LAB 1 Елементи мови Python. Основні інструкції та структури даних. Робота з файловою системою

Нехай протягом навчання програмні проєкти з різних предметів зберігалися в папках ... . Є перелік папок `folders`, в яких зберігаються ці проєкти (задає виконавець, виходячи з вмісту своїх носіїв даних).
Примітка. До цього переліку мають входити не окремі папки з проєктами, а папки, що містять, наприклад, всі проєкти такої-то дисципліни такого-то семестру тощо. Інакше кажучи, не папка одного solution, а папка, в якій зберігаються solutions (якщо в термінології Visual Studio).

Щоб профільтрувати допоміжні файли середовищ, введемо "білий список" розширень (`WHITE_LIST`) та список розширень файлів з текстом програм(задається) (`CODE_LIST`).

З'ясувати

1. Скільки різних розширень зустрічається у папках з заданого списку (тут і далі: включаючи вкладені), скільки файлів з кожним розширенням. (Для цієї задачі розглядаємо всі розширення, а не тільки "білого списку".)
   Відповідну інформацію вивести на екран охайною табличкою. На рядку виводиться розширення, кількість файлів та ознака (C,W,O відповідно для файлів з текстом програм (відповідно до `CODE_LIST`), інших з білого списку, та тих, що не потрапили в білий список). Інформація подається за спаданням кількості файлів.
   Наприклад,
   ```
   .txt     155    W
   .ipynb    73    W
   .cpp      45    C
   .exe       3    O
   ```

2. За датами створення файлів для кожного місяця, починаючи з вересня 2024-го року, знайти таку інформацію: кількість файлів з розширеннями з "білого списку", їх сумарний обсяг у байтах, кількість їх рядків (враховувати тільки для файлів категорії C; білі та порожні рядки не рахувати), середній розмір файлу, середня кількість рядків.
   Відповідну інформацію вивести на екран охайною табличкою в хронологічному порядку.

   Наприклад,
   ```
   2024  09   23   123456  256   5368   11
   2024  10   23   123456  256   5368   11
   2024  11  134   123456  256    921   11
   ```

3. Для кожного місяця та кожного розширення, що зустрічаються при перегляді файлів, розміщених у папках зі списку, знайти таку інформацію: рік, місяць, розширення, кількість файлів, сумарний обсяг у байтах, кількість їх рядків (тільки для розширень категорії C; білі та порожні рядки не рахувати; для інших розширень цю клітину таблиці залишаємо незаповненою)

   Цю інформацію записати в csv-файл з іменем `lab1_1.csv` (кодування тільки utf-8), на початку додати рядок з заголовками: year, mon, ext, cnt, size, lines, avg_size, avg_line.

Здається:

1. Цей робочий зошит (заповнений, вилучати клітини, які були в ньому з самого початку, не дозволяється) та pdf-звіт.
   Перед здачею необхідно зробити Kernel -> Restart Kernel and Run All Cells... Для отримання pdf-звіта блокнот можна просто роздрукувати.
   Увага! Повідомлень про помилки виконання бути не повинно.
2. Файли зі скриптами для вирішення трьох задач (див. вище).
3. Файл `lab1_1.csv`

Оцінювання (максимум 15 балів):

Обов'язково має бути повністю та правильно виконаний (без повідомлень про помилки) та заповнений робочий зошит (блокнот), має бути pdf-звіт та `lab1_1.csv`, що відповідає виконаному блокноту.

За відсутності скриптів оцінка не може перевищувати 12 балів.

Правильність виконання оцінюється з точки зору дотримання вимог та отриманого кінцевого результату.

Якщо було вибрано папки майже без коду, то оцінка не може перевищувати 6 балів.

За неохайне виведення знімається 3 бали.

In [1]:
```
CODE_LIST = {'.cpp', '.hpp', '.c', '.h', '.py', '.pyw', '.cs', '.js', '.java'}
WHITE_LIST = CODE_LIST | {'.html', '.json', '.txt', '.ipynb'}
```

In [2]:
```
import sys
print(sys.platform, sys.version, sep='\n')
```

```
win32
3.14.2 (tags/v3.14.2:df79316, Dec  5 2025, 17:18:21) [MSC v.1944 64 bit (AMD64)]
```

Інструкції імпортування зазвичай записують на початку. Далі вони будуть додатково продубльовані по місцю використання.

In [3]:
```
import datetime
from pathlib import Path
```

```
from datetime import datetime as dt
import csv
```

Вихідні відомості

In [4]:
```
language = 'python3'
```

Бібліотека `datetime` містить стандартні засоби для зображення дат, зокрема клас datetime.date

In [5]:
```
import datetime
year = datetime.date.today().year
```

In [6]:
```
year
```

Out[6]:
```
2026
```

In [7]:
```
f'Ця робота використовує мову програмування {language} і була виконана у {year}'
```

Out[7]:
```
'Ця робота використовує мову програмування python3 і була виконана у 2026 році.'
```

In [8]:
```
author = "Бережна Ольга"
group = "К-27"
lab_instructor = "Карнаух Тетяна"
```

Запишіть f-рядок, значенням якого є текст (розташовується на двох рядках)

`Виконавець: ...., група ...`

`Викладач:`

Значення виконавця, його групи та викладача мають підставлятися зі змінних `author`, `group` та `lab_instructor`.

▶ Підказка.

In [9]:
```
info =f"""Виконавець: {author}, група {group}
Викладач: {lab_instructor}"""
print(info)
```

```
Виконавець: Бережна Ольга, група К-27
Викладач: Карнаух Тетяна
```

Готуємось "гуляти" по папках (підготовка-дослідження)

Нам знадобиться клас `Path` з бібліотеки `pathlib`. Документація.

In [10]:
```
from pathlib import Path
```

Дізнаємось поточний робочий каталог. (**c**urrent **w**ork **d**ir)

In [11]:
```
Path().cwd()
```

Out[11]:
```
WindowsPath('C:/Users/User/python')
```

In [12]:
```
str(Path().cwd())
```

Out[12]:
```
'C:\\Users\\User\\python'
```

Шлях до якої-небудь папки з помірною кількістю файлів з різними розширеннями та складеними папками зробити значенням змінної `root`.
У кожного має бути свій шлях.

Наприклад, можна взяти папку з проєктом мови C++.

In [13]:
```
root = Path('/Users/User/source/repos/lab2')
root
```

Out[13]:
```
WindowsPath('/Users/User/source/repos/lab2')
```

Обійти папку, вивести всі наявні в ній файли

Цікавить шлях, тип файлу (каталог чи звичайний файл), для звичайного файлу цікавить розширення та розмір у байтах.

Сімлінки та інше будемо пропускати.

▶ Підказка 1

In [14]:
```
help(Path.glob)
```

```
Help on function glob in module pathlib:

glob(self, pattern, *, case_sensitive=None, recurse_symlinks=False)
    Iterate over this subtree and yield all existing files (of any
    kind, including directories) matching the given relative pattern.
```

▶ Підказка 2

▶ Підказка 3

Обійти всі наявні в `root` файли та вивести їх імена

In [15]:
```
for el in root.glob('**/*'):
      print(el)
```

```
\Users\User\source\repos\lab2\.vs
\Users\User\source\repos\lab2\CMakeLists.txt
\Users\User\source\repos\lab2\CMakePresets.json
\Users\User\source\repos\lab2\lab2
\Users\User\source\repos\lab2\out
\Users\User\source\repos\lab2\.vs\CMake Overview
\Users\User\source\repos\lab2\.vs\lab2
\Users\User\source\repos\lab2\.vs\ProjectSettings.json
\Users\User\source\repos\lab2\.vs\slnx.sqlite
\Users\User\source\repos\lab2\.vs\VSWorkspaceState.json
\Users\User\source\repos\lab2\lab2\CMakeLists.txt
\Users\User\source\repos\lab2\lab2\lab2.cpp
\Users\User\source\repos\lab2\lab2\lab2.h
\Users\User\source\repos\lab2\out\build
\Users\User\source\repos\lab2\out\build\x64-debug
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake
\Users\User\source\repos\lab2\out\build\x64-debug\.ninja_deps
\Users\User\source\repos\lab2\out\build\x64-debug\.ninja_log
\Users\User\source\repos\lab2\out\build\x64-debug\build.ninja
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeCache.txt
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles
\Users\User\source\repos\lab2\out\build\x64-debug\cmake_install.cmake
\Users\User\source\repos\lab2\out\build\x64-debug\lab2
\Users\User\source\repos\lab2\out\build\x64-debug\Testing
\Users\User\source\repos\lab2\out\build\x64-debug\VSInheritEnvironments.txt
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\cmake.check_cache
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\CMakeConfigureLog.ya
ml
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\pkgRedirects
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\rules.ninja
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\ShowIncludes
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\TargetDirectories.tx
t
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\cmake_install.cmake
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\lab2.exe
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\lab2.ilk
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\lab2.pdb
\Users\User\source\repos\lab2\out\build\x64-debug\Testing\Temporary
\Users\User\source\repos\lab2\out\build\x64-debug\Testing\Temporary\LastTest.log
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir\embed.
manifest
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir\interm
ediate.manifest
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir\lab2.c
pp.obj
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir\lab2.c
pp.obj.enc
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir\manife
st.rc
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir\manife
st.res
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir\vc140.
idb
\Users\User\source\repos\lab2\out\build\x64-debug\lab2\CMakeFiles\lab2.dir\vc140.
pdb
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeCC
```

```
ompiler.cmake
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeCX
XCompiler.cmake
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeDe
termineCompilerABI_C.bin
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeDe
termineCompilerABI_CXX.bin
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeRC
Compiler.cmake
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeSy
stem.cmake
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdC
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdCXX
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\ShowIncludes\foo.h
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\ShowIncludes\main.c
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\ShowIncludes\main.ob
j
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdC\CMakeCCompilerId.c
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdC\CMakeCCompilerId.exe
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdC\CMakeCCompilerId.obj
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdC\tmp
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdCXX\CMakeCXXCompilerId.cpp
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdCXX\CMakeCXXCompilerId.exe
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdCXX\CMakeCXXCompilerId.obj
\Users\User\source\repos\lab2\out\build\x64-debug\CMakeFiles\3.29.5-msvc4\Compile
rIdCXX\tmp
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\query
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\query\client-Micr
osoftVS
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply\cache-v2-b0
a87ca0b91c5463b806.json
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply\cmakeFiles-
v1-15c1d6e776bc169829bc.json
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply\codemodel-v
2-42ba6b42696e52a62d8f.json
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply\directory-.
-Debug-d0094a50bb2071803777.json
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply\directory-l
ab2-Debug-a34de045653a0b808c91.json
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply\index-2024-
11-09T12-28-30-0324.json
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply\target-lab2
-Debug-aabaa3c6fb531ff36ade.json
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\reply\toolchains-
v1-d19162570acee6f721be.json
\Users\User\source\repos\lab2\out\build\x64-debug\.cmake\api\v1\query\client-Micr
osoftVS\query.json
\Users\User\source\repos\lab2\.vs\lab2\FileContentIndex
\Users\User\source\repos\lab2\.vs\lab2\v17
```

```
\Users\User\source\repos\lab2\.vs\lab2\FileContentIndex\7fcd3efe-1a6b-4d3d-ba1f-a
db3755c5c95.vsidx
\Users\User\source\repos\lab2\.vs\lab2\v17\.wsuo
\Users\User\source\repos\lab2\.vs\lab2\v17\Browse.VC.db
\Users\User\source\repos\lab2\.vs\lab2\v17\DocumentLayout.backup.json
\Users\User\source\repos\lab2\.vs\lab2\v17\DocumentLayout.json
\Users\User\source\repos\lab2\.vs\lab2\v17\ipch
\Users\User\source\repos\lab2\.vs\lab2\v17\ipch\AutoPCH
\Users\User\source\repos\lab2\.vs\lab2\v17\ipch\AutoPCH\846260f32befc30c
\Users\User\source\repos\lab2\.vs\lab2\v17\ipch\AutoPCH\846260f32befc30c\LAB2.ipc
h
```

Є бажання виводити їх шляхи відносно заданої папки.

Тобто необхідно побудувати шлях, відносно шляху `root`. Допоможе метод `relative_to`.

In [16]:
```
help(Path.relative_to)
```

```
Help on function relative_to in module pathlib:

relative_to(self, other, *, walk_up=False)
    Return the relative path to another path identified by the passed
    arguments.  If the operation is not possible (because this is not
    related to the other path), raise ValueError.

    The *walk_up* parameter controls whether `..` may be used to resolve
    the path.
```

Примітка. Насправді це метод класу `PurePath`, який є базовим для `Path`.

Після завершення циклу змінна `el`, яка позначала поточний елемент обходу, продовжує існувати. Проведемо з нею кілька експериментів.

Виведемо її значення.

In [17]:
```
el
```

Out[17]:
```
WindowsPath('/Users/User/source/repos/lab2/.vs/lab2/v17/ipch/AutoPCH/846260f32b
efc30c/LAB2.ipch')
```

У наступній клітині побудуємо та виведемо відповідний відносний шлях.

▶ Підказка

In [18]:
```
el.relative_to(root)
```

Out[18]:
```
WindowsPath('.vs/lab2/v17/ipch/AutoPCH/846260f32befc30c/LAB2.ipch')
```

Явно перетворимо відносний шлях на рядок (значення типу `str`).

▶ Підказка

▶ Відповідь

In [19]:
```
str(el.relative_to(root))
```

Out[19]:
```
'.vs\\lab2\\v17\\ipch\\AutoPCH\\846260f32befc30c\\LAB2.ipch'
```

Тепер у нашому обході будемо виводити шляхи, відносно шляху `root`. У наступну клітину слід записати модифікацію обходу, який здійснювався раніше.

Примітка. Якщо ми збираємось відносний шлях тільки виводити, то перед виведенням перетворювати його явно на рядок потреби немає. Це чудово зробить сама функція `print`.

▶ Підказка

In [20]:
```
for el in root.glob('**/*'):
      print(el.relative_to(root))
```

```
.vs
CMakeLists.txt
CMakePresets.json
lab2
out
.vs\CMake Overview
.vs\lab2
.vs\ProjectSettings.json
.vs\slnx.sqlite
.vs\VSWorkspaceState.json
lab2\CMakeLists.txt
lab2\lab2.cpp
lab2\lab2.h
out\build
out\build\x64-debug
out\build\x64-debug\.cmake
out\build\x64-debug\.ninja_deps
out\build\x64-debug\.ninja_log
out\build\x64-debug\build.ninja
out\build\x64-debug\CMakeCache.txt
out\build\x64-debug\CMakeFiles
out\build\x64-debug\cmake_install.cmake
out\build\x64-debug\lab2
out\build\x64-debug\Testing
out\build\x64-debug\VSInheritEnvironments.txt
out\build\x64-debug\.cmake\api
out\build\x64-debug\CMakeFiles\3.29.5-msvc4
out\build\x64-debug\CMakeFiles\cmake.check_cache
out\build\x64-debug\CMakeFiles\CMakeConfigureLog.yaml
out\build\x64-debug\CMakeFiles\pkgRedirects
out\build\x64-debug\CMakeFiles\rules.ninja
out\build\x64-debug\CMakeFiles\ShowIncludes
out\build\x64-debug\CMakeFiles\TargetDirectories.txt
out\build\x64-debug\lab2\CMakeFiles
out\build\x64-debug\lab2\cmake_install.cmake
out\build\x64-debug\lab2\lab2.exe
out\build\x64-debug\lab2\lab2.ilk
out\build\x64-debug\lab2\lab2.pdb
out\build\x64-debug\Testing\Temporary
out\build\x64-debug\Testing\Temporary\LastTest.log
out\build\x64-debug\lab2\CMakeFiles\lab2.dir
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\embed.manifest
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\intermediate.manifest
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\lab2.cpp.obj
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\lab2.cpp.obj.enc
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\manifest.rc
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\manifest.res
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\vc140.idb
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\vc140.pdb
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeCCompiler.cmake
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeCXXCompiler.cmake
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeDetermineCompilerABI_C.bin
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeDetermineCompilerABI_CXX.bin
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeRCCompiler.cmake
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeSystem.cmake
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX
out\build\x64-debug\CMakeFiles\ShowIncludes\foo.h
out\build\x64-debug\CMakeFiles\ShowIncludes\main.c
out\build\x64-debug\CMakeFiles\ShowIncludes\main.obj
```

```
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC\CMakeCCompilerId.c
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC\CMakeCCompilerId.exe
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC\CMakeCCompilerId.obj
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC\tmp
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX\CMakeCXXCompilerId.cpp
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX\CMakeCXXCompilerId.exe
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX\CMakeCXXCompilerId.obj
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX\tmp
out\build\x64-debug\.cmake\api\v1
out\build\x64-debug\.cmake\api\v1\query
out\build\x64-debug\.cmake\api\v1\reply
out\build\x64-debug\.cmake\api\v1\query\client-MicrosoftVS
out\build\x64-debug\.cmake\api\v1\reply\cache-v2-b0a87ca0b91c5463b806.json
out\build\x64-debug\.cmake\api\v1\reply\cmakeFiles-v1-15c1d6e776bc169829bc.json
out\build\x64-debug\.cmake\api\v1\reply\codemodel-v2-42ba6b42696e52a62d8f.json
out\build\x64-debug\.cmake\api\v1\reply\directory-.-Debug-d0094a50bb2071803777.js
on
out\build\x64-debug\.cmake\api\v1\reply\directory-lab2-Debug-a34de045653a0b808c9
1.json
out\build\x64-debug\.cmake\api\v1\reply\index-2024-11-09T12-28-30-0324.json
out\build\x64-debug\.cmake\api\v1\reply\target-lab2-Debug-aabaa3c6fb531ff36ade.js
on
out\build\x64-debug\.cmake\api\v1\reply\toolchains-v1-d19162570acee6f721be.json
out\build\x64-debug\.cmake\api\v1\query\client-MicrosoftVS\query.json
.vs\lab2\FileContentIndex
.vs\lab2\v17
.vs\lab2\FileContentIndex\7fcd3efe-1a6b-4d3d-ba1f-adb3755c5c95.vsidx
.vs\lab2\v17\.wsuo
.vs\lab2\v17\Browse.VC.db
.vs\lab2\v17\DocumentLayout.backup.json
.vs\lab2\v17\DocumentLayout.json
.vs\lab2\v17\ipch
.vs\lab2\v17\ipch\AutoPCH
.vs\lab2\v17\ipch\AutoPCH\846260f32befc30c
.vs\lab2\v17\ipch\AutoPCH\846260f32befc30c\LAB2.ipch
```

Складові шляху

Інформацію про частини шляху можна знайти тут.

Альтернатива: використати довідку.

Знайдемо всі атрибути, доступні для класу `Path`. (У Python під атрибутами розуміють і класичні атрибути (описують дані об'єкта), і методи (це наслідок технічної реалізації методів).)

Функція `dir` поверне список.

In [21]:
```
dir(Path)
```

Out[21]:
```
['__bytes__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__firstlineno__',
 '__format__',
 '__fspath__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rtruediv__',
 '__setattr__',
 '__sizeof__',
 '__slots__',
 '__static_attributes__',
 '__str__',
 '__subclasshook__',
 '__truediv__',
 '_copy_from',
 '_copy_from_file',
 '_copy_from_file_fallback',
 '_copy_from_symlink',
 '_delete',
 '_drv',
 '_filter_trailing_slash',
 '_format_parsed_parts',
 '_from_dir_entry',
 '_from_parsed_parts',
 '_from_parsed_string',
 '_hash',
 '_info',
 '_parse_path',
 '_parse_pattern',
 '_parts_normcase',
 '_parts_normcase_cached',
 '_raw_path',
 '_raw_paths',
 '_remove_leading_dot',
 '_remove_trailing_slash',
 '_root',
 '_str',
 '_str_normcase',
 '_str_normcase_cached',
 '_tail',
 '_tail_cached',
 'absolute',
```

```
'anchor',
'as_posix',
'as_uri',
'chmod',
'copy',
'copy_into',
'cwd',
'drive',
'exists',
'expanduser',
'from_uri',
'full_match',
'glob',
'group',
'hardlink_to',
'home',
'info',
'is_absolute',
'is_block_device',
'is_char_device',
'is_dir',
'is_fifo',
'is_file',
'is_junction',
'is_mount',
'is_relative_to',
'is_reserved',
'is_socket',
'is_symlink',
'iterdir',
'joinpath',
'lchmod',
'lstat',
'match',
'mkdir',
'move',
'move_into',
'name',
'open',
'owner',
'parent',
'parents',
'parser',
'parts',
'read_bytes',
'read_text',
'readlink',
'relative_to',
'rename',
'replace',
'resolve',
'rglob',
'rmdir',
'root',
'samefile',
'stat',
'stem',
'suffix',
'suffixes',
'symlink_to',
```

```
 'touch',
 'unlink',
 'walk',
 'with_name',
 'with_segments',
 'with_stem',
 'with_suffix',
 'write_bytes',
 'write_text']
```

Серед них ті, що починаються з двох підкреслень, є службовими. Ті, що починаються з одного підкреслення, — допоміжними (не є відкритими). Виведемо всі, крім них.

In [22]:
```
cnt = 0
for a in dir(Path):
    if not a.startswith('_'):
        print(a, end=', ')
        cnt += 1
        if (cnt > 5):
            cnt = 0
            print()
```

```
absolute, anchor, as_posix, as_uri, chmod, copy,
copy_into, cwd, drive, exists, expanduser, from_uri,
full_match, glob, group, hardlink_to, home, info,
is_absolute, is_block_device, is_char_device, is_dir, is_fifo, is_file,
is_junction, is_mount, is_relative_to, is_reserved, is_socket, is_symlink,
iterdir, joinpath, lchmod, lstat, match, mkdir,
move, move_into, name, open, owner, parent,
parents, parser, parts, read_bytes, read_text, readlink,
relative_to, rename, replace, resolve, rglob, rmdir,
root, samefile, stat, stem, suffix, suffixes,
symlink_to, touch, unlink, walk, with_name, with_segments,
with_stem, with_suffix, write_bytes, write_text,
```

Повертаємось до нашого примірника шляху.

In [23]:
```
el
```

Out[23]:
```
WindowsPath('/Users/User/source/repos/lab2/.vs/lab2/v17/ipch/AutoPCH/846260f32b
efc30c/LAB2.ipch')
```

Визначимо, чи задає цей шлях папку, файл, яке в нього ім'я, розширення, який його розмір.

▶ Підказка

Записати вираз, що перевіряє, чи задає шлях `el` папку.

In [24]:
```
el.is_dir()
```

Out[24]:
```
False
```

Записати вираз, що перевіряє, чи задає шлях `el` файл.

In [25]:
```
el.is_file()
```

Out[25]:
```
True
```

Записати вираз, що визначає ім'я файлу `el`.

In [26]:
```
el.name
```

Out[26]:
```
'LAB2.ipch'
```

Записати вираз, що визначає розширення файлу `el`.

In [27]:
```
el.suffix
```

Out[27]:
```
'.ipch'
```

Записати вираз, що визначає розмір файлу `el`.

In [28]:
```
el.stat().st_size
```

Out[28]:
```
69074944
```

Для розв'язання загальної задачі нам буде необхідно вміти визначати рік та місяць створення файла.

Не тільки розмір файлу, але й інша інформація про файл може бути знайдена за допомогою метода `stat` (документація)

Нас цікавить дата створення файла. Тут виникає невелика залежність від ОС. (документація)

Знайдемо час створення файлу (у секундах) і запам'ятаємо в змінній `ctime`.

In [29]:
```
if sys.platform.startswith('win'):
    ctime = el.stat().st_birthtime
else:
    ctime = el.stat().st_ctime
ctime
```

Out[29]:
```
1730733691.6317875
```

Тепер слід перетворити на звичайну дату, щоб з'ясувати рік та місяць.

Використаємо клас `datetime`, точніше його метод `fromtimestamp`. (документація)

Нижче клас `datetime` імпортований з одноіменної бібліотеки `datetime` під псевдонімом `dt`.

In [30]:
```
from datetime import datetime as dt
```

In [31]:
```
cdate = dt.fromtimestamp(ctime)
cdate
```

Out[31]:
```
datetime.datetime(2024, 11, 4, 17, 21, 31, 631788)
```

In [32]:
```
cdate.year, cdate.month
```

Out[32]:
```
(2024, 11)
```

Напишіть **кросплатформену** функцію `created_at`, що за шляхом до файлу визначає рік та місяць його створення. Функція має отримувати шлях до файлу та повертати пару цілих чисел.

Приклад функції, що повертає два значення.
`def f(arg):`
`return arg, arg+arg`

Питання для обговорення. В який момент краще розгалужити обчислення дати створення?

▶ Відповідь

In [33]:
```
def _created_at_Linux(path):     # треба записати власні реалізації цих двох фун
    p = Path(path)
    stat = p.stat()
    timestamp = min(stat.st_ctime, stat.st_mtime)
    dt_obj = dt.fromtimestamp(timestamp)
    return dt_obj.year, dt_obj.month

def _created_at_Windows(path):
    p = Path(path)
    stat = p.stat()
    timestamp = stat.st_ctime
    dt_obj = dt.fromtimestamp(timestamp)
    return dt_obj.year, dt_obj.month


if sys.platform.startswith('win'):
    created_at = _created_at_Windows
else:
    created_at = _created_at_Linux
```

Перевіримо працездатність функції.

In [34]:
```
created_at(el)
```

Out[34]:
```
(2024, 11)
```

In [35]:
```
y, m = created_at(el)
print(y, m)
```

```
2024 11
```

Повертаємось до задачі обходу та виведення

Тепер для шляху `el` виведемо:
відповідний йому відносний шлях, далі на тому ж рядку `file` або `dir`
для файлу на новому рядку з абзацного відступу виведемо його розширення в 10 символів та розмір в 10 символів

In [36]:
```
s = '.txt'
size = 12345
print(f'\t {s:10} {size:10}')
```

```
        .txt            12345
```

Записати код, що виводить інформацію про файл як у прикладі нижче

In [37]:
```
if el.is_dir():
    type = "dir"
else:
    type="file"

print(f'{el} --- {type}')
print(f'\t{el.suffix}    {el.stat().st_size}')
```

```
\Users\User\source\repos\lab2\.vs\lab2\v17\ipch\AutoPCH\846260f32befc30c\LAB2.ipc
h --- file
        .ipch    69074944
```

Приклад формату виведення:

```
build/.cmake/api/v1/query/client-vscode/query.jsonfile --- file
.json      144
```

Зберемо обхід, за якого будемо виводити інформацію тільки про папки та звичайні файли.

У базовий обхід додаємо виведення інформації про файли.

In [38]:
```
for el in root.glob("**/*"):
    if el.is_dir():
        type = "dir"
    if el.is_file():
        type = "file"
    rel_path = el.relative_to(root)
    print (f"{rel_path} --- {type}")

    if el.is_file():
        print(f"\t{el.suffix}   {el.stat().st_size}")
```

```
.vs --- dir
CMakeLists.txt --- file
        .txt   742
CMakePresets.json --- file
        .json   1795
lab2 --- dir
out --- dir
.vs\CMake Overview --- file
          0
.vs\lab2 --- dir
.vs\ProjectSettings.json --- file
        .json   44
.vs\slnx.sqlite --- file
        .sqlite   114688
.vs\VSWorkspaceState.json --- file
        .json   491
lab2\CMakeLists.txt --- file
        .txt   573
lab2\lab2.cpp --- file
        .cpp   455
lab2\lab2.h --- file
        .h   418
out\build --- dir
out\build\x64-debug --- dir
out\build\x64-debug\.cmake --- dir
out\build\x64-debug\.ninja_deps --- file
          160
out\build\x64-debug\.ninja_log --- file
          824
out\build\x64-debug\build.ninja --- file
        .ninja   13814
out\build\x64-debug\CMakeCache.txt --- file
        .txt   15134
out\build\x64-debug\CMakeFiles --- dir
out\build\x64-debug\cmake_install.cmake --- file
        .cmake   1674
out\build\x64-debug\lab2 --- dir
out\build\x64-debug\Testing --- dir
out\build\x64-debug\VSInheritEnvironments.txt --- file
        .txt   12
out\build\x64-debug\.cmake\api --- dir
out\build\x64-debug\CMakeFiles\3.29.5-msvc4 --- dir
out\build\x64-debug\CMakeFiles\cmake.check_cache --- file
        .check_cache   86
out\build\x64-debug\CMakeFiles\CMakeConfigureLog.yaml --- file
        .yaml   15979
out\build\x64-debug\CMakeFiles\pkgRedirects --- dir
out\build\x64-debug\CMakeFiles\rules.ninja --- file
        .ninja   4353
out\build\x64-debug\CMakeFiles\ShowIncludes --- dir
out\build\x64-debug\CMakeFiles\TargetDirectories.txt --- file
        .txt   410
out\build\x64-debug\lab2\CMakeFiles --- dir
out\build\x64-debug\lab2\cmake_install.cmake --- file
        .cmake   1069
out\build\x64-debug\lab2\lab2.exe --- file
        .exe   74240
out\build\x64-debug\lab2\lab2.ilk --- file
        .ilk   845184
out\build\x64-debug\lab2\lab2.pdb --- file
```

```
        .pdb   2371584
out\build\x64-debug\Testing\Temporary --- dir
out\build\x64-debug\Testing\Temporary\LastTest.log --- file
        .log   150
out\build\x64-debug\lab2\CMakeFiles\lab2.dir --- dir
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\embed.manifest --- file
        .manifest   406
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\intermediate.manifest --- file
        .manifest   381
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\lab2.cpp.obj --- file
        .obj   100897
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\lab2.cpp.obj.enc --- file
        .enc   79187
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\manifest.rc --- file
        .rc   183
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\manifest.res --- file
        .res   472
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\vc140.idb --- file
        .idb   183296
out\build\x64-debug\lab2\CMakeFiles\lab2.dir\vc140.pdb --- file
        .pdb   552960
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeCCompiler.cmake --- file
        .cmake   3166
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeCXXCompiler.cmake --- file
        .cmake   6180
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeDetermineCompilerABI_C.bin --- f
ile
        .bin   53248
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeDetermineCompilerABI_CXX.bin ---
file
        .bin   53248
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeRCCompiler.cmake --- file
        .cmake   276
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CMakeSystem.cmake --- file
        .cmake   395
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC --- dir
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX --- dir
out\build\x64-debug\CMakeFiles\ShowIncludes\foo.h --- file
        .h   2
out\build\x64-debug\CMakeFiles\ShowIncludes\main.c --- file
        .c   33
out\build\x64-debug\CMakeFiles\ShowIncludes\main.obj --- file
        .obj   650
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC\CMakeCCompilerId.c --- fi
le
        .c   28391
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC\CMakeCCompilerId.exe ---
file
        .exe   109568
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC\CMakeCCompilerId.obj ---
file
        .obj   1967
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdC\tmp --- dir
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX\CMakeCXXCompilerId.cpp
--- file
        .cpp   27942
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX\CMakeCXXCompilerId.exe
--- file
        .exe   109568
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX\CMakeCXXCompilerId.obj
```

```
--- file
        .obj   2034
out\build\x64-debug\CMakeFiles\3.29.5-msvc4\CompilerIdCXX\tmp --- dir
out\build\x64-debug\.cmake\api\v1 --- dir
out\build\x64-debug\.cmake\api\v1\query --- dir
out\build\x64-debug\.cmake\api\v1\reply --- dir
out\build\x64-debug\.cmake\api\v1\query\client-MicrosoftVS --- dir
out\build\x64-debug\.cmake\api\v1\reply\cache-v2-b0a87ca0b91c5463b806.json --- fi
le
        .json   23295
out\build\x64-debug\.cmake\api\v1\reply\cmakeFiles-v1-15c1d6e776bc169829bc.json -
-- file
        .json   6421
out\build\x64-debug\.cmake\api\v1\reply\codemodel-v2-42ba6b42696e52a62d8f.json --
- file
        .json   1306
out\build\x64-debug\.cmake\api\v1\reply\directory-.-Debug-d0094a50bb2071803777.js
on --- file
        .json   168
out\build\x64-debug\.cmake\api\v1\reply\directory-lab2-Debug-a34de045653a0b808c9
1.json --- file
        .json   174
out\build\x64-debug\.cmake\api\v1\reply\index-2024-11-09T12-28-30-0324.json --- f
ile
        .json   2663
out\build\x64-debug\.cmake\api\v1\reply\target-lab2-Debug-aabaa3c6fb531ff36ade.js
on --- file
        .json   1936
out\build\x64-debug\.cmake\api\v1\reply\toolchains-v1-d19162570acee6f721be.json -
-- file
        .json   1417
out\build\x64-debug\.cmake\api\v1\query\client-MicrosoftVS\query.json --- file
        .json   144
.vs\lab2\FileContentIndex --- dir
.vs\lab2\v17 --- dir
.vs\lab2\FileContentIndex\7fcd3efe-1a6b-4d3d-ba1f-adb3755c5c95.vsidx --- file
        .vsidx   10406
.vs\lab2\v17\.wsuo --- file
           28160
.vs\lab2\v17\Browse.VC.db --- file
        .db   19865600
.vs\lab2\v17\DocumentLayout.backup.json --- file
        .json   2252
.vs\lab2\v17\DocumentLayout.json --- file
        .json   2254
.vs\lab2\v17\ipch --- dir
.vs\lab2\v17\ipch\AutoPCH --- dir
.vs\lab2\v17\ipch\AutoPCH\846260f32befc30c --- dir
.vs\lab2\v17\ipch\AutoPCH\846260f32befc30c\LAB2.ipch --- file
        .ipch   69074944
```

Визначимо, файли з якими розширеннями присутні в папці, що обходиться.

▶ Підказка 1 (про алгоритм).

▶ Підказка 2 (про деталі реалізації).

Значенням змінної `suffixes` зробіть множину наявних розширень файлів.

In [39]:
```
root = Path(".")
extensions = set()
for el in root.glob('**/*'):
        if el.is_file():
            extensions.add(el.suffix.lower())
print(extensions)
```

```
{'', '.ini', '.pxd', '.cff', '.cfg', '.1', '.pth', '.f90', '.hpp', '.lua', '.in',
'.lib', '.cpp', '.css', '.h', '.md', '.pdb', '.sh', '.orig', '.xml', '.apache',
'.pot', '.pyf', '.mat', '.pyx', '.csv', '.pem', '.po', '.ani', '.build', '.ipyn
b', '.key', '.whl', '.exe', '.arff', '.toml', '.yaml', '.c', '.npy', '.zip', '.py
i', '.mplstyle', '.bat', '.bz2', '.external', '.nc', '.bsd', '.template', '.jso
n', '.pxi', '.pkl', '.ico', '.f95', '.sav', '.pdf', '.npz', '.wav', '.typed', '.a
fm', '.mo', '.svg', '.tmpl', '.dat', '.ps1', '.j2', '.txt', '.eot', '.pc', '.woff
2', '.xrc', '.jpg', '.desktop', '.unicode', '.png', '.pyc', '.gz', '.html', '.cm
d', '.dll', '.crt', '.zi', '.yml', '.f', '.lock', '.sha256sum', '.jupyterlab-work
space', '.map', '.a', '.fits', '.lark', '.ttf', '.tab', '.rst', '.pyd', '.egg',
'.js', '.tpl', '.woff', '.inc', '.fish', '.py'}
```

In [40]:
```
extensions
```

Out[40]:
```
{'',
 '.1',
 '.a',
 '.afm',
 '.ani',
 '.apache',
 '.arff',
 '.bat',
 '.bsd',
 '.build',
 '.bz2',
 '.c',
 '.cff',
 '.cfg',
 '.cmd',
 '.cpp',
 '.crt',
 '.css',
 '.csv',
 '.dat',
 '.desktop',
 '.dll',
 '.egg',
 '.eot',
 '.exe',
 '.external',
 '.f',
 '.f90',
 '.f95',
 '.fish',
 '.fits',
 '.gz',
 '.h',
 '.hpp',
 '.html',
 '.ico',
 '.in',
 '.inc',
 '.ini',
 '.ipynb',
 '.j2',
 '.jpg',
 '.js',
 '.json',
 '.jupyterlab-workspace',
 '.key',
 '.lark',
 '.lib',
 '.lock',
 '.lua',
 '.map',
 '.mat',
 '.md',
 '.mo',
 '.mplstyle',
 '.nc',
 '.npy',
 '.npz',
 '.orig',
 '.pc',
```

```
 '.pdb',
 '.pdf',
 '.pem',
 '.pkl',
 '.png',
 '.po',
 '.pot',
 '.ps1',
 '.pth',
 '.pxd',
 '.pxi',
 '.py',
 '.pyc',
 '.pyd',
 '.pyf',
 '.pyi',
 '.pyx',
 '.rst',
 '.sav',
 '.sh',
 '.sha256sum',
 '.svg',
 '.tab',
 '.template',
 '.tmpl',
 '.toml',
 '.tpl',
 '.ttf',
 '.txt',
 '.typed',
 '.unicode',
 '.wav',
 '.whl',
 '.woff',
 '.woff2',
 '.xml',
 '.xrc',
 '.yaml',
 '.yml',
 '.zi',
 '.zip'}
```

Перетворіть її на відсортований список `suffixes_sorted`.

In [41]:
```
suffixes_sorted = sorted(extensions)
```

Виведіть знайдені розширення у відсортованому порядку.

In [42]:
```
for el in suffixes_sorted:
    print(el, end=', ')
print()
```

```
, .1, .a, .afm, .ani, .apache, .arff, .bat, .bsd, .build, .bz2, .c, .cff, .cfg, .
cmd, .cpp, .crt, .css, .csv, .dat, .desktop, .dll, .egg, .eot, .exe, .external, .
f, .f90, .f95, .fish, .fits, .gz, .h, .hpp, .html, .ico, .in, .inc, .ini, .ipynb,
.j2, .jpg, .js, .json, .jupyterlab-workspace, .key, .lark, .lib, .lock, .lua, .ma
p, .mat, .md, .mo, .mplstyle, .nc, .npy, .npz, .orig, .pc, .pdb, .pdf, .pem, .pk
l, .png, .po, .pot, .ps1, .pth, .pxd, .pxi, .py, .pyc, .pyd, .pyf, .pyi, .pyx, .r
st, .sav, .sh, .sha256sum, .svg, .tab, .template, .tmpl, .toml, .tpl, .ttf, .txt,
.typed, .unicode, .wav, .whl, .woff, .woff2, .xml, .xrc, .yaml, .yml, .zi, .zip,
```

Скільки різних розширень було знайдено? (Виведіть це значення. Вручну нічого не рахуємо!)

▶ Підказка

In [43]:
```
print(len(extensions))
```

```
101
```

Знайдемо скільки файлів з певним розширенням наявно у досліджуваній папці.

Для кожного розширення визначимо, скільки файлів з цим розширенням присутні в папці, що обходиться.

▶ Підказка 1 (про алгоритм).

▶ Підказка 2 (про деталі реалізації).

Значенням змінної `d` зробіть словник, що знає кількість файлів для кожного наявного розширення.

In [44]:
```
d = {}

for el in root.glob("**/*"):
    if el.is_file():
        ext = el.suffix.lower()
        d[ext] = d.setdefault(ext, 0) + 1
```

In [45]:
```
d
```

Out[45]:

```
{'': 1290,
 '.ipynb': 8,
 '.csv': 37,
 '.cfg': 9,
 '.py': 6855,
 '.bat': 3,
 '.fish': 1,
 '.ps1': 1,
 '.exe': 59,
 '.desktop': 2,
 '.1': 2,
 '.js': 781,
 '.txt': 232,
 '.map': 258,
 '.json': 252,
 '.j2': 32,
 '.tpl': 10,
 '.css': 20,
 '.orig': 80,
 '.woff': 75,
 '.eot': 9,
 '.woff2': 9,
 '.svg': 32,
 '.ttf': 47,
 '.html': 30,
 '.png': 30,
 '.pth': 1,
 '.whl': 1,
 '.pyd': 210,
 '.typed': 84,
 '.pyi': 2226,
 '.dat': 1112,
 '.pem': 2,
 '.h': 46,
 '.lib': 67,
 '.yml': 7,
 '.c': 17,
 '.pxd': 28,
 '.dll': 10,
 '.toml': 4,
 '.rst': 9,
 '.cff': 1,
 '.lark': 5,
 '.a': 109,
 '.md': 25,
 '.tmpl': 2,
 '.pyc': 6849,
 '.pxi': 1,
 '.tab': 4,
 '.zi': 1,
 '.crt': 1,
 '.key': 1,
 '.xml': 2,
 '.bz2': 1,
 '.gz': 6,
 '.mo': 1,
 '.po': 3,
 '.apache': 4,
 '.bsd': 4,
 '.pyx': 14,
```

```
 '.npz': 25,
 '.npy': 12,
 '.build': 6,
 '.sav': 47,
 '.nc': 3,
 '.wav': 22,
 '.ani': 1,
 '.mat': 110,
 '.arff': 16,
 '.zip': 1,
 '.egg': 1,
 '.pkl': 1,
 '.cpp': 6,
 '.fits': 1,
 '.ini': 3,
 '.pc': 1,
 '.template': 1,
 '.f90': 61,
 '.f': 24,
 '.pyf': 7,
 '.inc': 1,
 '.f95': 1,
 '.lua': 1,
 '.pdf': 11,
 '.xrc': 1,
 '.jpg': 1,
 '.mplstyle': 29,
 '.afm': 60,
 '.pot': 2,
 '.ico': 8,
 '.yaml': 4,
 '.jupyterlab-workspace': 2,
 '.lock': 6,
 '.in': 3,
 '.external': 1,
 '.pdb': 6,
 '.hpp': 10,
 '.sh': 3,
 '.cmd': 1,
 '.sha256sum': 1,
 '.unicode': 1}
```

In [46]:
```
suffixes_cnts = list(d.items())
for el in suffixes_cnts:
    print(el, end=', ')
print()
```

```
('', 1290), ('.ipynb', 8), ('.csv', 37), ('.cfg', 9), ('.py', 6855), ('.bat', 3),
('.fish', 1), ('.ps1', 1), ('.exe', 59), ('.desktop', 2), ('.1', 2), ('.js', 78
1), ('.txt', 232), ('.map', 258), ('.json', 252), ('.j2', 32), ('.tpl', 10), ('.c
ss', 20), ('.orig', 80), ('.woff', 75), ('.eot', 9), ('.woff2', 9), ('.svg', 32),
('.ttf', 47), ('.html', 30), ('.png', 30), ('.pth', 1), ('.whl', 1), ('.pyd', 21
0), ('.typed', 84), ('.pyi', 2226), ('.dat', 1112), ('.pem', 2), ('.h', 46), ('.l
ib', 67), ('.yml', 7), ('.c', 17), ('.pxd', 28), ('.dll', 10), ('.toml', 4), ('.r
st', 9), ('.cff', 1), ('.lark', 5), ('.a', 109), ('.md', 25), ('.tmpl', 2), ('.py
c', 6849), ('.pxi', 1), ('.tab', 4), ('.zi', 1), ('.crt', 1), ('.key', 1), ('.xm
l', 2), ('.bz2', 1), ('.gz', 6), ('.mo', 1), ('.po', 3), ('.apache', 4), ('.bsd',
4), ('.pyx', 14), ('.npz', 25), ('.npy', 12), ('.build', 6), ('.sav', 47), ('.n
c', 3), ('.wav', 22), ('.ani', 1), ('.mat', 110), ('.arff', 16), ('.zip', 1), ('.
egg', 1), ('.pkl', 1), ('.cpp', 6), ('.fits', 1), ('.ini', 3), ('.pc', 1), ('.tem
plate', 1), ('.f90', 61), ('.f', 24), ('.pyf', 7), ('.inc', 1), ('.f95', 1), ('.l
ua', 1), ('.pdf', 11), ('.xrc', 1), ('.jpg', 1), ('.mplstyle', 29), ('.afm', 60),
('.pot', 2), ('.ico', 8), ('.yaml', 4), ('.jupyterlab-workspace', 2), ('.lock',
6), ('.in', 3), ('.external', 1), ('.pdb', 6), ('.hpp', 10), ('.sh', 3), ('.cmd',
1), ('.sha256sum', 1), ('.unicode', 1),
```

Знайдемо найбільш популярні розширення.

▶ Підказки

Є список пар.

In [47]:
```
lst = [(1, 2), (1, 5), (8, 2), (2, 4)]
lst
```

Out[47]:
```
[(1, 2), (1, 5), (8, 2), (2, 4)]
```

Відсортуємо пари лексикографічно.

In [48]:
```
lst.sort()
lst
```

Out[48]:
```
[(1, 2), (1, 5), (2, 4), (8, 2)]
```

Відсортуємо пари за ключем. В якості ключа візьмемо обернену пару.

In [49]:
```
lst.sort(key = lambda el: (el[1], el[0]))
lst
```

Out[49]:
```
[(1, 2), (8, 2), (2, 4), (1, 5)]
```

Знайти найбільшу кількість файлів, що мають однакове розширення. Присвоїти значення змінній `max_count`.

▶ Підказка

In [50]:
```
max_count = max(d.items(), key =lambda x: x[1])
max_count
```

Out[50]:
```
('.py', 6855)
```

In [51]:
```
max_count = max_count[1]
max_count
```

Out[51]: 6855

Вивести всі розширення, що відповідають цій найбільшій кількості входжень.

▶ Підказка

In [52]:
```
max_ext = [ext for ext, count in d.items() if count == max_count]
max_ext
```

Out[52]: ['.py']

Класифікація розширень

Як за розширенням визначити, чи потрапляє воно в задані переліки?

▶ Підказка

Запишіть логічний вираз, що задає ознаку того, що розширення .txt наявне у білому списку.

▶ Відповідь

In [53]:
```
'.txt' in WHITE_LIST
```

Out[53]: True

Напишіть функцію `kind`, що за розширенням повертає рядок з символом W,C,O (див. умову)

In [54]:
```
def kind(extension):
    if extension in WHITE_LIST:
        return 'W'
    elif extension in CODE_LIST:
        return 'C'
    else:
        return 'O'
```

Модифікуйте словник так, щоб для кожного розширення він зберігав:

1. кількість файлів
2. їх сумарний розмір
3. їх сумарну кількість рядків тексту (тільки для файлів з текстом програм, для інших — 0)

Значеннями словника можуть бути списки з трьох елементів (кількість файлів, сумарний розмір, сумарна кількість рядків тексту)

In [55]:
```
sample = {'.cpp':[23, 123456, 56]}   # приклад
```

При перегляді файлів слід оновлювати поля значення. Наприклад
`sample['.cpp'][0]+=1` збільшує кількість файлів з розширенням на 1
`sample['.cpp'][1]+=1234` додає розмір файлу (1234) до сумарного розміру

Якщо треба рахувати інформацію не тільки в розрізі розширень, а ще й місяців, то в якості ключів можна використати трійки: рік, місяць, розширення.

In [56]:
```
sample = {(2024, 9, '.cpp'):[23, 123456, 56]}   # приклад
sample
```

Out[56]:
```
{(2024, 9, '.cpp'): [23, 123456, 56]}
```

In [57]:
```
sample[2024, 9, '.cpp']
```

Out[57]:
```
[23, 123456, 56]
```

Кількість рядків файлу

Створити шлях до якого-небудь власного файлу з текстом програми. (Підставити власний шлях.)

In [58]:
```
sample = root.joinpath('C:\\Users\\User\\source\\repos\\lab2\\lab2\\lab2.cpp')
print(sample.exists())
```

```
True
```

In [59]:
```
sample.stat().st_size
```

Out[59]:
```
455
```

Знайдемо кількість рядків файлу

In [60]:
```
if sample.stat().st_size < 2**20:
    lines = 0
    with open(sample) as f:
        for _ in f:  # ітеруємо рядками    _  тут ім'я змінної, що обходить рядк
            lines += 1
    print(lines)
```

```
29
```

Якщо рядок порожній, то він хибний. Якщо рядок містить тільки білі символи, то це можна визначити за допомогою методу `isspace` (методи класу `str`)

In [61]:
```
help(str.isspace)
```

```
Help on method_descriptor isspace:

isspace(self, /) unbound builtins.str method
    Return True if the string is a whitespace string, False otherwise.

    A string is whitespace if all characters in the string are whitespace and the
re
    is at least one character in the string.
```

Модифікувати підрахунок кількості рядків файлу так, щоб додатково рахувалась кількість рядків файлу, що містять небілі символи (непорожні і не тільки з білих символів). У кінці вивести кількість рядків та кількість рядків з небілими символами.

In [62]:
```
if sample.stat().st_size < 2**20:
    lines = 0
    notempty_lines = 0
    with open(sample) as f:
        for s in f:  # ітеруємо рядками
            lines += 1
            if s and not s.isspace():
                notempty_lines += 1
    print(lines, notempty_lines)
```

```
29 24
```

Написати функцію, що визначає для файла кількість його рядків з небілими символами.

```
def lines_counter(path):
    ???
    return ???
```

In [63]:
```
def lines_counter(path):
    count = 0
    with open(path, encoding ="utf-8", errors="ignore") as f:
        for line in f:
            if line and not line.isspace():
                count+=1
    return count
```

Перевіримо нашу фукцію зоча б трішечки.

In [64]:
```
lines_counter(sample)
```

Out[64]:
```
24
```

Як зберегти табличну інформацію в csv-файл

Приклад даних. Табличні дані (як список контейнерів). (Використовувати списки чи кортежі тут різниці не має.) Та перелік заголовків до таблиці.

In [65]:
```
data = [(2025, 1, 13, 23456, 4.5, 99, 'текст1'),
        (2025, 7, 35, 23456, 4.5, 99, 'текст2'),
        (2025, 4, 34, 23456, 4.5, 99, 'текст3'),
        (2025, 11, 24, 23456, 4.5, 99, 'текст1')]
headers = ['h1', 'h2', 'h3', 'h4', 'h5', 'h6', 'h7']
```

Роботу виконуватиме бібліотека (документація), більш конкретно `writer` (документація) . Для читання є `reader` (документація).

In [66]:
```
import csv
```

In [67]:
```
with open('data.csv', 'w', encoding='utf-8', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(headers)
```

```
    for row in data:
        writer.writerow(row)
```

In [68]:
```
with open('data.csv', encoding='utf-8', newline='') as f:
    reader = csv.reader(f)
    headers = next(reader)
    lst = list()
    for row in reader:
        lst.append(row)
        print(row)
```

```
['2025', '1', '13', '23456', '4.5', '99', 'текст1']
['2025', '7', '35', '23456', '4.5', '99', 'текст2']
['2025', '4', '34', '23456', '4.5', '99', 'текст3']
['2025', '11', '24', '23456', '4.5', '99', 'текст1']
```

Збирання задачі

Нехай протягом навчання програмні проєкти з різних предметів зберігалися в папках ... . Є перелік папок `folders`, в яких зберігаються ці проєкти (задає виконавець, виходячи з вмісту своїх носіїв даних).
Примітка. До цього переліку мають входити не окремі папки з проєктами, а папки, що містять, наприклад, всі проєкти такої-то дисципліни такого-то семестру тощо. Інакше кажучи, не папка одного solution, а папка, в якій зберігаються solutions (якщо в термінології Visual Studio).

In [69]:
```
folders = [r"C:\Users\User\Projects\C++.SEM1",
           r"C:\Users\User\Projects\C++.SEM3",
           r"C:\Users\User\Projects\OPP.SEM1"]
# задати перелік
folders
```

Out[69]:
```
['C:\\Users\\User\\Projects\\C++.SEM1',
 'C:\\Users\\User\\Projects\\C++.SEM3',
 'C:\\Users\\User\\Projects\\OPP.SEM1']
```

Щоб профільтрувати допоміжні файли середовищ, введемо "білий список" розширень (`WHITE_LIST`) та список розширень файлів з текстом програм(задається) (`CODE_LIST`).

Ці переліки були задані в умові.

In [70]:
```
for el in WHITE_LIST: print(el, end=' ')
```

```
.c .ipynb .cpp .txt .js .java .h .pyw .cs .json .hpp .html .py
```

In [71]:
```
for el in CODE_LIST: print(el, end=' ')
```

```
.c .cpp .js .java .h .pyw .cs .hpp .py
```

Задача 1 Скільки різних розширень зустрічається у папках з заданого списку (тут і далі: включаючи вкладені), скільки файлів з кожним розширенням.
(Для цієї задачі розглядаємо всі розширення, а не тільки "білого списку".)

Відповідну інформацію вивести на екран охайною табличкою. На рядку виводиться розширення, кількість файлів та ознака (C,W,O відповідно для файлів з текстом програм (відповідно до `CODE_LIST`), інших з білого списку, та тих, що не потрапили в білий список). Інформація подається за спаданням кількості файлів.

Наприклад,

```
.txt     155    W
.ipynb    73    W
.cpp      45    C
.exe       3    O
```

1. Створити порожній словник `d`. Його ключами будуть розширення, а значеннями відповідні кількості файлів.
2. Обійти всі папки з переліку `folders`. Для кожної папки запустити рекурсивний обхід, під час якого рахувати кількості файлів окремо для кожного розширення.
3. Перетворити словник на список пар і відсортувати цей список по другій координаті в спадному порядку.
4. Обійти список пар і для кожного елемента вивести запитувану інформацію за допомогою f-рядка (налаштовуючи ширину поля виведення(, щоб досягти охайної таблиці.

Використати раніше розроблену функцію `kind`

In [72]:
```
from pathlib import Path

CODE_LIST = {'.cpp', '.hpp', '.c', '.h', '.py', '.pyw', '.cs', '.js', '.java'}
WHITE_LIST = CODE_LIST | {'.html', '.json', '.txt', '.ipynb'}

folders = [r"C:\Users\User\Projects\C++.SEM1",
           r"C:\Users\User\Projects\C++.SEM3",
           r"C:\Users\User\Projects\OPP.SEM1"]
```

In [73]:
```
def kind(extension):
    if extension in WHITE_LIST:
        return 'W'
    elif extension in CODE_LIST:
        return 'C'
    else:
        return 'O'
```

In [74]:
```
d={}

for folder in folders:
    folder_path = Path(folder)
    for el in folder_path.glob("**/*"):
        if el.is_file():
            ext = el.suffix.lower()
            d[ext]=d.setdefault(ext,0)+1

sorted_list = sorted(d.items(), key=lambda x:x[1], reverse=True)
print (f"{'Ext':>23} {'Cnt':>6} {'Cat':>3}")
for ext, cnt in sorted_list:
```

```
    cat = kind(ext)
    print(f"{ext:>23} {cnt:6} {cat:>3}")
```

Ext	Cnt	Cat
.xml	3744	O
.dll	2719	O
.class	1056	O
.mui	1032	O
.png	1021	O
.java	775	W
.txt	599	W
.pdb	447	O
.stamp	325	O
.cache	314	O
.winmd	252	O
.jar	236	O
.zip	221	O
.flat	175	O
.json	163	W
.properties	159	O
.flata	156	O
.cs	146	W
	122	O
.a	78	O
.exe	45	O
.vsidx	45	O
.ipch	42	O
.tlog	41	O
.db	40	O
.dylib	40	O
.bin	38	O
.ttf	38	O
.inputs	36	O
.items	27	O
.cmake	24	O
.intermediate	24	O
.editorconfig	23	O
.obj	21	O
.xaml	20	O
.props	20	O
.ll	20	O
.o	20	O
.v2	18	O
.plist	18	O
.resfiles	18	O
.manifest	15	O
.pri	15	O
.sln	14	O
.sample	14	O
.cpp	13	W
.outputs	12	O
.targets	11	O
.log	10	O
.csproj	10	O
.buildwithskipanalyzers	10	O
.h	9	W
.svg	9	O
.user	8	O
.ninja	8	O
.c	8	W
.idb	8	O
.up2date	8	O
.flag	8	O

```
              .xbf      8   O
              .ilk      7   O
   .lastbuildstate      6   O
     .appxmanifest      6   O
              .ico      6   O
             .msix      6   O
             .html      6   W
             .aidl      6   O
          .vcxproj      5   O
          .filters      5   O
           .recipe      5   O
           .sqlite      4   O
     .check_cache      4   O
            .yaml      4   O
              .so      4   O
             .dex      4   O
        .uptodate      4   O
              .rc      3   O
             .res      3   O
       .xcprivacy      3   O
             .p7x      3   O
             .man      3   O
             .513      3   O
             .cat      3   O
      .storyboard      3   O
             .enc      2   O
            .hash      2   O
          .config      2   O
            .xslt      2   O
          .backup      2   O
            .iobj      1   O
            .ipdb      1   O
```

Задача 2 За датами створення файлів для кожного місяця, починаючи з вересня 2024-го року, знайти таку інформацію: кількість файлів з розширеннями з "білого списку", їх сумарний обсяг у байтах, кількість їх рядків (враховувати тільки для файлів категорії C; білі та порожні рядки не рахувати), середній розмір файлу, середня кількість рядків.

Відповідну інформацію вивести на екран охайною табличкою в хронологічному порядку.

Наприклад,

```
2024  09   23   123456  256   5368   11
2024  10   23   123456  256   5368   11
2024  11  134   123456  256    921   11
```

1. Створити порожній словник `d`. Його ключами будуть пари (рік, місяць), а значеннями відповідні кількості файлів, їх сумарний обсяг, кількість їх рядків.
2. Обійти всі папки з переліку `folders`. Для кожної папки запустити рекурсивний обхід, під час якого обробляти тільки файли з розширеннями з білого списку.

При обробленні файлу визначати рік та місяць його створення та оновлювати інформацію в словнику (за потреби додавати ключі).

3. Перетворити словник на список пар і відсортувати цей список за хронологією.
4. Обійти список пар і для кожного елемента вивести запитувану інформацію за допомогою f-рядка (налаштовуючи ширину поля виведення), щоб досягти охайної таблиці. Середні значення обчислювати під час обходу, не зберігаючи в структурі даних.

Використати раніше розроблені раніше функції `created_at`, `lines_counter`

In [75]:
```
import sys
import datetime
from pathlib import Path
from datetime import datetime as dt
```

In [76]:
```
CODE_LIST = {'.cpp', '.hpp', '.c', '.h', '.py', '.pyw', '.cs', '.js', '.java'}
WHITE_LIST = CODE_LIST | {'.html', '.json', '.txt', '.ipynb'}

folders = [r"C:\Users\User\Projects\C++.SEM1",
           r"C:\Users\User\Projects\C++.SEM3",
           r"C:\Users\User\Projects\OPP.SEM1"]
```

In [77]:
```
def _created_at_Linux(path):
    p = Path(path)
    stat = p.stat()
    timestamp = min(stat.st_ctime, stat.st_mtime)
    dt_obj = dt.fromtimestamp(timestamp)
    return dt_obj.year, dt_obj.month

def _created_at_Windows(path):
    p = Path(path)
    stat = p.stat()
    timestamp = stat.st_ctime
    dt_obj = dt.fromtimestamp(timestamp)
    return dt_obj.year, dt_obj.month

if sys.platform.startswith('win'):
    created_at = _created_at_Windows
else:
    created_at = _created_at_Linux
```

In [78]:
```
def lines_counter(path):
   count = 0
   with open(path, encoding ="utf-8", errors="ignore") as f:
       for line in f:
           if line and not line.isspace():
               count+=1
   return count
```

In [79]:
```
d = {}

for folder in folders:
    folder_path = Path(folder)
    for el in folder_path.glob("**/*"):
        if el.is_file():
            ext=el.suffix.lower()
```

```
            if ext in WHITE_LIST:
                year, month = created_at(el)
                if(year,month)>=(2024, 9):
                    key = (year, month)
                    if key not in d:
                        d[key]=[0,0,0]
                    d[key][0]+=1
                    d[key][1]+=el.stat().st_size
                    if ext in CODE_LIST:
                        d[key][2]+=lines_counter(el)

items=sorted(d.items())
print (f"{'Year':>6} {'Mon':>4} {'Cnt':>6} {'Bytes':>9} {'AvgSize':>10} {'Lines'
for (year, month), (cnt, total_size, total_lines) in items:
    if cnt:
        avg_size = total_size // cnt
    else:
        avg_size=0
    if cnt:
        avg_lines = total_lines // cnt
    else:
        avg_lines=0
    print (f"{year:6} {month:4} {cnt:6} {total_size:10} {avg_size:8} {total_line
```

```
 Year  Mon    Cnt     Bytes    AvgSize    Lines    AvgLn
 2024   10     16      77898     4868     1488       93
 2024   11     78     372656     4777     4707       60
 2025   10    621    8604834    13856    84248      135
 2025   11    387    4803710    12412    21509       55
 2025   12    617    8601186    13940    84414      136
```

Задача 3 Для кожного місяця та кожного розширення, що зустрічаються при перегляді файлів, розміщених у папках зі списку, знайти таку інформацію: рік, місяць, розширення, кількість файлів, сумарний обсяг у байтах, кількість їх рядків (тільки для розширень категорії C; білі та порожні рядки не рахувати; для інших розширень цю клітину таблиці залишаємо незаповненою)

Цю інформацію записати в csv-файл з іменем `lab1_1.csv` (кодування тільки utf-8), на початку додати рядок з заголовками: year, mon, ext, cnt, size, lines, avg_size, avg_line.

1. Створити порожній словник `d`. Його ключами будуть пари (рік, місяць, розширення), а значеннями відповідні кількості файлів, їх сумарний обсяг, кількість їх рядків.
2. Обійти всі папки з переліку `folders`. Для кожної папки запустити рекурсивний обхід, під час якого обробляти тільки файли з розширеннями з білого списку. При обробленні файлу визначати рік та місяць його створення та оновлювати інформацію в словнику (за потреби додавати ключі).
3. Перетворити словник на список пар і відсортувати цей список за хронологією.
4. Обійти список пар і для кожного елемента вивести запитувану інформацію за допомогою f-рядка (налаштовуючи ширину поля виведення), щоб досягти охайної таблиці. Середні значення обчислювати під час обходу, не зберігаючи в структурі даних.

Використати раніше розроблені раніше функції `created_at` , `lines_counter`

Примітка. Не забороняється спочатку виконати задачу 3, а потім за її даними отримати результат задачі 2. Але це може виявитись зробити складніше, ніж у пропонованій вище схемі.

In [80]:
```
import sys
import datetime
from pathlib import Path
from datetime import datetime as dt
import csv
```

In [81]:
```
CODE_LIST = {'.cpp', '.hpp', '.c', '.h', '.py', '.pyw', '.cs', '.js', '.java'}
WHITE_LIST = CODE_LIST | {'.html', '.json', '.txt', '.ipynb'}

folders = [r"C:\Users\User\Projects\C++.SEM1",
           r"C:\Users\User\Projects\C++.SEM3",
           r"C:\Users\User\Projects\OPP.SEM1"]
```

In [82]:
```
def _created_at_Linux(path):
    p = Path(path)
    stat = p.stat()
    timestamp = min(stat.st_ctime, stat.st_mtime)
    dt_obj = dt.fromtimestamp(timestamp)
    return dt_obj.year, dt_obj.month

def _created_at_Windows(path):
    p = Path(path)
    stat = p.stat()
    timestamp = stat.st_ctime
    dt_obj = dt.fromtimestamp(timestamp)
    return dt_obj.year, dt_obj.month

if sys.platform.startswith('win'):
    created_at = _created_at_Windows
else:
    created_at = _created_at_Linux
```

In [83]:
```
def lines_counter(path):
   count = 0
   with open(path, encoding ="utf-8", errors="ignore") as f:
       for line in f:
           if line and not line.isspace():
               count+=1
   return count
```

In [84]:
```
d = {}
for folder in folders:
    folder_path = Path(folder)
    for el in folder_path.glob("**/*"):
        if el.is_file():
            ext = el.suffix.lower()
            if ext in WHITE_LIST:
                year, month = created_at(el)
                key = (year, month, ext)
```

```
                if key not in d:
                    d[key] = [0, 0, 0]

                d[key][0] += 1
                d[key][1] += el.stat().st_size

                if ext in CODE_LIST:
                    d[key][2] += lines_counter(el)
items = sorted(d.items())
with open("lab1_1.csv", "w", encoding="utf-8", newline="") as f:
    writer = csv.writer(f, delimiter =";")
    writer.writerow(["year", "mon", "ext", "cnt", "size", "lines", "avg_size", "
    for (year, month, ext), (cnt, size, lines) in items:

        if cnt > 0:
            avg_size = size // cnt
        else:
            avg_size = 0

        if ext in CODE_LIST:
            lines_out = lines
            if cnt > 0:
                avg_line = lines // cnt
            else:
                avg_line = 0
        else:
            lines_out = ""
            avg_line = ""
        writer.writerow([year, month, ext, cnt, size, lines_out, avg_size, avg_l
```

In [ ]: