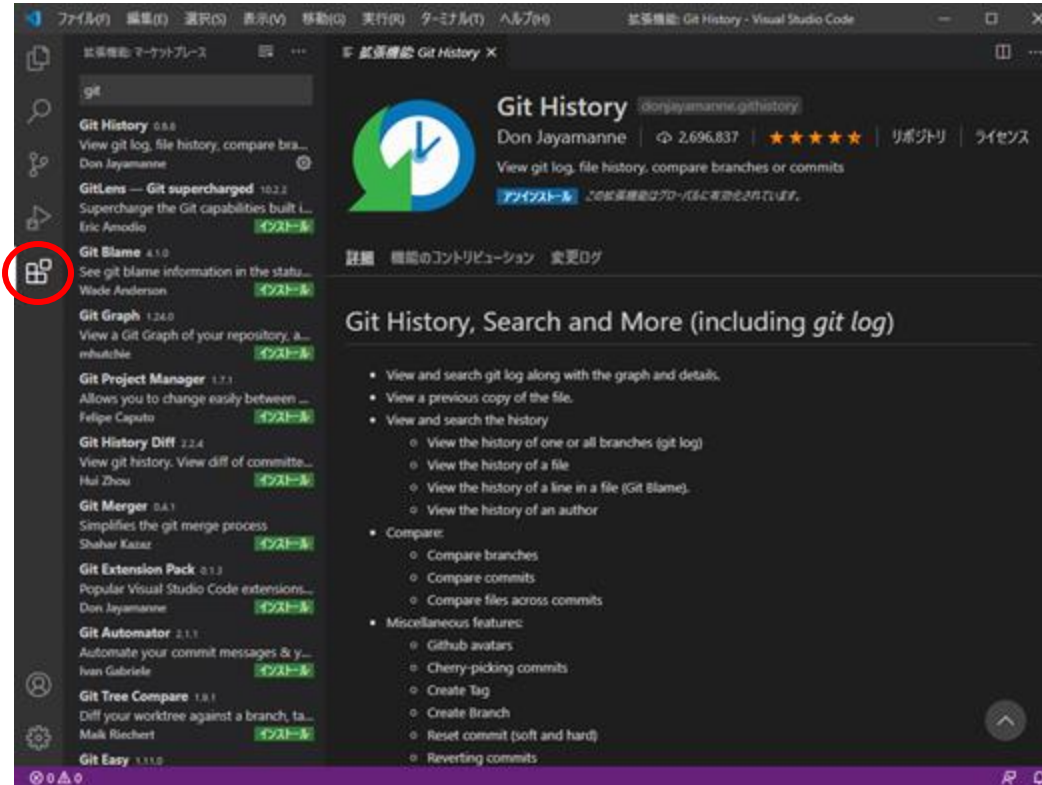
## 演習前の準備

- 学生ポータルで出席登録をする
- 講義ページから、本日の講義資料をダウンロードする

# 事前準備：拡張機能のインストール

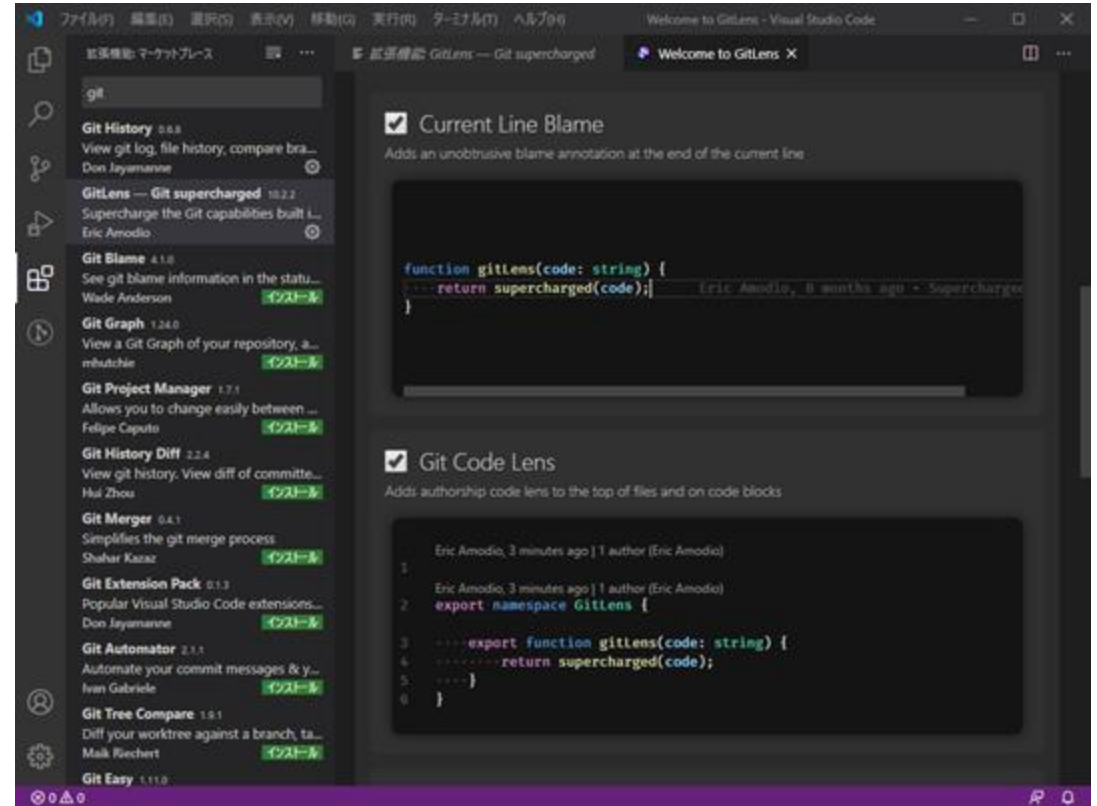
# Git History 拡張機能のインストール

- リポジトリのコミット履歴（git log）をグラフィカルに表示
  - 左側アクティビティバーの拡張機能アイコンをクリックし、検索ボックスに「Git」と入力し**Git History**を検索し「インストール」を実行



# Git Lens 拡張機能のインストール

- リポジトリのファイル変更履歴の詳細を表示
  - 左側アクティビティバーの拡張機能アイコンをクリックし検索ボックスに「Git」と入力しGit Lensを検索し「インストール」を実行





# Gitによるファイルの履歴管理の基礎



# Gitとは？

ソースコードのバージョンを管理するツール

- バージョンを管理するメリット
  - 変更前のソースコードの違いを調べることができる
  - 変更する前の状態に戻すことができる
- Gitを使うことでこれらの管理が容易にできるようになる



# Gitはどんなときに使う？①

- 自分の変更履歴を確認することができる

test.py



...



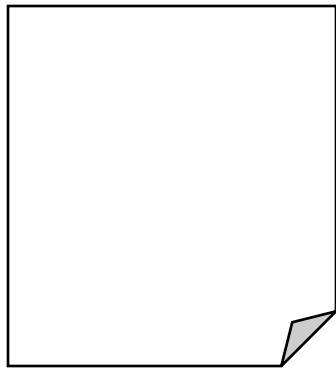
今までに、いつ・どんな変更をしたかの履歴を見ることができる

# Gitはどんなときに使う？②

- 編集した履歴を確認したり戻ったりできる

例えば「test.py」をどんどん編集していたがある時不具合が  
起こった時

test.py



...



問題なく動作していたころに  
戻したい・・・！

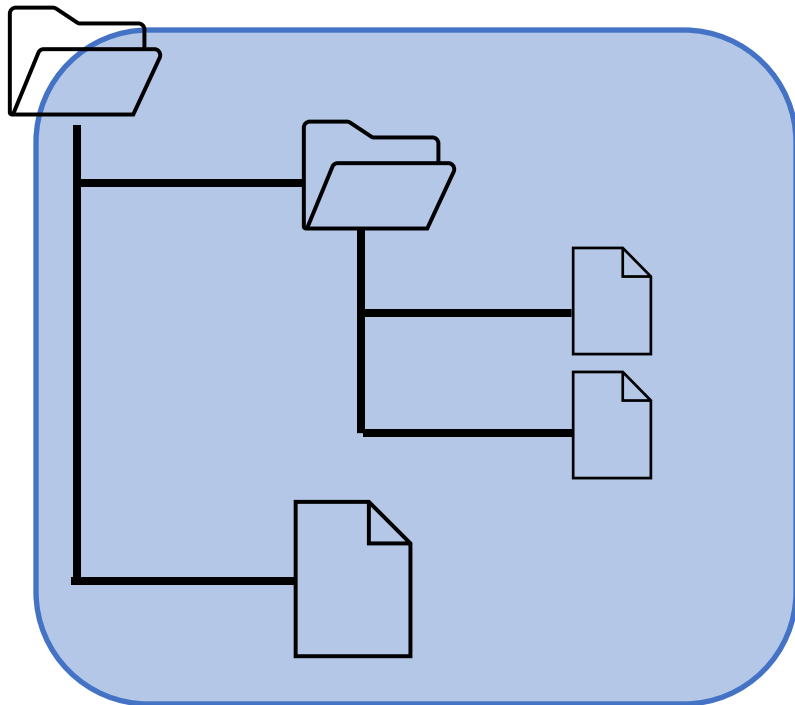
不具合が発生して動作しなくな  
った



プログラムが動いていたときまで戻したい  
→Gitをつかえば容易に元に戻すことができる

# Gitの基本用語：リポジトリ

- リポジトリとは？
  - 管理したいコードの一つのまとまり

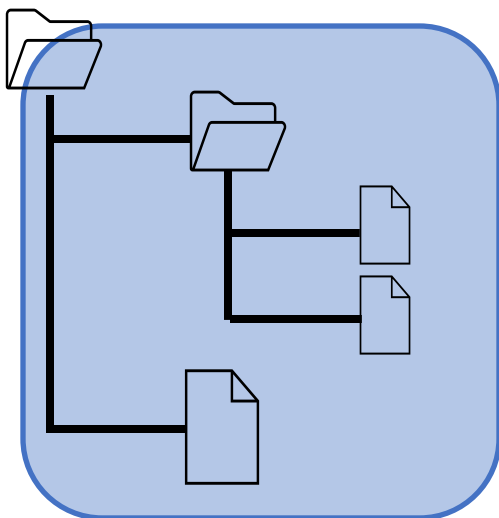


ある開発に必要な複数のファイルを管理する場所＝リポジトリ

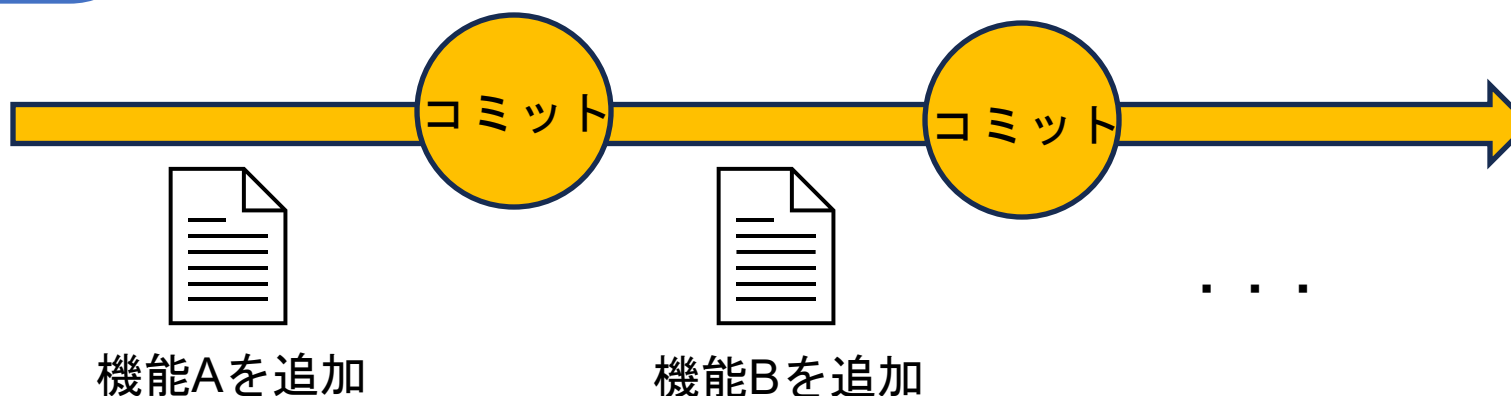


# Gitの基本用語：コミット

- ソースコードの変更・追加・削除を記録すること

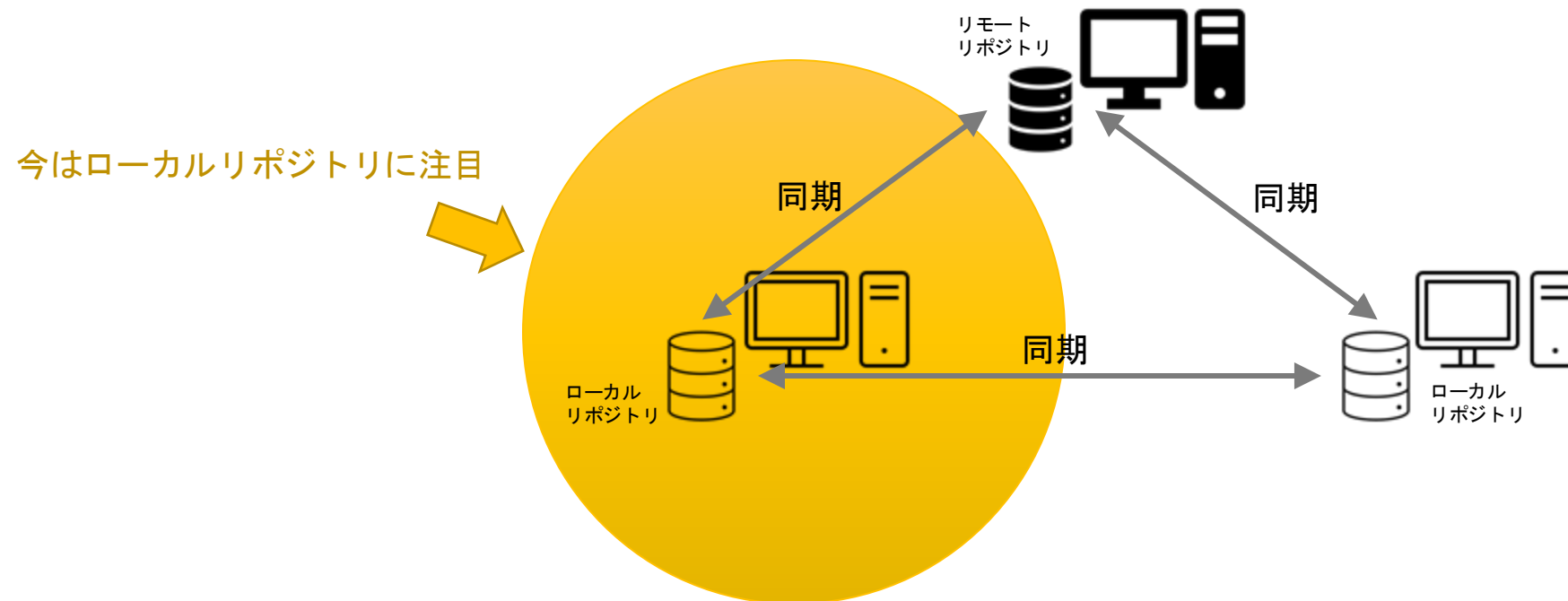


リポジトリの中のファイルにどんどん機能を追加していく  
→区切りのいいところでコミットし作業の記録をとる



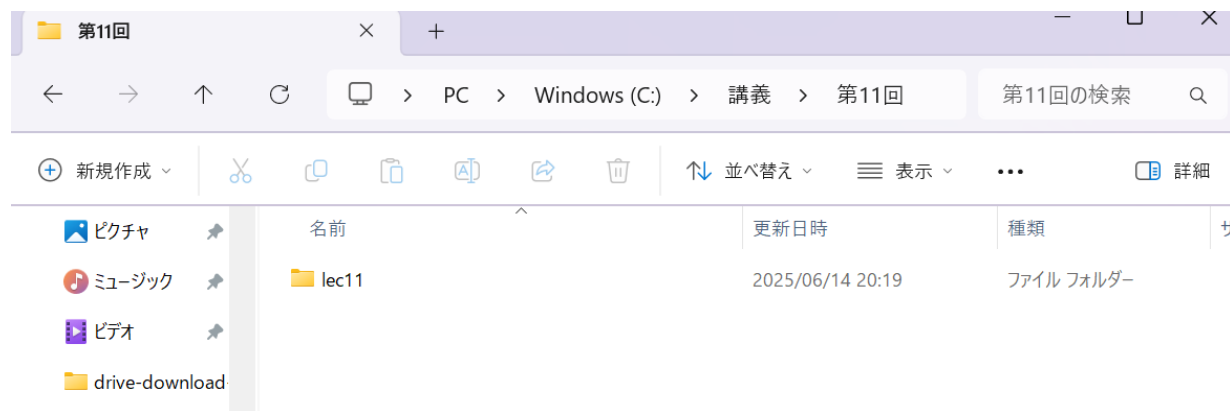
# TU ローカルリポジトリを作成してみよう

- 自分のPCに、リポジトリを作成する



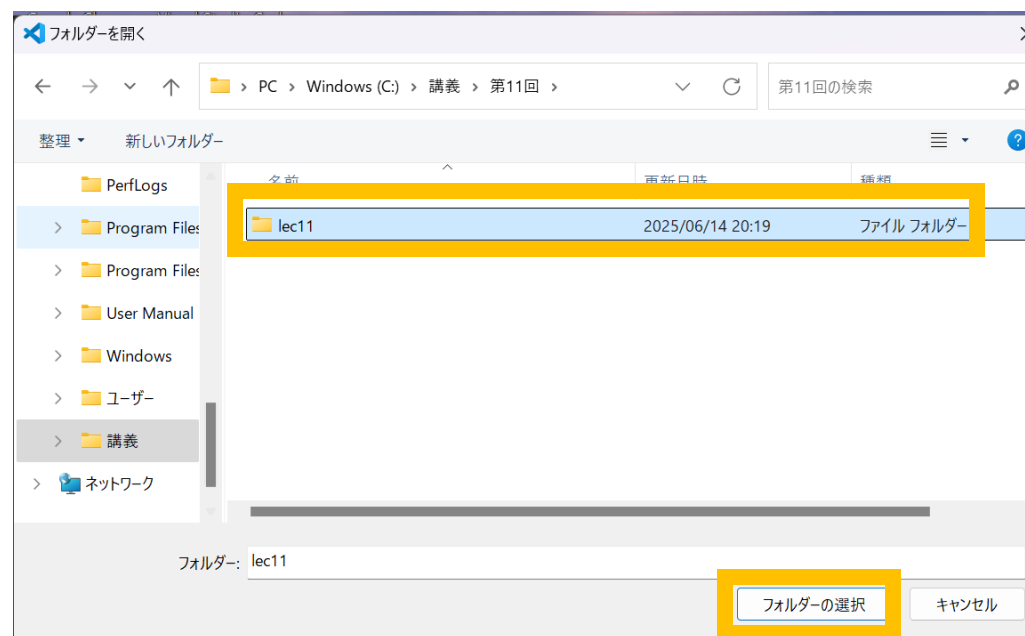
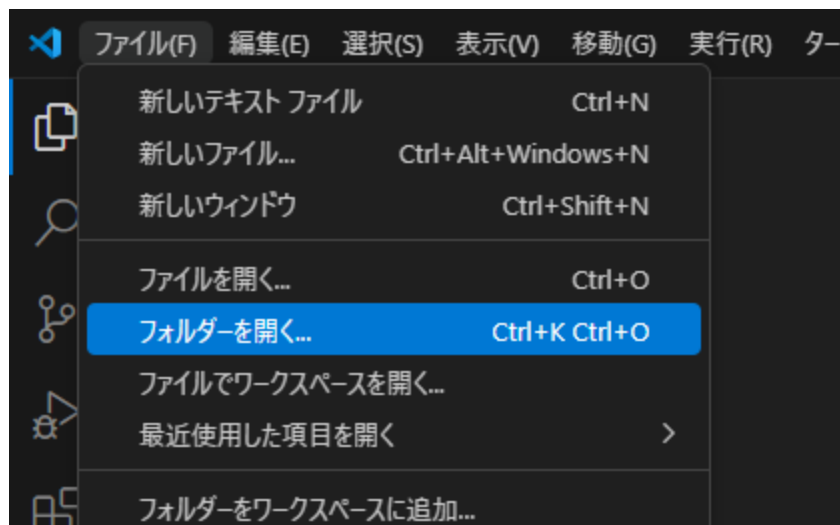
# 授業フォルダの作成

- /Cドライブ/講義/情報リテラシー演習/第11回 のフォルダを作成する
- 今回の演習用に第11回のフォルダ直下に「lec11」のフォルダを作成する
- 以降の演習はlec11のフォルダで作業する



# フォルダの選択

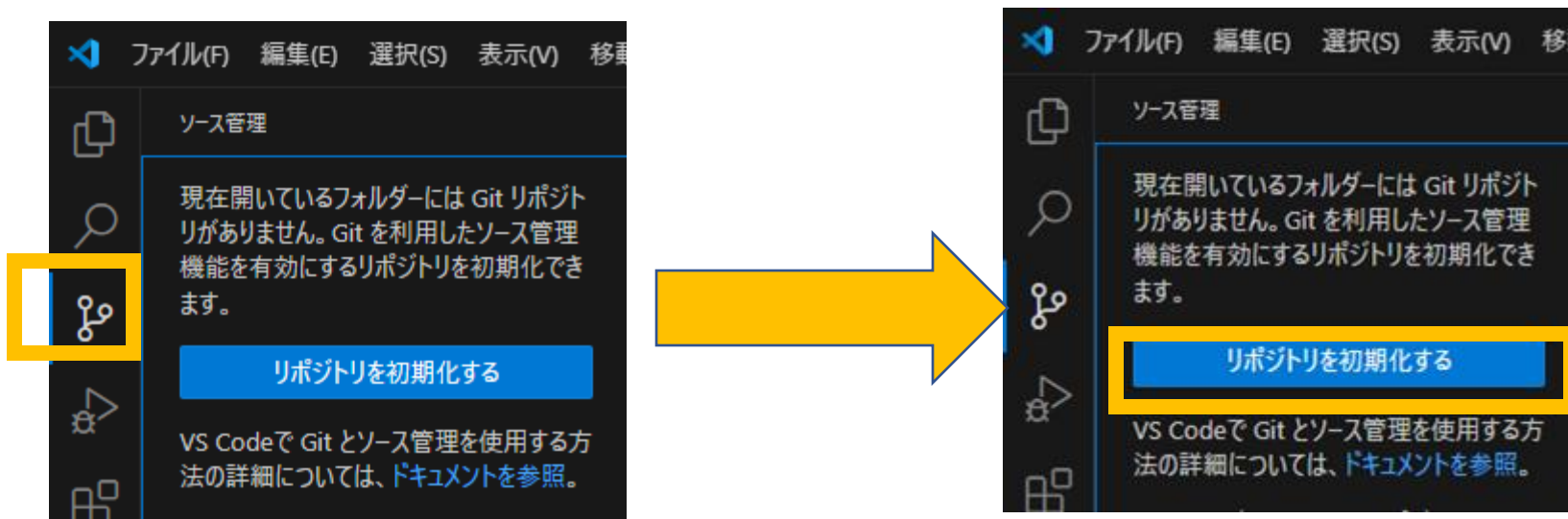
- VSCodeでフォルダを開く



メニューから「ファイル」→「フォルダーを開く」

# ローカルリポジトリの初期化

- 11回のフォルダが開かれていることを確認して以下を行う  
 サイドバーのソース管理を表示して、リポジトリの初期化アイコンをクリック

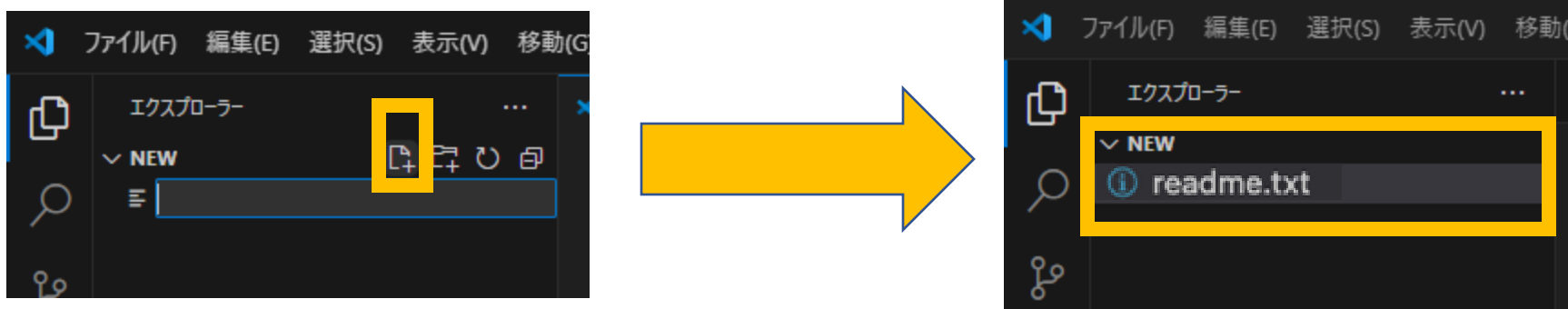


※初期化するディレクトリを聞かれるが、現在VSCodeで開いている場所が初期表示されているのでそのまま選択。

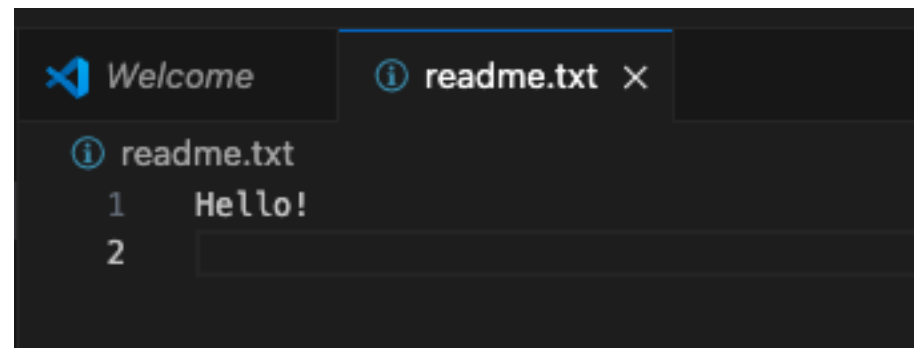


# ファイルの作成

- ファイルアイコンを選び新しくファイルを作成する  
ファイルは「readme.txt」とする



作成したファイルに「Hello ! 」と記入して保存



# ステージング

- ステージングとは：次にコミットする内容を準備しておく場所に追加すること
- ファイルの作成と編集を行うとサイドバーのソース管理に更新ファイルの一覧が表示される
- +アイコンでステージングされる



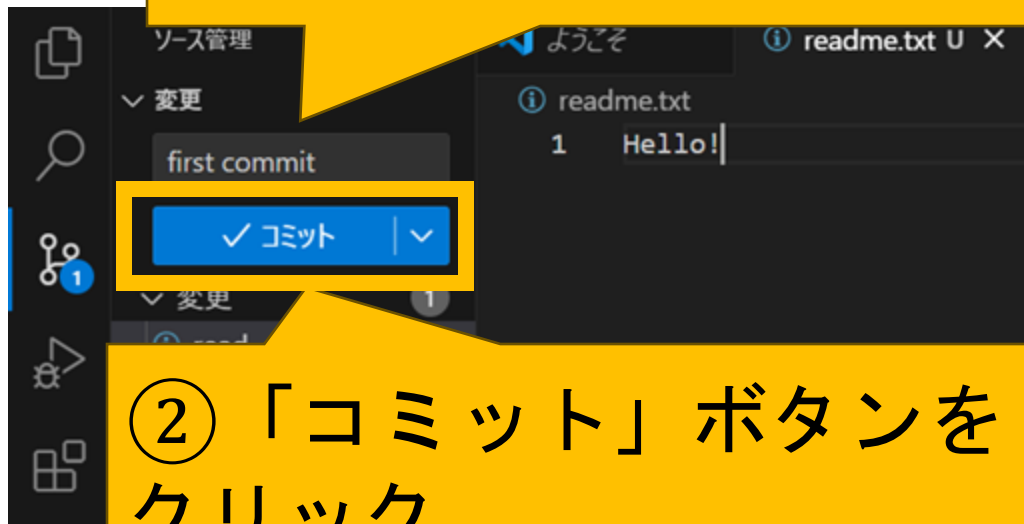
① ソース管理のアイコンをクリック

※変更したファイルがあるときはアイコンに数値が表示される

# コミット

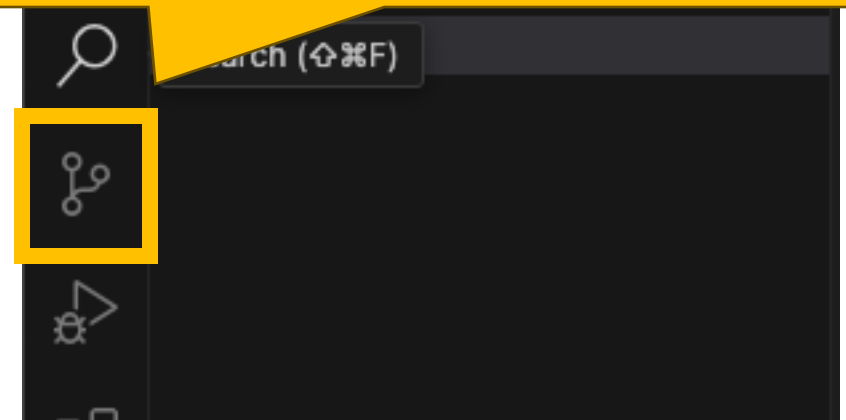
- ・コミットとは、ステージに追加された「暫定」状態にあるファイルを「確定」させるため行う

① コミットメッセージを入力「first commit」



② 「コミット」ボタンをクリック

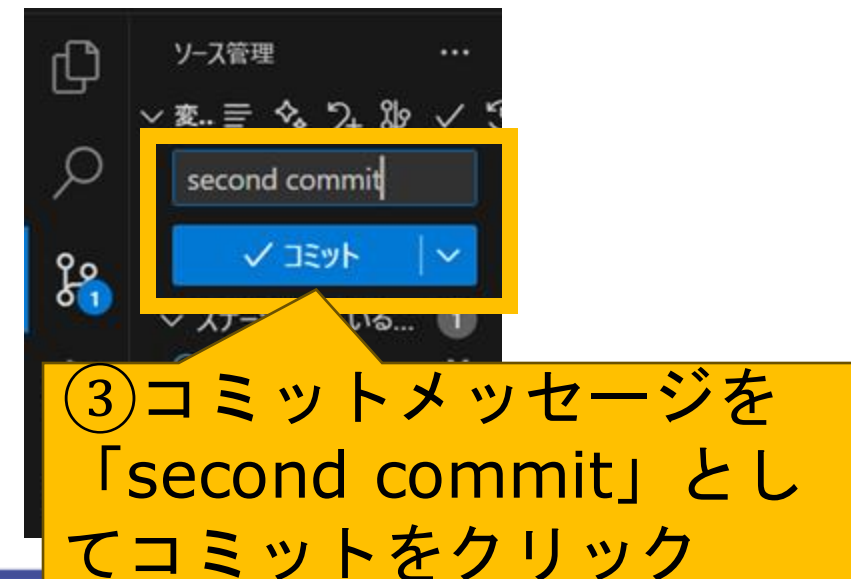
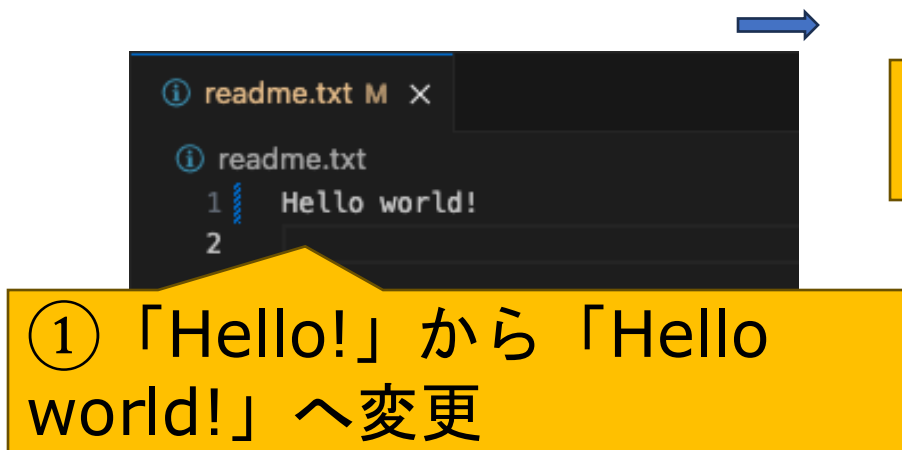
アイコンに表示されていた①が消えている



# ファイルの編集

- readme.txtの内容を編集してみよう

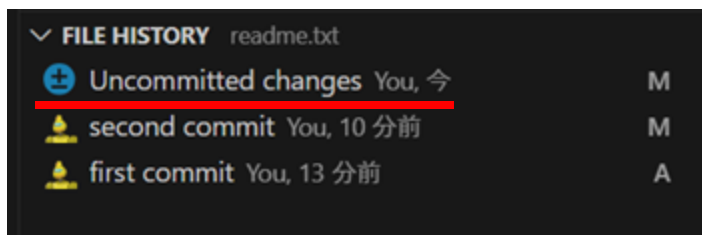
- ①最初に書いた「Hello!」を「Hello world!」に変更して保存する
- ②「ソース管理 」を選択して、ファイル名の隣の+をクリックしステージング
- ③コミットメッセージを「second commit」としてコミットボタンをクリック



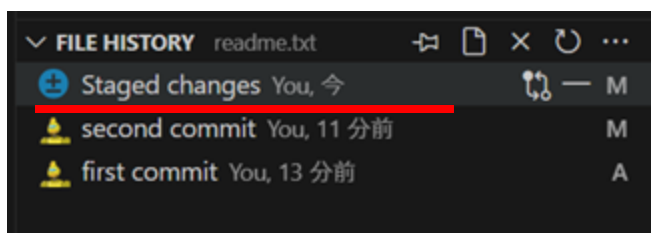
# ステータスの確認 (GitLens)

- 今までのコミットはサイドバーのGitLensで見ることができる

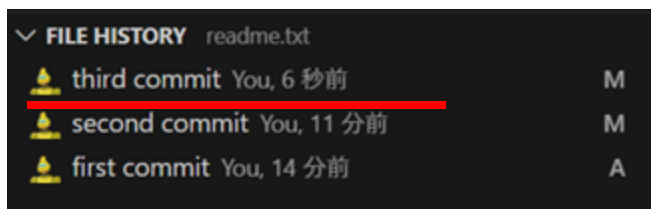
- ・ 新規追加ファイル (ステージング未済)



- ・ ステージング済 (コミット予定)





- ・ コミット済





ステージング

コミット

# 練習

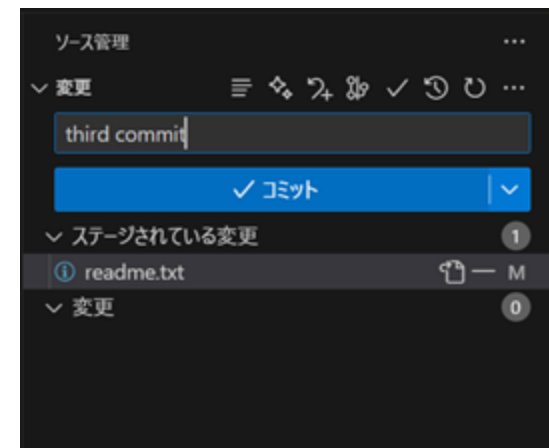
- ファイルの編集, ステージング, コミットをやってみよう
  1. 「readme.txt」の2行目に「情報リテラシー演習」と記入して編集し、内容を保存する
  2.  をクリックして、ファイル名の横の「+」をクリックしてステージングする
  3. コミットメッセージに「third commit」と入力してコミットする
  4.  のFile Historyからコミットが完了していることを確認

# 練習（解答）

- ファイルの編集，ステージング，コミットをやってみよう
  1. 「readme.txt」の2行目に「情報リテラシー演習」と記入して編集する
  2.  をクリックして，ファイル名の横の「+」をクリックしてステージングする
  3. コミットメッセージに「third commit」と入力してコミットする
  4.  のFile Historyからコミットが完了していることを確認

```

i readme.txt
    You, 1 秒前 | 1 author (You)
1  Hello world!
2  情報リテラシー演習
3
  
```

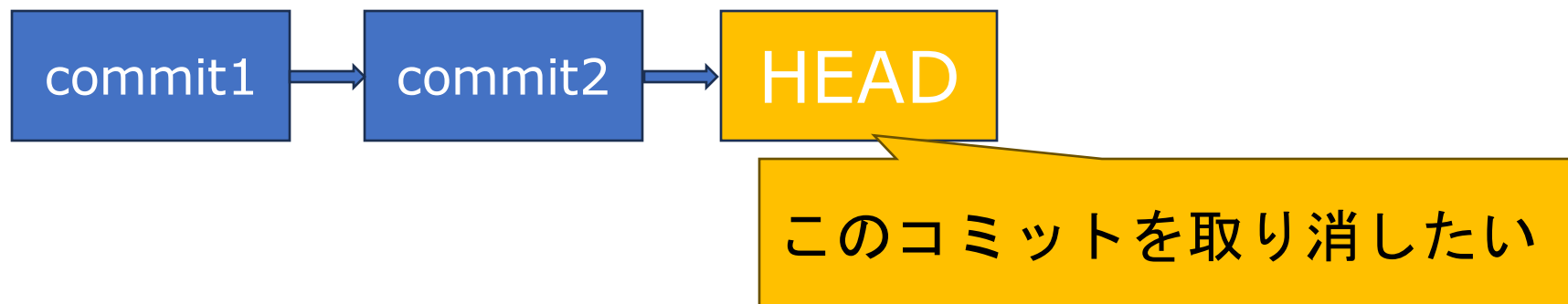


FILE HISTORY readme.txt		
	third commit You, 6 秒前	M
	second commit You, 11 分前	M
	first commit You, 14 分前	A

# Gitにおけるコミットの取り消し

- コミットを取り消したい時

- ①間違えてコミットしてしまった (soft reset)
  - ②そもそもこの編集は必要なかった (hard reset)
- 巻き戻す対象によって2種類の取り消しがある



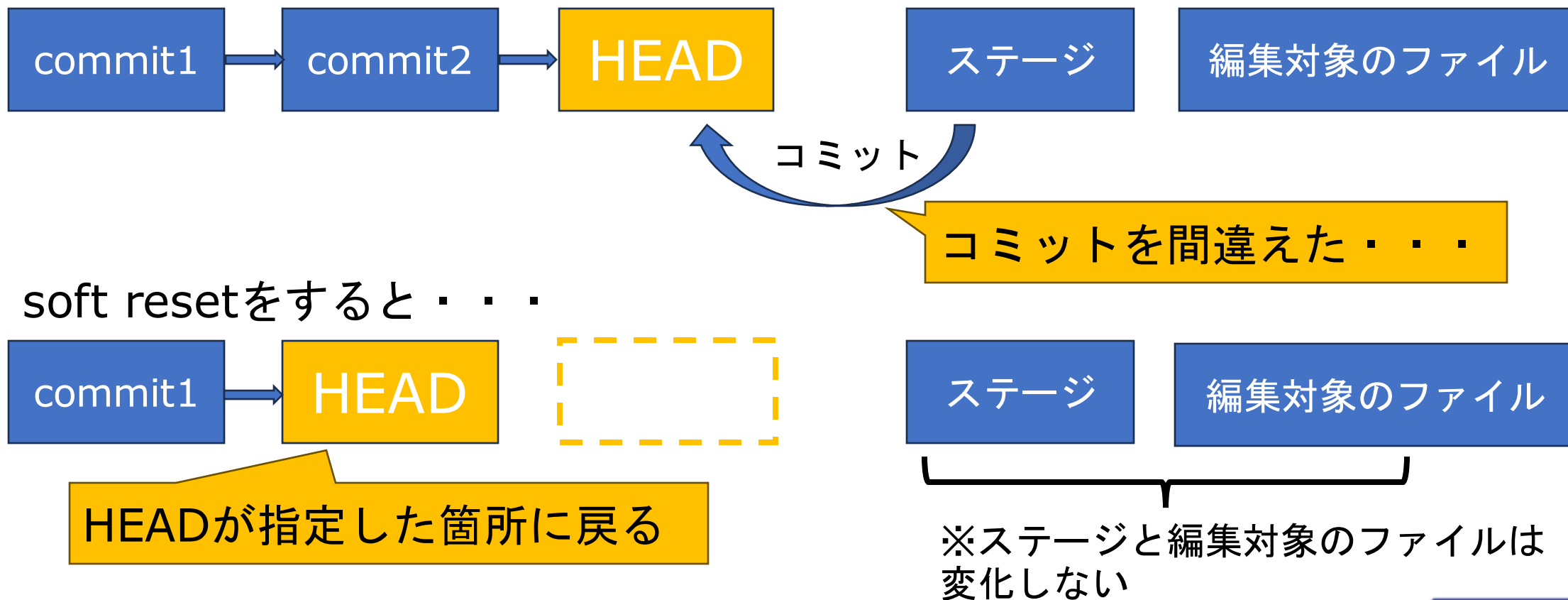
※HEADは現在コミットしているポイント





# コミットの取り消し①

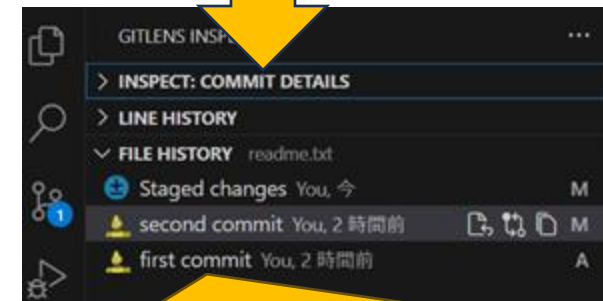
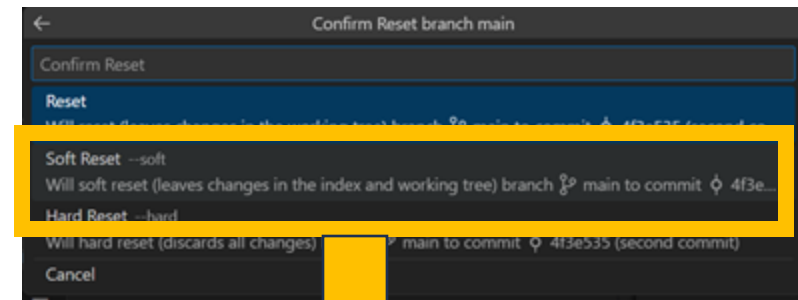
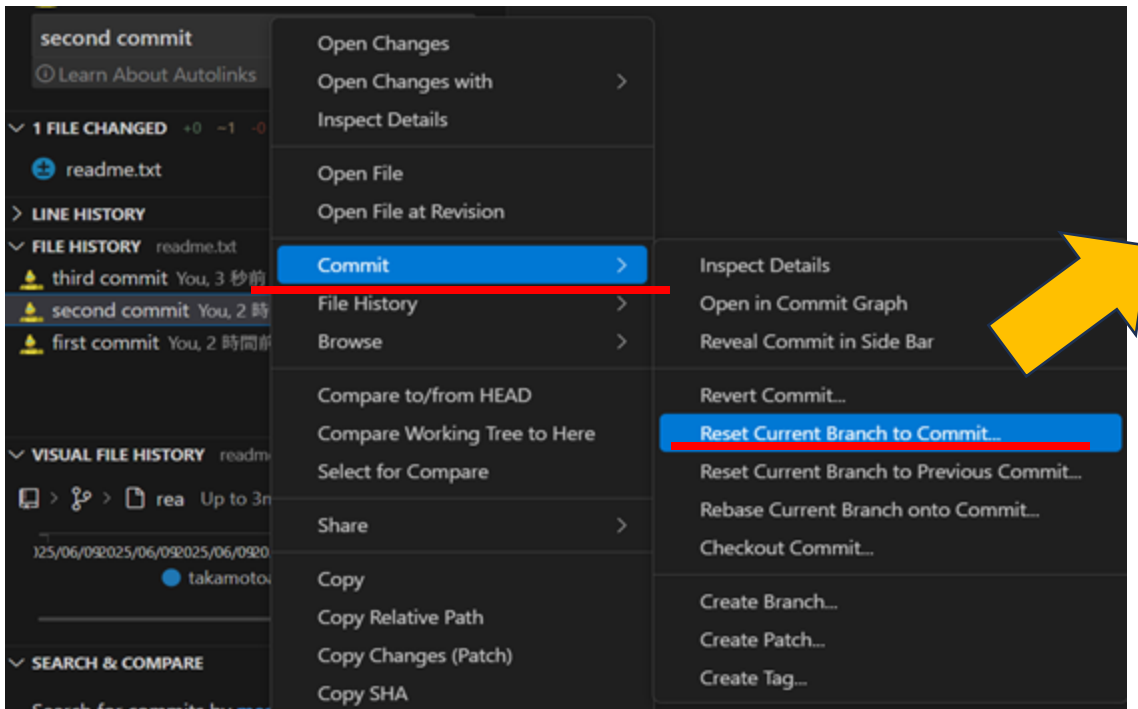
- 直前のコミット操作を取り消すsoft reset  
→編集対象のファイルとステージはそのまま



# コミットの取り消し①

## • GitLensを使ったsoft reset

①戻したいコミットを右クリック (second commit)  
→「Commit」→「Reset Current Branch to commit」→soft reset

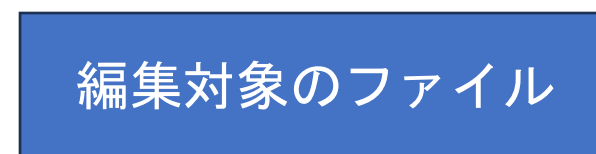
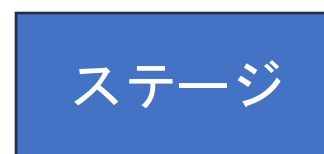


second commitの直後までコミットが戻る

# コミットの取り消し②

- 過去のコミット直後に戻るhard reset

→HEADのみでなく，ステージ，ファイル上の変更もすべて取り消す



hard resetをすると・・・

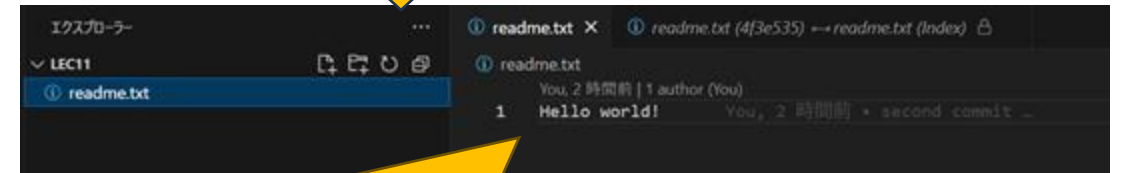
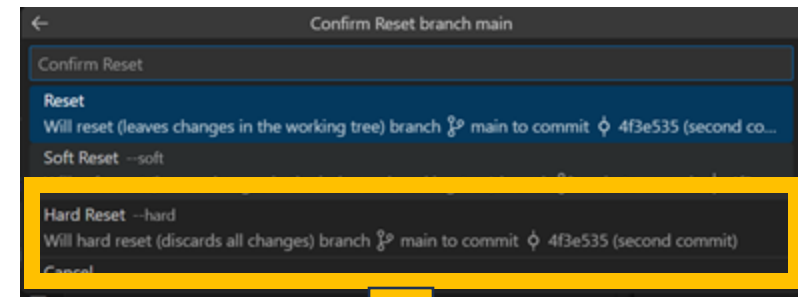
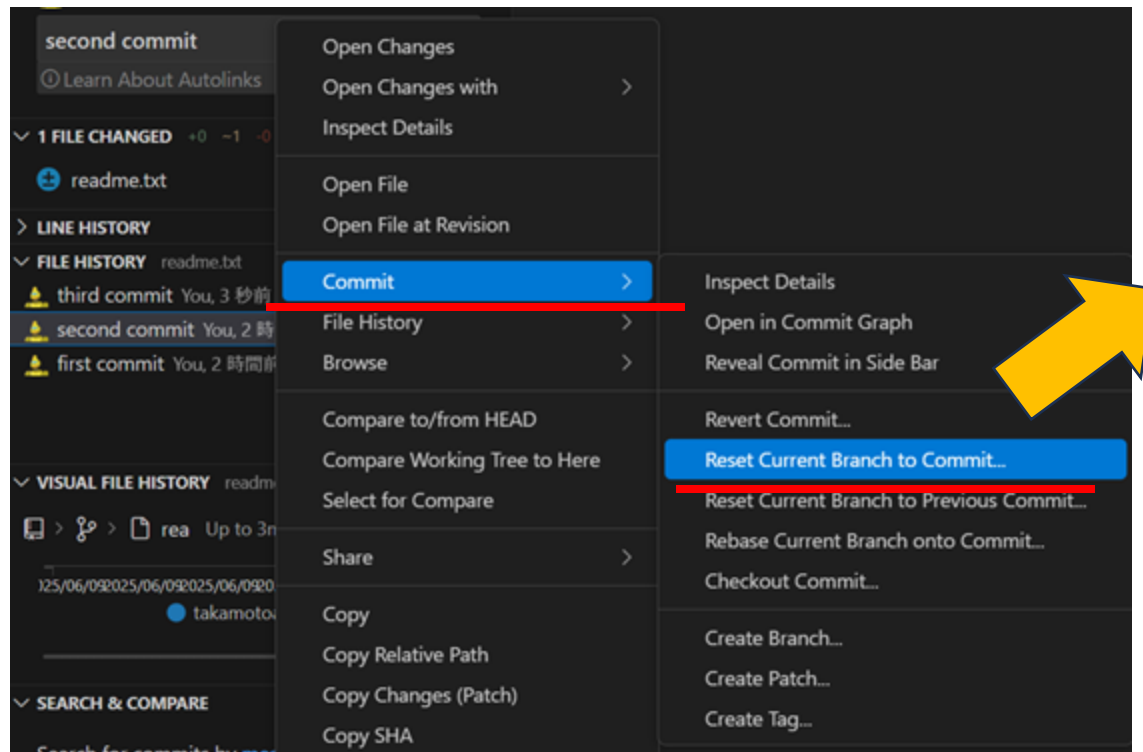


※ステージと編集対象のファイルもリセットした時の内容に巻き戻る

# コミットの取り消し②

- GitLensを使ったsoft reset

①戻したいコミットを右クリック (second commit)  
→「Commit」→「Reset Current Branch . . .」→hard reset



編集した「情報リテラシー演習」の文字列  
が消えている（変更した内容も消える）

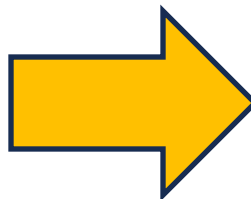
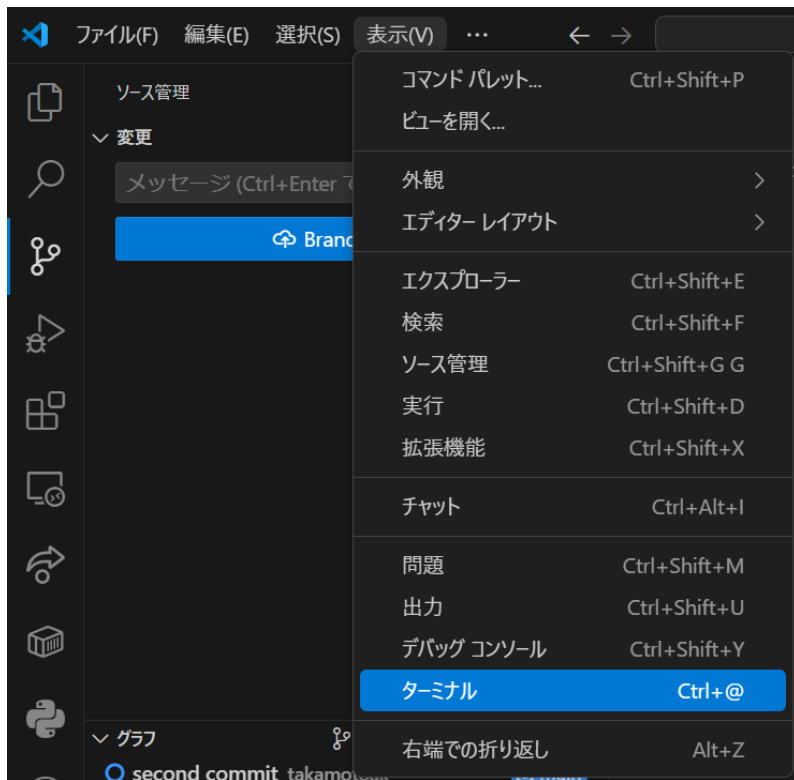
# コミットの取り消し まとめ

Resetの種類	HEADの移動	ステージの変更	編集対象のファイルの変更	主な用途
soft reset	○	×	×	コミットのみを巻き戻す
hard reset	○	○	○	編集対象のファイルを含め全ての内容を巻き戻す

※Hardリセットはコミットしていない内容全てが消えるので注意



# Gitをコマンドでやってみよう

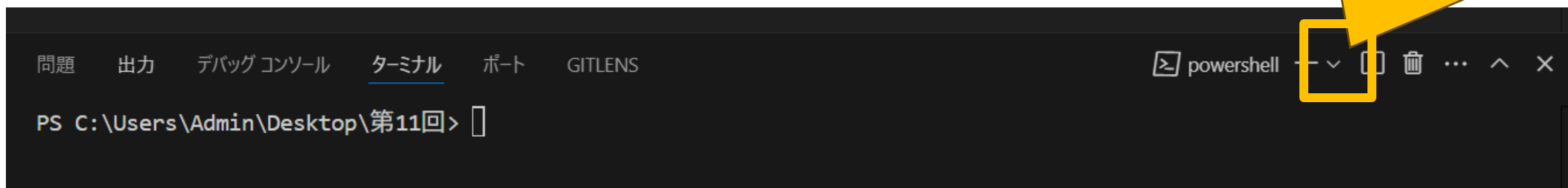
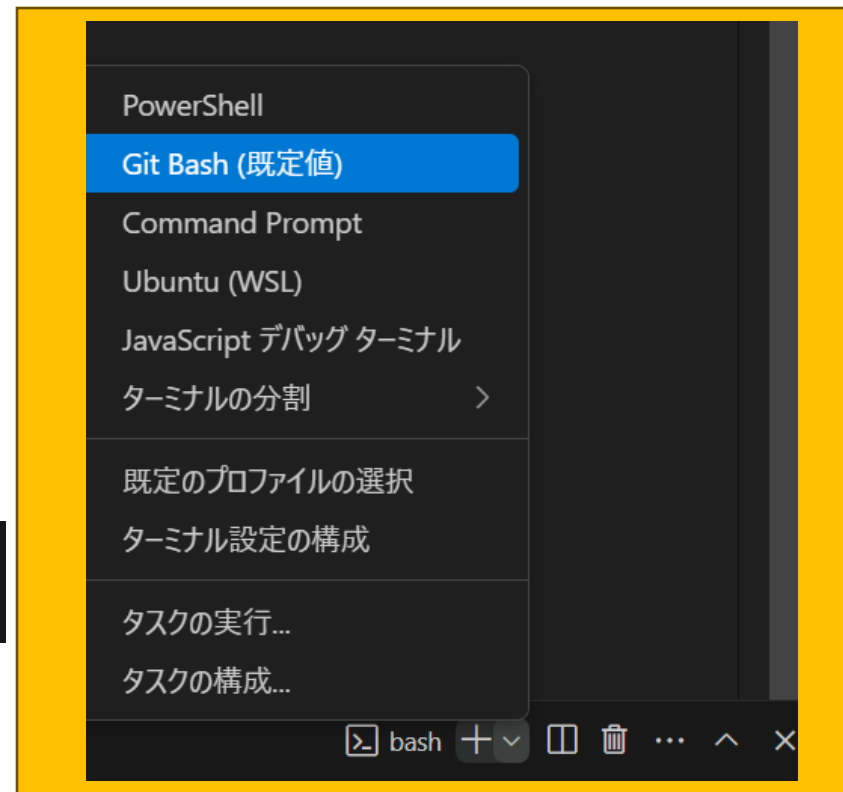
- ターミナルの準備  
「表示」→「ターミナル」をクリック



# Gitをコマンドでやってみよう

- ターミナルの上部をクリックし、「Git Bash」を選択する
- 表示の色が以下のようなになればOK

```
Admin@MSI MINGW64 /c/講義/第11回/lec11
$   を押して、 に何らかの操作を依頼します。 入力を開始して閉じます。
```



# ① ファイルを作成

- コマンドでファイルを新規作成する (test.txt)

```
$ cd lec11
$ vi test.txt
```

```
Admin@MSI MINGW64 /c/講義/第11回/lec11
$ vi test.txt

Admin@MSI MINGW64 /c/講義/第11回/lec11
$ cat test.txt
C0XXXXXX
```

- `vi`で編集を試みよう (Viの復習)
- `i`を入力して挿入モードに切り替え
- 自分の学籍番号を入力して保存しよう  
([ESC]、`:wq`)



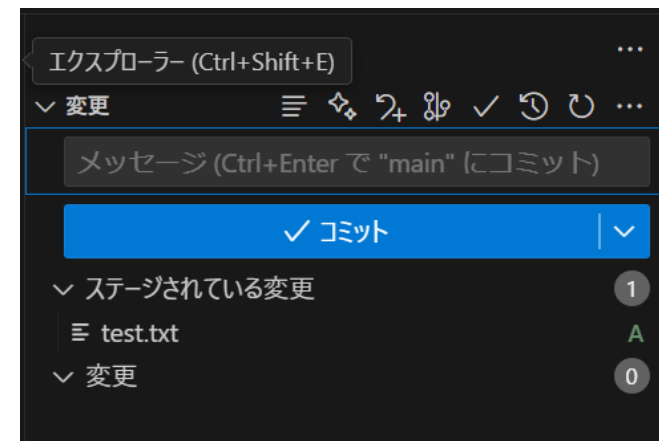
## ② ステージング

コマンドでステージングを試みよう

```
$ git add ファイル名
```

```
Admin@MSI MINGW64 /c/講義/第11回/lec11
$ git add test.txt
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it
```

サイドバーのソース管理をみると  
ステージされていることがわかる



### ③コミット

コミットメッセージを「test commit」としてコミットしてみよう

```
$ git commit -m “コミットメッセージ(変更内容など)”
```

```
Admin@MSI MINGW64 /c/講義/第11回/lec11(main)
• $ git commit -m "test commit"
[main 15ef42f] test commit
1 file changed, 1 insertion(+)
create mode 100644 test.txt
```

-mオプション：コミットメッセージをコマンドラインで直接指定するためのオプション

**※基本的にメッセージ無しでコミットすることはできない**



# ファイルの状態とコミットの状態を見てみよう

## ファイルの状態を確認する(status)

```
$ git status
```

```
Admin@MSI MINGW64 /c/講義/第11回/lec11 (main)
$ git status
On branch main
nothing to commit, working tree clean
```

## コミットの状態を確認する(log)

```
$ git log
```

```
Admin@MSI MINGW64 /c/講義/第11回/lec11 (main)
$ git log
commit 15ef42ffac8a09016cfc311cf80efb4effc6260c (HEAD -> main)
Author: takamotoak <takamotoak@edu.teu.ac.jp>
Date: Thu Jun 12 18:54:18 2025 +0900
```

ログの確認が終わったら  
「q」で終了する

commit の後に続く英数字が「コミットID」です

- コミットID部分を選択し、右クリックする(またはCtrl+Insert)とコピーできる
- コピーした文字列は、右クリックするとペーストできる



# コミットの取消し

誤ったコミットを取り消したいときもコマンドで行うことができる

- ソフトリセット：直前のコミットを取り消すが、編集対象のファイルやステージには影響を与えない

```
git reset --soft <コミットID>(ファイルの中身は書き換わらない)
```

- ハードリセット：コミット履歴も、編集対象のファイルやステージもすべてを指定した状態に戻す

```
git reset --hard <コミットID>(ファイルの中身も書き換わる)
```

# 練習

- コマンドでsoft reset と hard resetをやってみよう
- 1. test.txtに名前を追記してステージング(git add)を行いコミットメッセージを「test commit2」としてコミットせよ
- 2. test.txtに「東京工科大学」と追記・保存してステージング(git add)を行い「test commit3」としてコミット(git commit -m “test commit3”)せよ
- 3. git logを用いて今までのコミットIDを確認せよ
- 4. git reset --softで「test commit3」を取り消せ  
※test commit3を取り消すにはtest commit2のIDを用いる
- 5. git statusでコミットの状態を確認せよ  
※ (Changes to be committed:になっている)
- 6. git reset --hardで「test commit」の直後まで戻せ
- 7. test.txtの状態がもとに戻っていることを確認せよ



# Gitの基本コマンドのまとめ

コマンド	説明
git init	現在のディレクトリをGitリポジトリにする
git add ファイル名	ファイルをステージに追加
git commit -m "メッセージ"	ステージされた変更をコミット
git log	コミット履歴の確認
git reset --soft ログID	最新のコミットを取り消し、変更はステージに残す
git reset --hard ログID	コミット・ステージ・作業内容すべてを巻き戻す



# GitHubによるリモートリポジトリの利用

# GitHubとは？

- Gitで管理されたファイルをインターネット上にアップロードして扱うことができるWebサービス



アップロード



GitHub

ちなみにこのキャラはOctocat（オクトキャット）という公式マスコット

- GitHub上のリポジトリはリモートリポジトリと呼ばれる



# GitHubを使うメリット

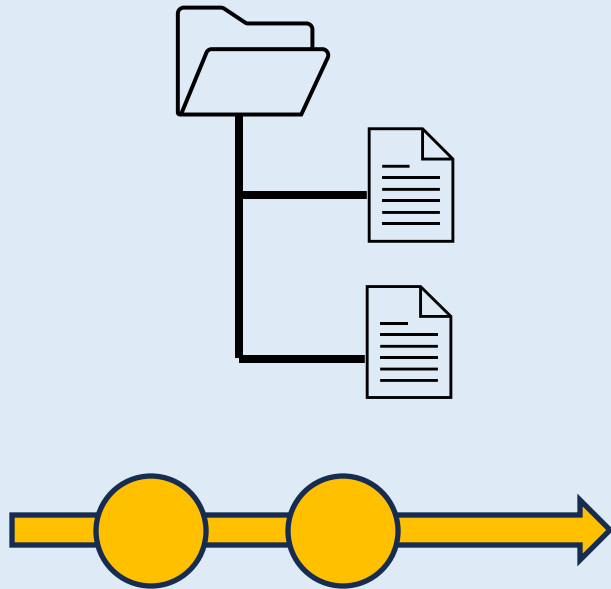
- チーム開発がスムーズになる  
→ 自分がGitで管理しているコードを、複数人で共同開発したいときに便利
- 変更履歴が明確に残る（バージョン管理）  
→ いつ・誰が・何をしたか記録される
- どこからでもアクセス可能  
→ インターネットがあれば作業できる
- バックアップとして機能  
→ ローカルが壊れても安心

# リモートリポジトリへのプッシュ

自分のPC

GitHub

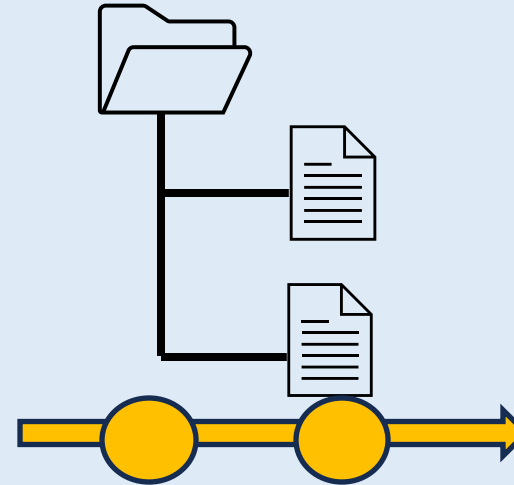
ローカルリポジトリ



コミットの履歴

Push  
(プッシュ)

リモートリポジトリ



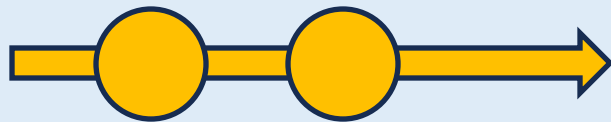
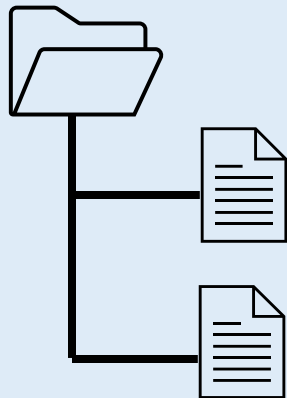
リモートリポジトリとリンクすることで、リモートのコミットなどをアップロードできる

# リモートリポジトリからのクローン

自分のPC

GitHub

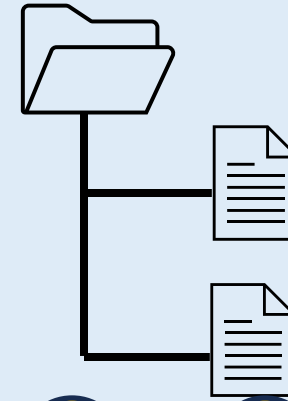
ローカルリポジトリ



コミットの履歴

Clone  
(クローン)

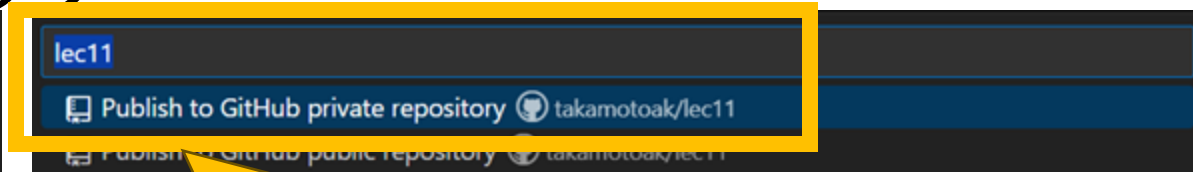
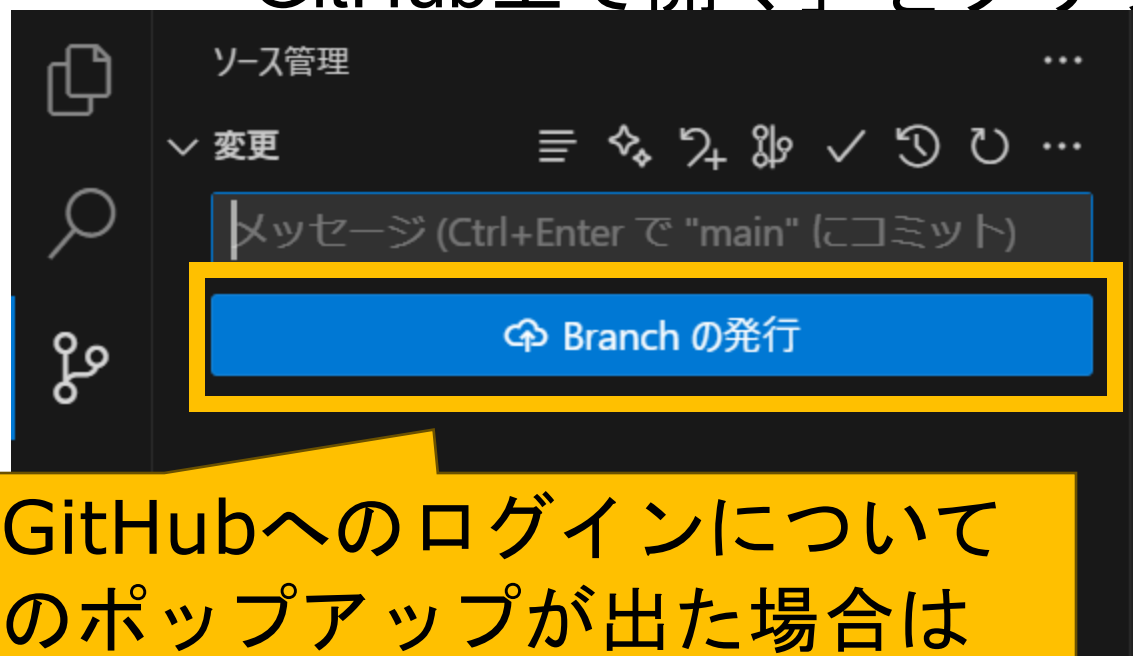
リモートリポジトリ



リモートリポジトリを自分のローカル環境にダウンロードすることもできる（ほかの人が作ったものもクローン出来る）

# VSCodeとGitHubの連携

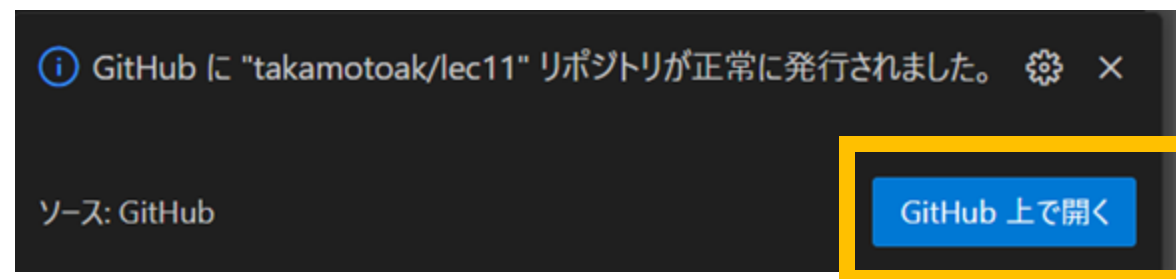
- lec11をGitHubにプッシュする
  - ローカルのリポジトリをリモートリポジトリへ
  - 「GitHub上で開く」をクリック



Publish to GitHub **private** repositoryを選択

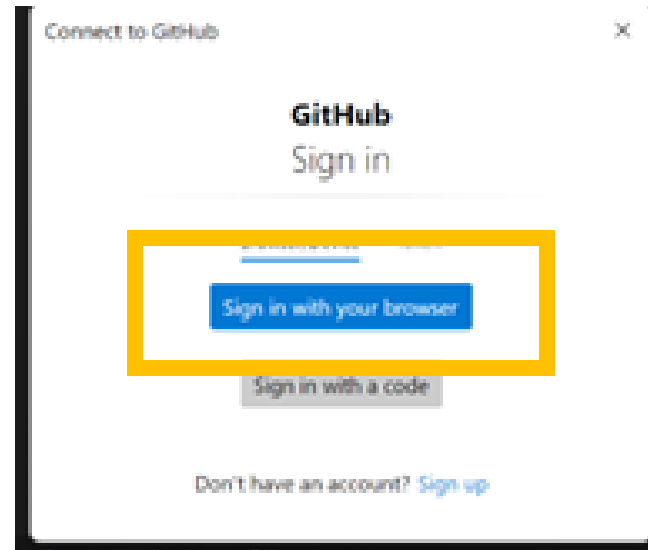
※別ウィンドウが開いた人は次ページ参照

GitHubへのログインについてのポップアップが出た場合は「OK」を選択して自分のIDを選択する



# GitHubへのサインイン

- Publish to をクリックしたあとにGitHubへのサインイン画面が出た人は「Sign in with your browser」をクリックしてGitHubへサインイン
- 「リポジトリが正常に発行されました」というメッセージが出ればOK



# ローカルリポジトリからリモートリポジトリ

- GitHubにlec11のリポジトリができているか確認



- ※GitHubのURL : <https://github.co.jp/>

# 参考：VS CodeのGitHubのマーク意味

アルファベット	単語	意味
A	added	新規追加
M	modified	変更あり
U	untracked	gitが未追跡(新規作成、add前)
D	deleted	削除済み
C	conflict	コンフリクト発生中
R	renamed	ファイル名変更済み
S	submodule	サブモジュール

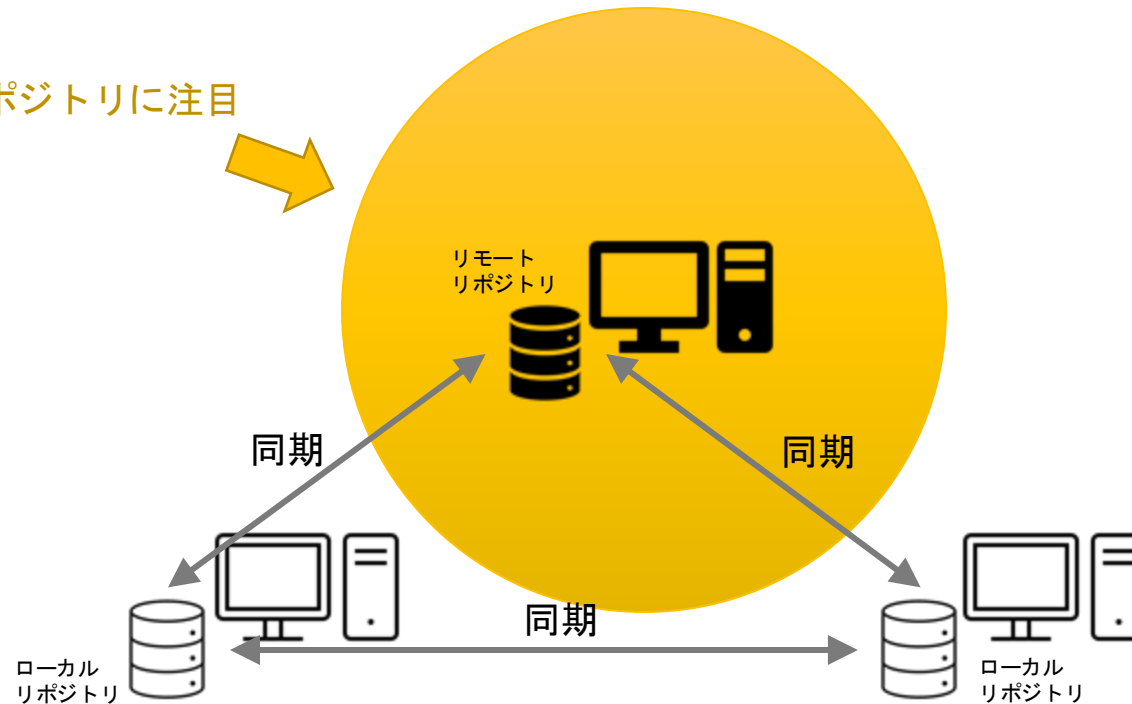
GitHubに作成したリモートリポジトリを  
ローカルにクローンする



# リモートリポジトリを作成してみよう

- GitHub上に、リポジトリを作成する
  - GitHubをブラウザから操作し、リポジトリを作成する

今はリモートリポジトリに注目



• リポジトリを新規作成

- 名前: **myhtml**
- 公開フラグ: **Private**
- READMEファイル: **なし**

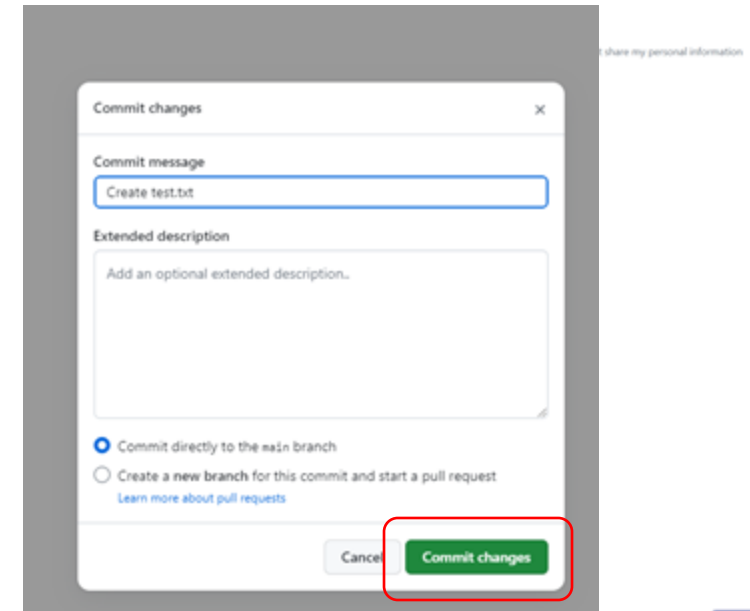
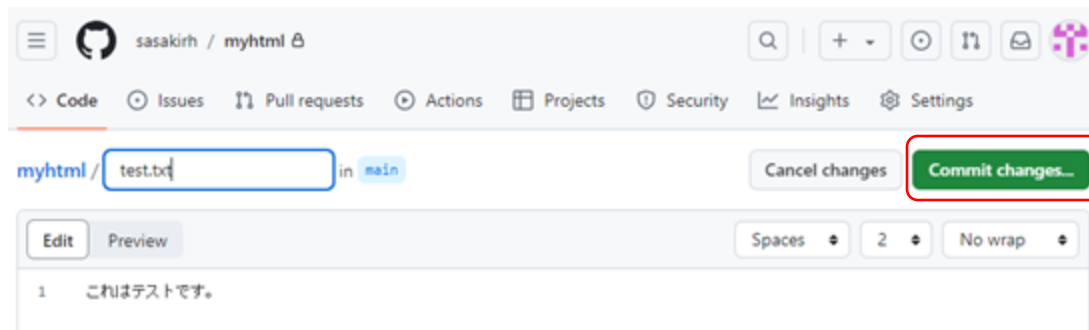
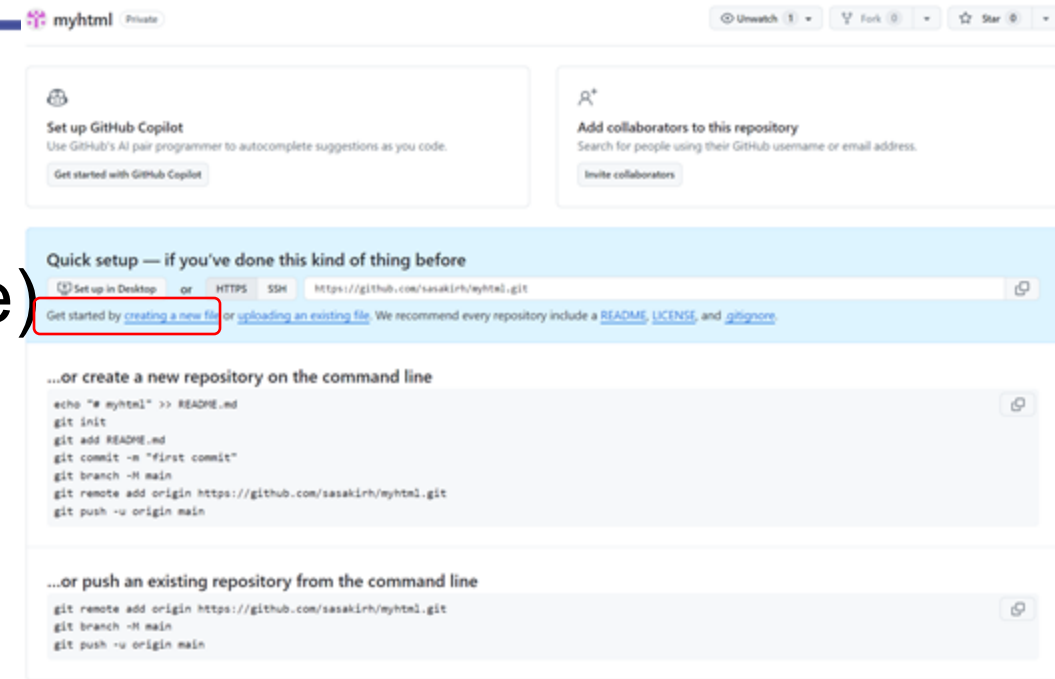
⇒ **Create repository** をクリック

# GitHub

- ファイルを新規作成(create a new file)

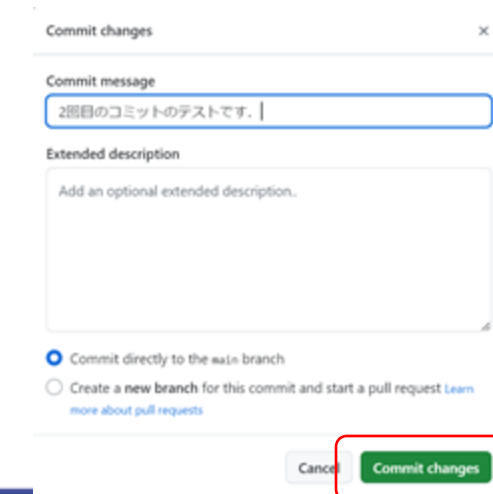
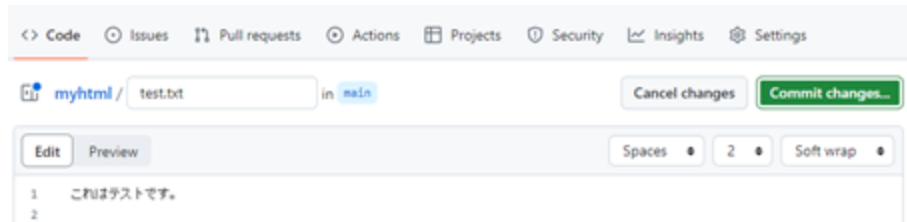
- ファイル名 : **test.txt**
- 1行目に以下を記入  
**これはテストです。**

**Commit changes...**を  
クリックし、コミットする



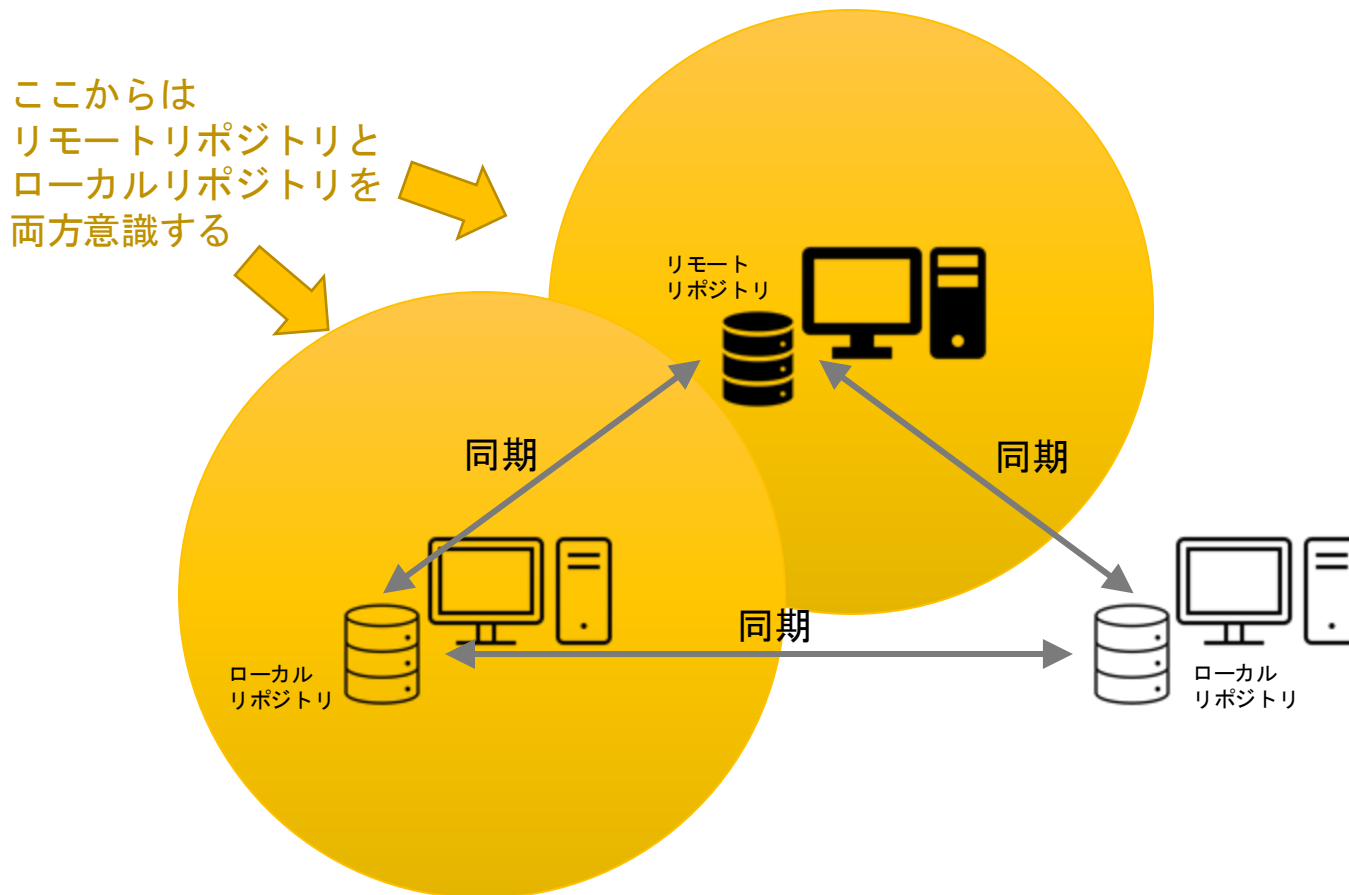
# GitHub

- test.txtをクリックしファイルを開き、次にEditボタンを押して編集モードにして、ファイルを更新し、commitする
  - 2行目に以下を追記する  
GitとGitHubを使用します。
  - Commit changesのボックスに、  
2回目のコミットのテストです。  
とコミットメッセージを入力



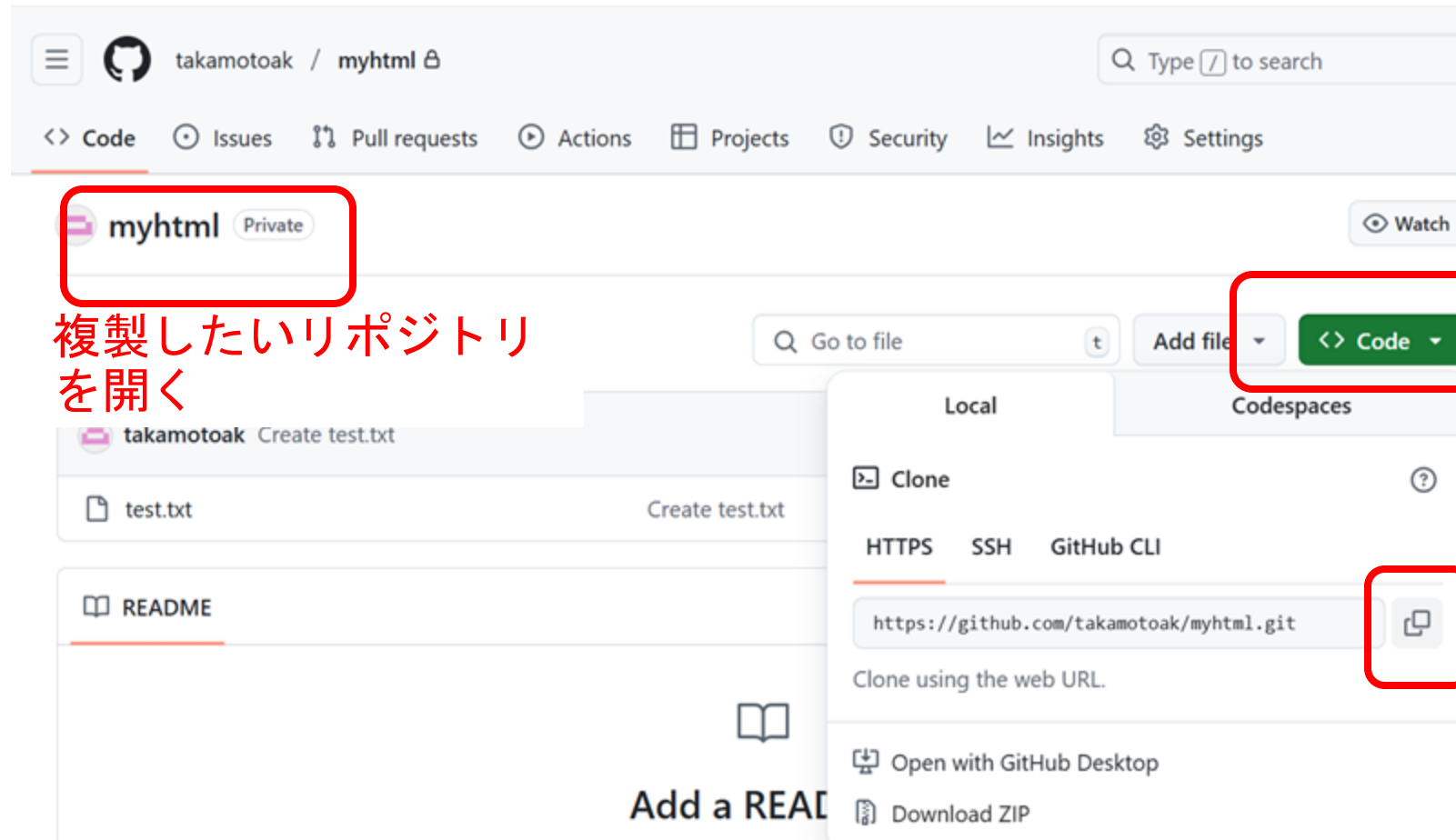
# ローカルリポジトリに複製（クローン）する

- GitHubにあるリモートリポジトリをローカルに持ってくる
  - 複製＝「クローン」する



# ローカルリポジトリにクローンする

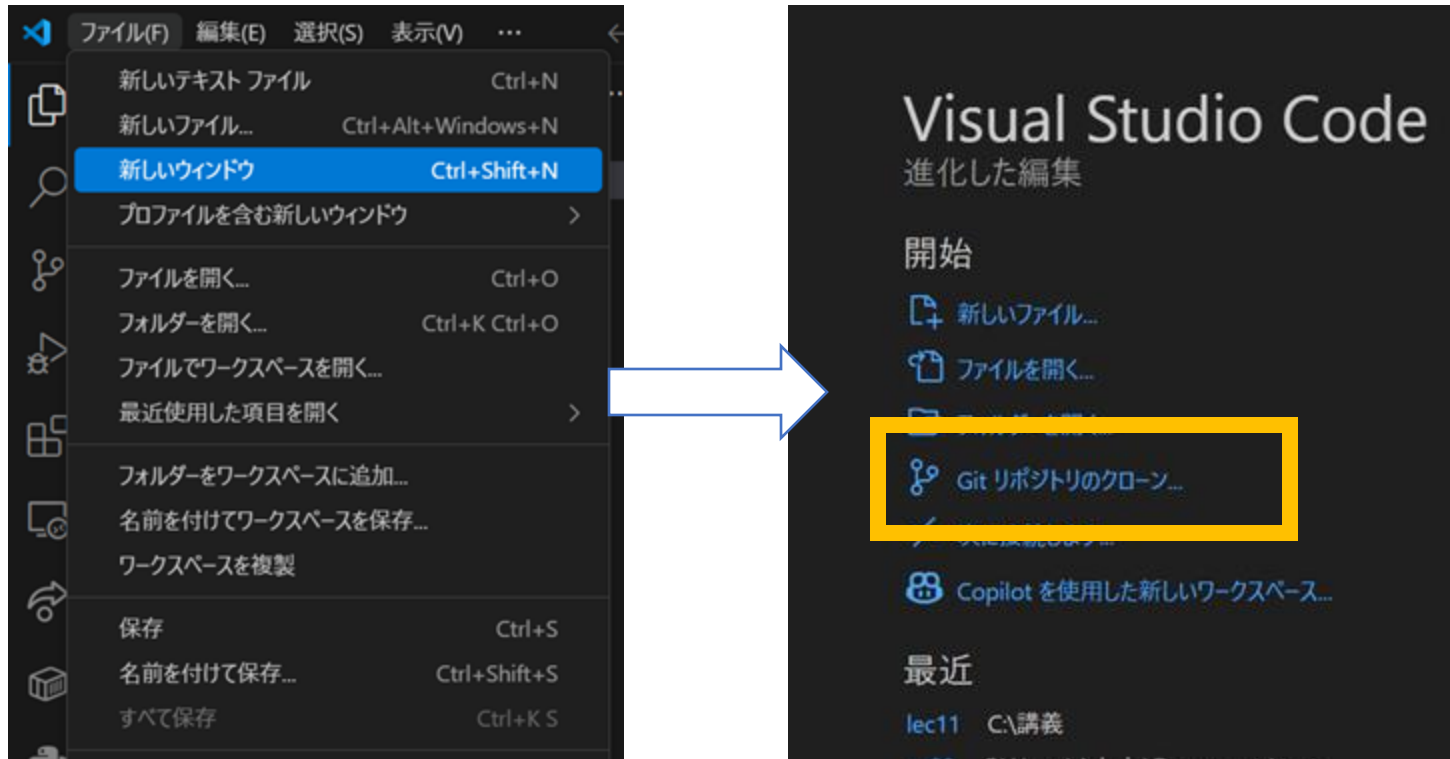
- ・クローン先リモートリポジトリのURLを張り付ける



リポジトリのURLをコピー

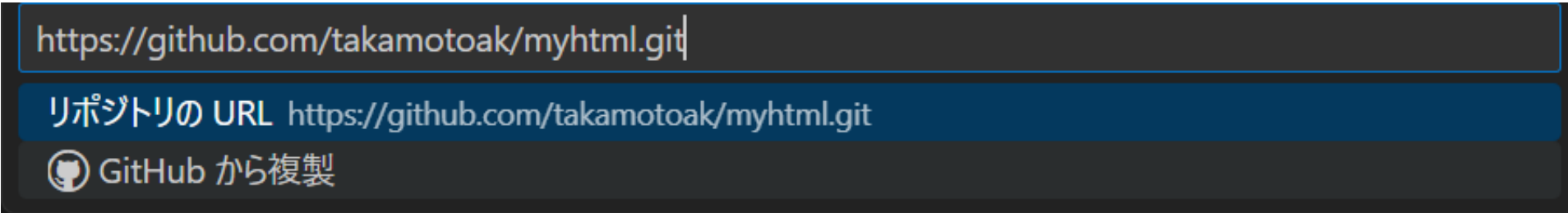
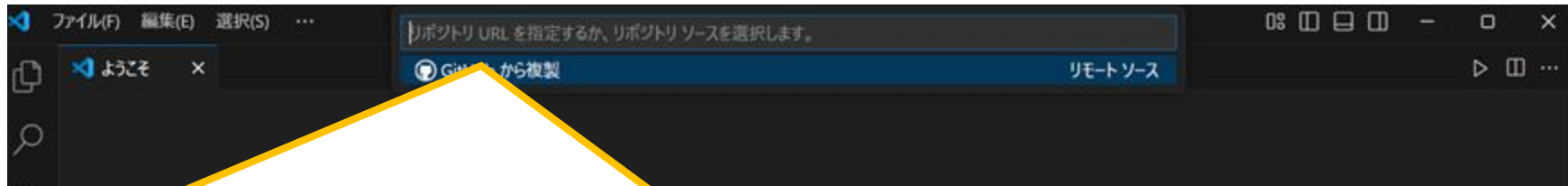
# ローカルリポジトリにクローンする

- VSCodeをGitHubと連携する
- 新しいウィンドウを開いて，Gitリポジトリのクローンを選択



# ローカルリポジトリにクローンする

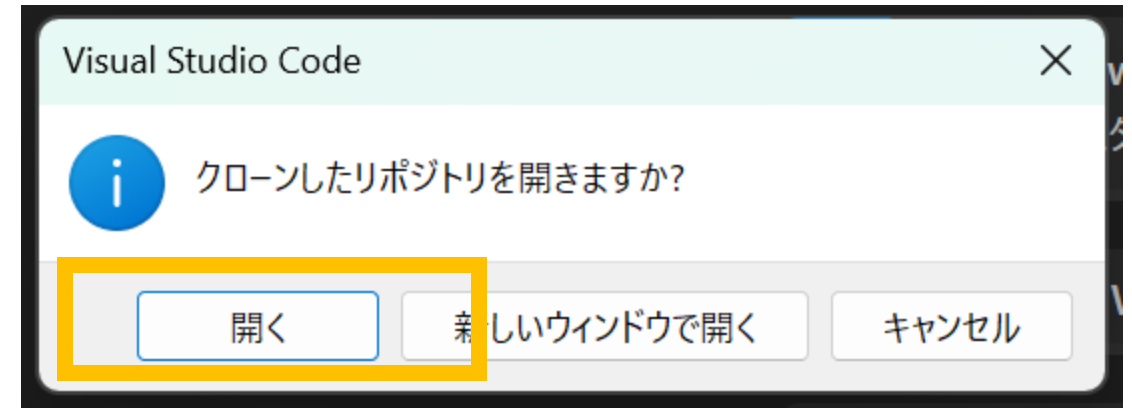
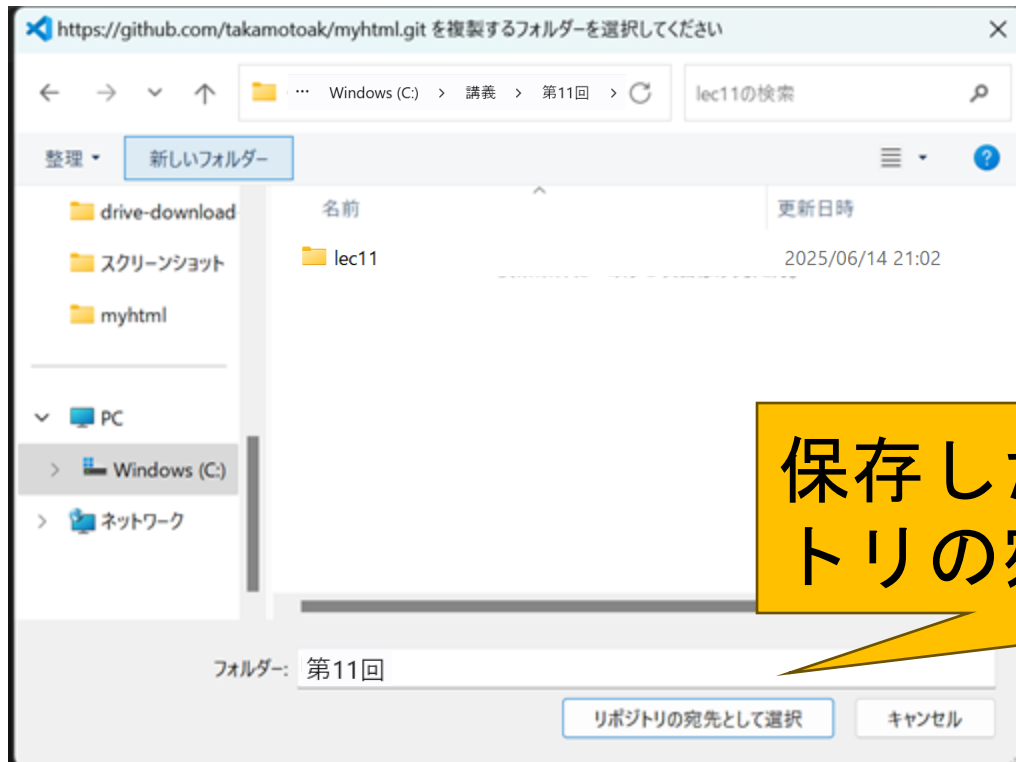
- 画面上部の欄にコピーしてきたリモートリポジトリの URL を張り付ける





# ローカルリポジトリにクローンする

- クローンしたリポジトリを保存する場所を選ぶ  
(**情報リテラシー演習/第11回**のフォルダの下に保存)

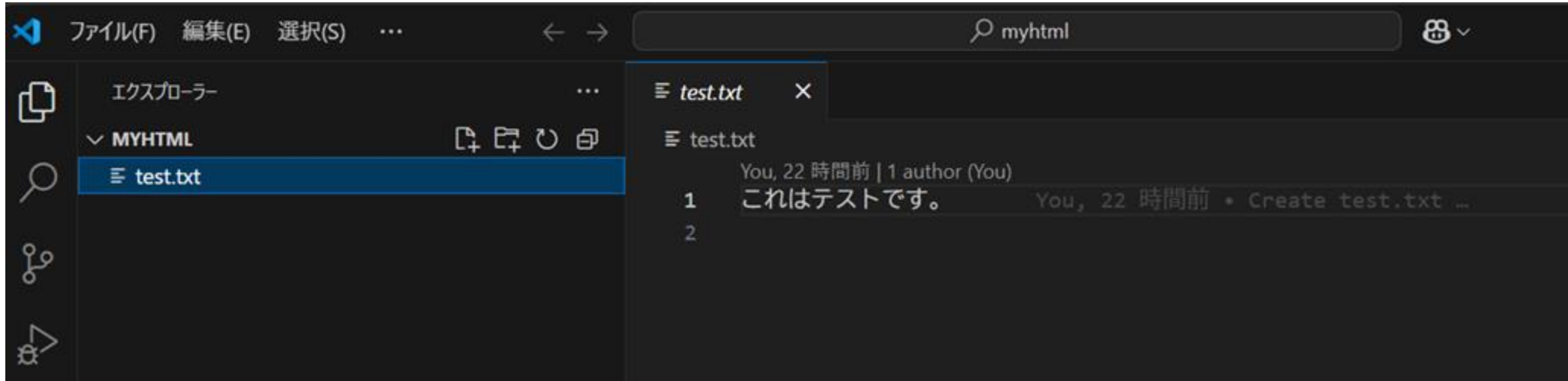


保存したい場所を選んだら「リポジトリの宛先として選択」をクリック



# ローカルリポジトリにクローンする

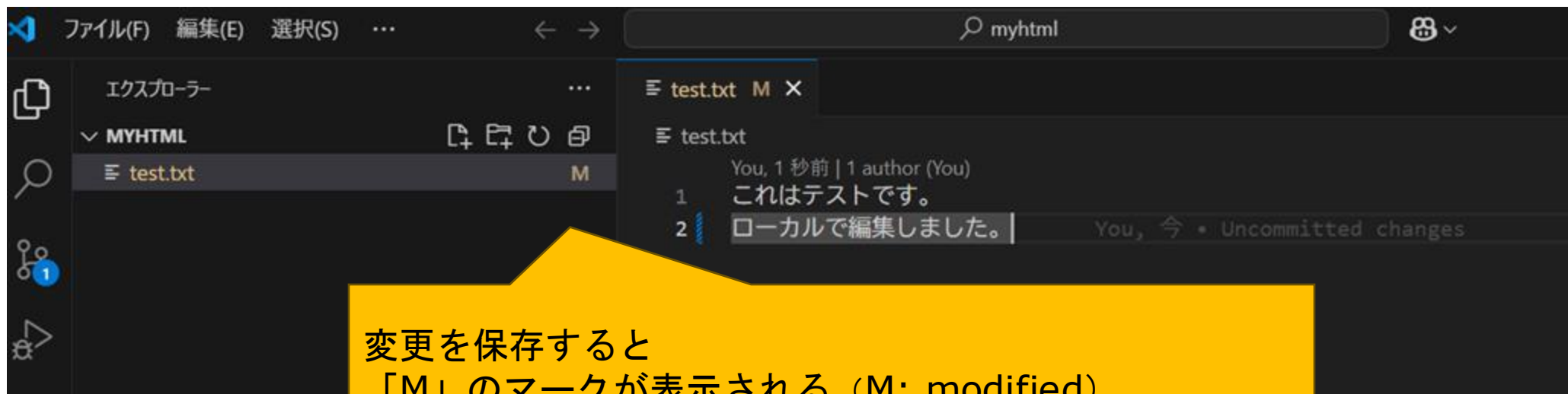
- ・リモートリポジトリで作成したデータがローカルで確認できる



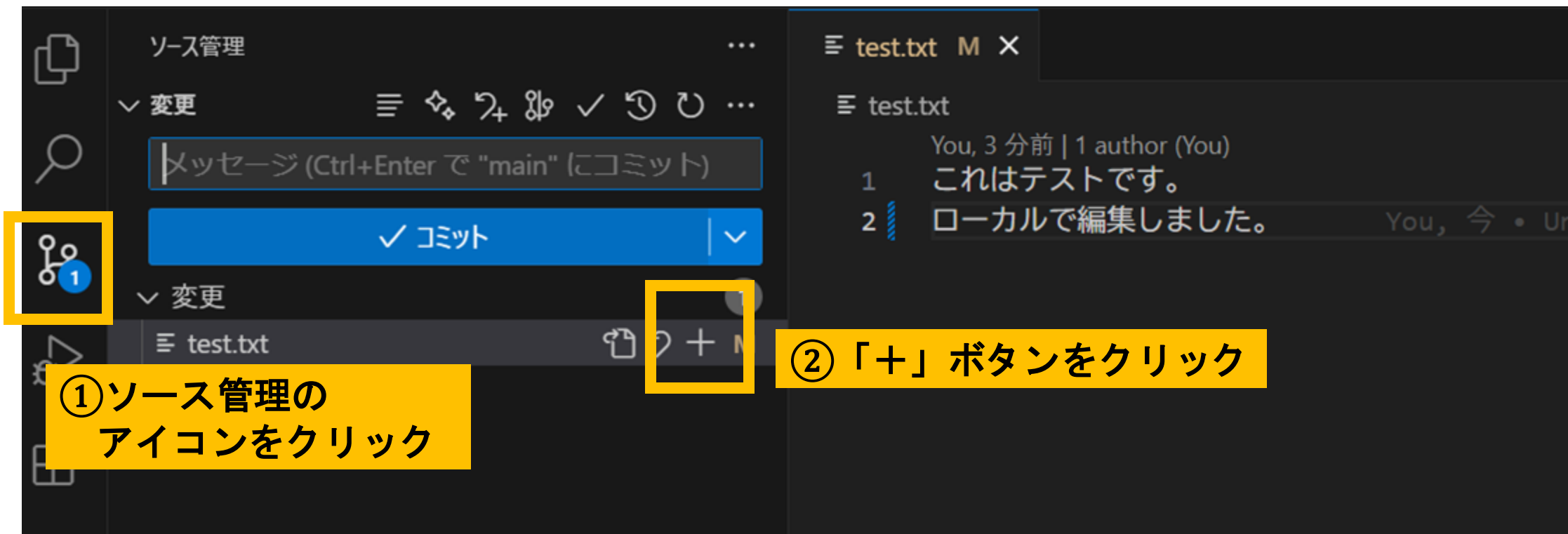
# 「test.txt」をローカルで編集

- クローンしたリポジトリにある「test.txt」を編集してみよう

「ローカルで編集しました。」と記入して保存



# ローカルでステージング



ソース管理

変更

メッセージ (Ctrl+Enter で "main" にコミット)

✓ コミット

変更

test.txt

test.txt M X

test.txt

You, 3 分前 | 1 author (You)

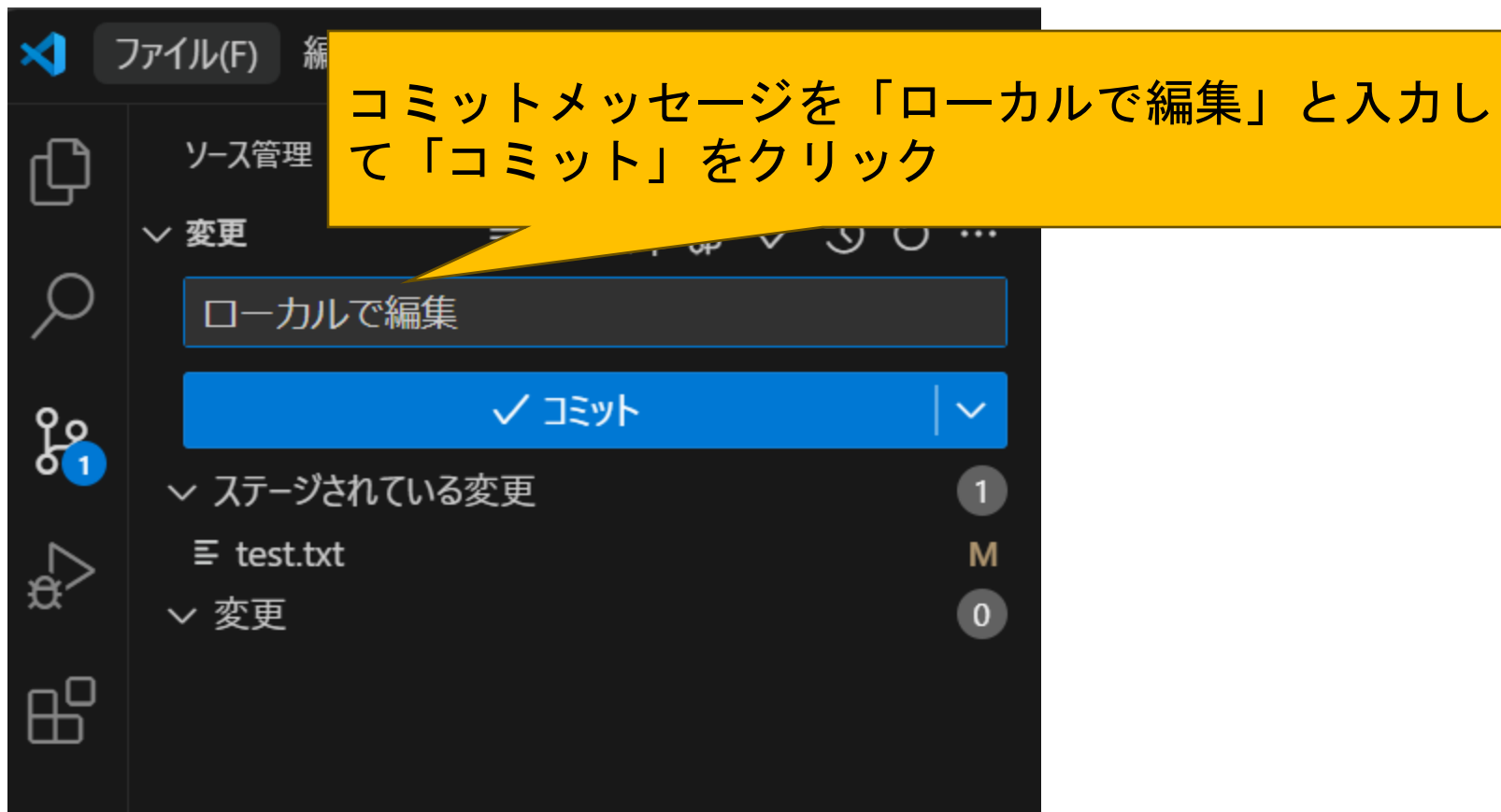
1 これはテストです。

2 ローカルで編集しました。 You, 今 • Un

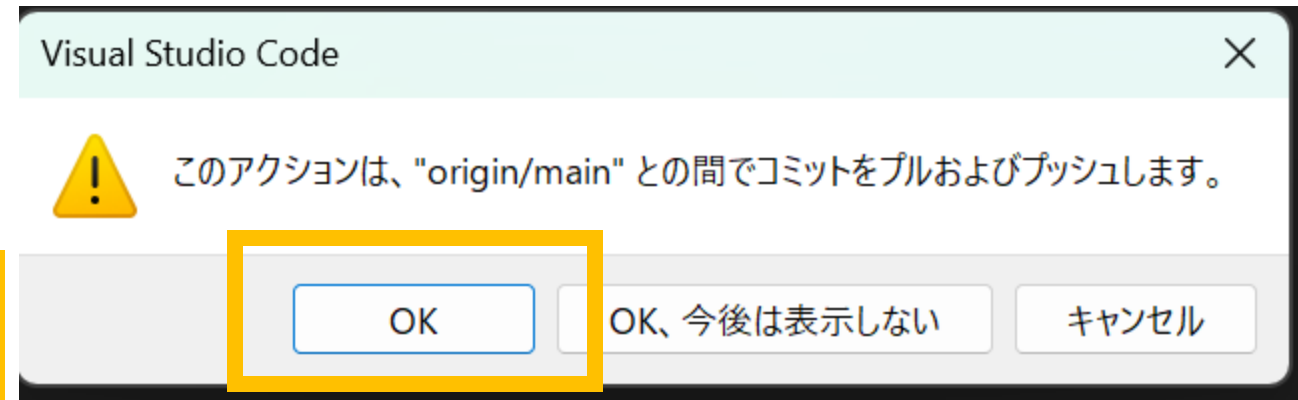
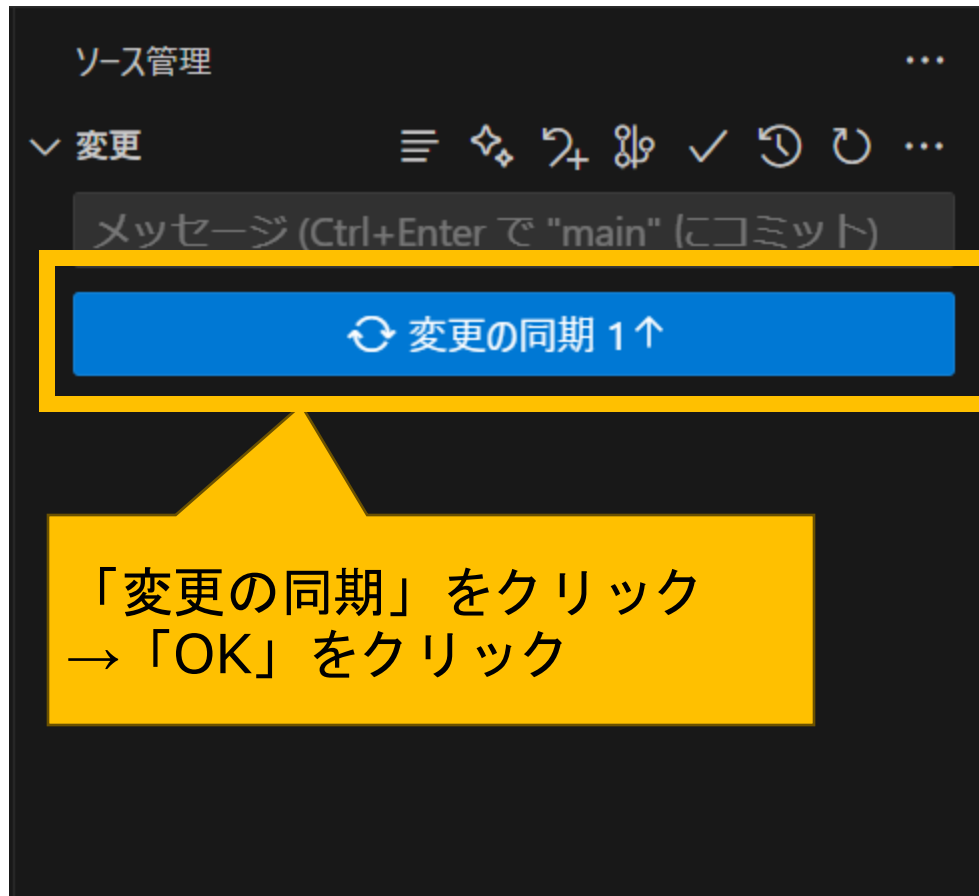
①ソース管理のアイコンをクリック

②「+」ボタンをクリック

# ローカルからリモートリポジトリへコミット



# GitHub（リモートリポジトリ）へ同期する



# 練習

- P.49～P.62をやってみよう
- 1. リモートリポジトリを作成せよ (myhtml)
- 2. リモートリポジトリにファイル (test.txt) を作成し、登録せよ
- 3. ファイルを更新し、コミットせよ
- 4. 自分のPCに、リモートリポジトリを複製せよ
- 5. 自分のPC上でファイルを更新し、自分のPCのローカルリポジトリにコミットせよ
- 6. ローカルリポジトリの変更をリモートリポジトリに反映せよ
- 7. GitHubを開いて更新し、コミットが反映されていることを確認せよ

