Software Training

Task 1

Subtask 1

You are required to count the occurrence count of all words in a string.

Start by making an empty `dict` to store every word as a key and its value is an `int` representing the number of occurrences. Write a loop to go over a dummy list of words (strings). If a word already exists as a key in the dictionary, increment its corresponding value by 1. If the word doesn't exist yet, create its key-value pair easily using indexing, e.g `mydict[word] = 1`

For checking if a key exists in a dict, you may use a condition OR a `try-except` block. Consult the documentation:
`dict`: https://docs.python.org/3/library/stdtypes.html#mapping-types-dict
Search for a method or operation that returns a distinct value or raises an exception if a key is not found. You may also instead use a search engine to look for a boolean operation on dictionaries and keys.

Now place the code you made inside a function with the following definition:

count_words()

```python
def count_words(sentence: str) -> dict[str, int]:
  word_count_dict = {}
  # turn string into list
  # insert your loop here
  return word_count_dict
```

Required Code
One more requirement we need is a default argument that allows similar words with different capitalization to count towards the same word.

count_words

```python
def count_words(sentence: str, case_sensitive=False) -> dict[str, int]:
  word_count_dict = {}
  # turn string into list
  # insert your loop here
    # if case_sensitive == False:
      # normalize current word to upper or lower case
    # continue looping
  return word_count_dict
```

Make sure to consult the string documentation for managing its case/ capitalization, and for splitting into a list:
`str.lower()`: https://docs.python.org/3/library/stdtypes.html#str.lower
`str.split()`: https://docs.python.org/3/library/stdtypes.html#str.split

Testing Subtask 1

Save the previous function in a file called `word_counter.py` and download the tester main file from here:
`test_st1.py`: https://drive.google.com/file/d/1_VtxnnSmeah6r1RBRtEE8e8ecNeqv0LA/

```
Subtask 1
├─ test_st1.py+
└─ word_counter.py
```

+: file is given

Subtask 2

You will find a recap on file operations after the task description.

You are required to take user input as strings and write these sentences to a file in plaintext format. The program will only exit when you use `Ctrl+C`. So you have to save the input strings to the files line-by-line. Here's an example of the code being run and its output file:

Terminal I/O

```
  > Never gonna give you up
  > never gonna let you down
```

output.txt

```
Never gonna give you up
never gonna let you down
```

Required Code

save_sentence()

```python
from pathlib import Path
def save_sentence(sentence: str, file_name = 'output.txt') -> bool:
  file_path = Path(file_name)
  # open file with append mode; see page 4 for hints
  # write the string with a newline in the end
  # make sure the file is closed or use the "with" keyword
  # optionally handle permission exceptions with try-except and return False if an
    exception occurs, also print error description
  return True # success
```

Testing Subtask 2

Save the function in a file called `line_saver.py`; write your own testing file similar to that in **Subtask 1**

```
Subtask 2
├─ test_st2.py+
└─ line_saver.py
```

+: FILE TO BE INCLUDED IN SUBMISSION

Subtask 3

Take a second to review looping techniques on data-structures:
https://docs.python.org/3/tutorial/datastructures.html#looping-techniques

You are required to implement a new text-based structured data format, like json. Example of transformation from `dict` to this new format:

dict

```python
test_dict = {
  'Name': 'Jon',
  'Passport Number': 'A23B120',
  'Occupation': 'Airfoce Commander',
  'Married': True,
  'Age': 34
}
```

Output.txt

```
Name = Jon
Passport Number = A23B120
Occupation = Airfoce Commander
Married = True
Age = 34
```

Required Code

data_saver()

```python
from pathlib import Path
import json
def data_saver(data: dict, data_format = 'json', file_name = 'output') -> bool:
  # valid formats: json, txt
  file_name += data_format # output.txt vs output.json
  file_path = Path(file_name)
  if data_format == 'json':
    with file_path.open('w') as file:
      json.dump(data, file)
  elif data_format == 'txt':
    with file_path.open('a') as file: # append-mode
    # loop over data keys
    # for each key, write a new-line terminated string in the format: key = value
  # optionally handle exceptions, around both "with file_path.open()" blocks, with
   try-except and return False if an exception occurs, also print error description
  return True # success
```

Make sure to write your test running code and submit the file containing the function above, and the test running code.

Revision on File Handling

Please take some time to revise basic file operations:
`file` methods: https://docs.python.org/3/tutorial/inputoutput.html#methods-of-file-objects
`file.open(mode)` basic modes:

Mode	Description
r	Default. Opens existing file for reading only.
w	Write mode. If file doesn't exist, creates it. If it exists, erases all contents first.
a	Append mode. If you write to file, text is added to the end of the file. If the file doesn't exist, it is automatically created.
t	Text mode. Default with `r` mode, i.e `open()` is `open('rt')`. Can be used with other modes.
b	Binary mode. Can be used with other modes.
r+	Read-write mode. Opens existing file, places a cursor at the start of the file, allows you to move the cursor around for writing.

`file.open()` techniques:

Opening File for Read

```python
from pathlib import Path
file_path = Path('output.txt')
if file_path.exists():
  file = file_path.open()
  first_sentence = file.read_line()
  print(first_sentence)
  file.close()
```

(Unsafe) Opening File for Write

```python
from pathlib import Path
file_path = Path('output.txt')
file = file_path.open('w') # or file_path.open('a')
file.write('Hello World\n')
file.close() # very important when writing
```

(Safe) Opening File for Write

```python
from pathlib import Path
file_path = Path('output.txt')
with file_path.open('w') as file: # automatically closes the file after the block
  file.write('Hello World\n')
```