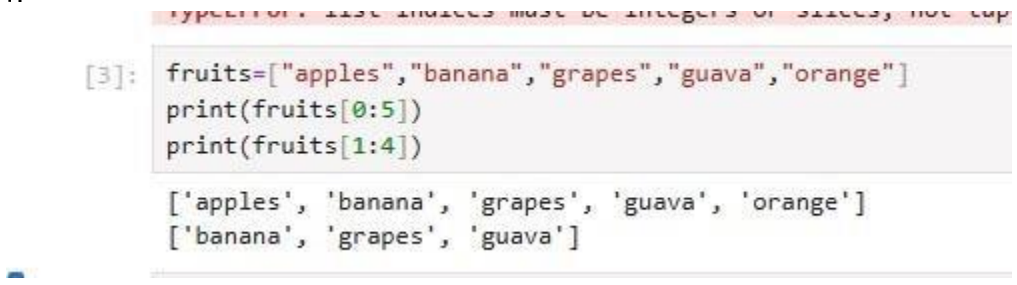
Kanishk Kaushik

23104015

LAB ASSIGNMENT 2

1.

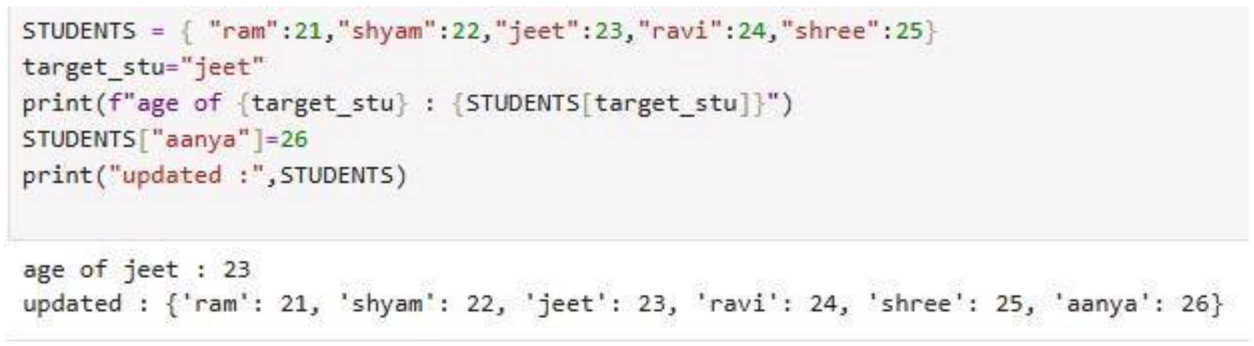


```
TypeError: list indices must be integers or slices, not tup
```

```
[3]: fruits=["apples","banana","grapes","guava","orange"]
     print(fruits[0:5])
     print(fruits[1:4])
```

```
['apples', 'banana', 'grapes', 'guava', 'orange']
['banana', 'grapes', 'guava']
```

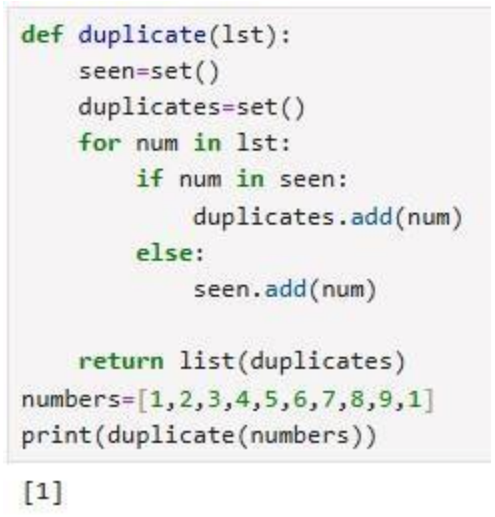
2.



```
STUDENTS = { "ram":21,"shyam":22,"jeet":23,"ravi":24,"shree":25}
target_stu="jeet"
print(f"age of {target_stu} : {STUDENTS[target_stu]}")
STUDENTS["aanya"]=26
print("updated :",STUDENTS)
```

```
age of jeet : 23
updated : {'ram': 21, 'shyam': 22, 'jeet': 23, 'ravi': 24, 'shree': 25, 'aanya': 26}
```

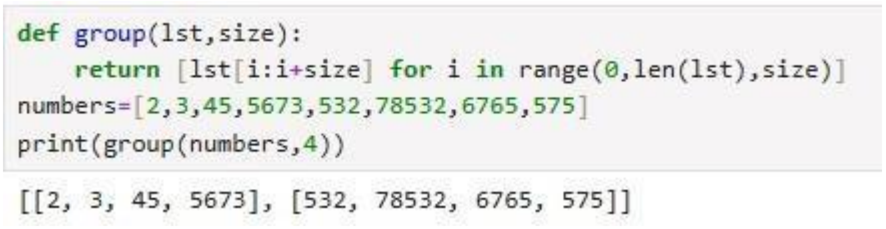
3.



```
def duplicate(lst):
    seen=set()
    duplicates=set()
    for num in lst:
        if num in seen:
            duplicates.add(num)
        else:
            seen.add(num)

    return list(duplicates)
numbers=[1,2,3,4,5,6,7,8,9,1]
print(duplicate(numbers))
```

```
[1]
```

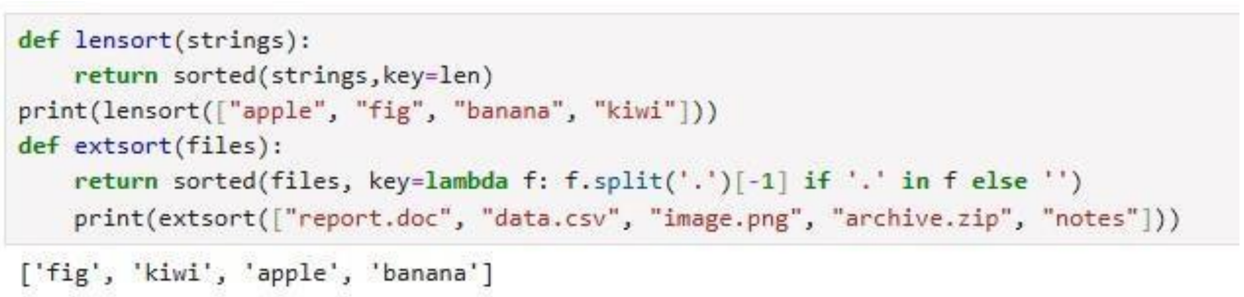
4.



```
def group(lst,size):
    return [lst[i:i+size] for i in range(0,len(lst),size)]
numbers=[2,3,45,5673,532,78532,6765,575]
print(group(numbers,4))
```

```
[[2, 3, 45, 5673], [532, 78532, 6765, 575]]
```

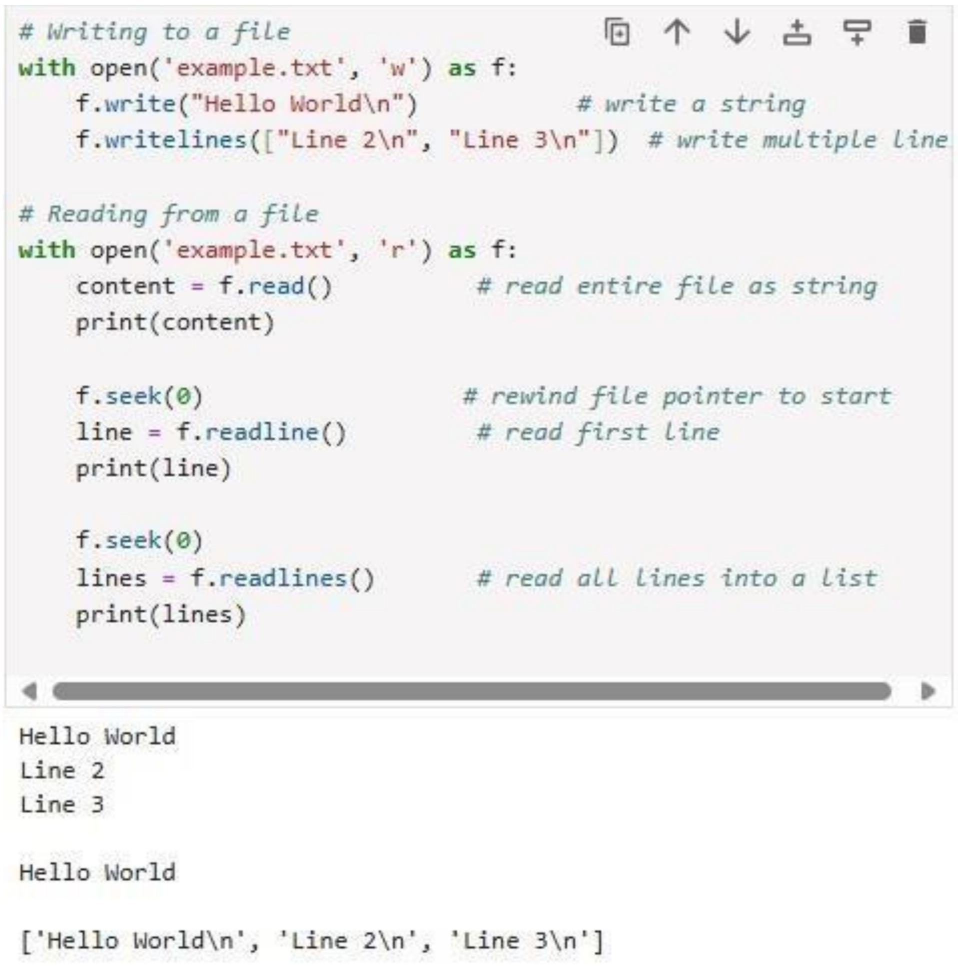
5.



```
def lensort(strings):
    return sorted(strings,key=len)
print(lensort(["apple", "fig", "banana", "kiwi"]))
def extsort(files):
    return sorted(files, key=lambda f: f.split('.')[-1] if '.' in f else '')
    print(extsort(["report.doc", "data.csv", "image.png", "archive.zip", "notes"]))
```

```
['fig', 'kiwi', 'apple', 'banana']
```

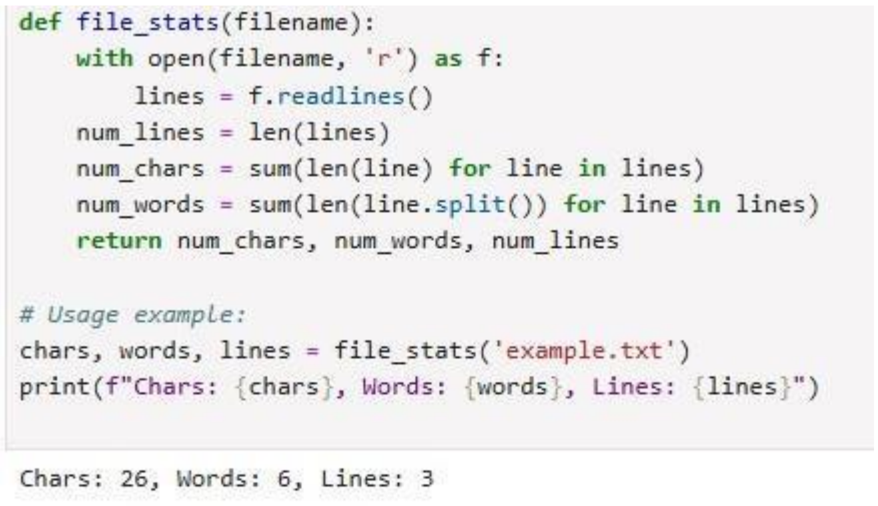
6.



```
# Writing to a file
with open('example.txt', 'w') as f:
    f.write("Hello World\n")            # write a string
    f.writelines(["Line 2\n", "Line 3\n"])  # write multiple line

# Reading from a file
with open('example.txt', 'r') as f:
    content = f.read()          # read entire file as string
    print(content)

    f.seek(0)                  # rewind file pointer to start
    line = f.readline()         # read first line
    print(line)

    f.seek(0)
    lines = f.readlines()       # read all lines into a list
    print(lines)
```

```
Hello World
Line 2
Line 3

Hello World

['Hello World\n', 'Line 2\n', 'Line 3\n']
```

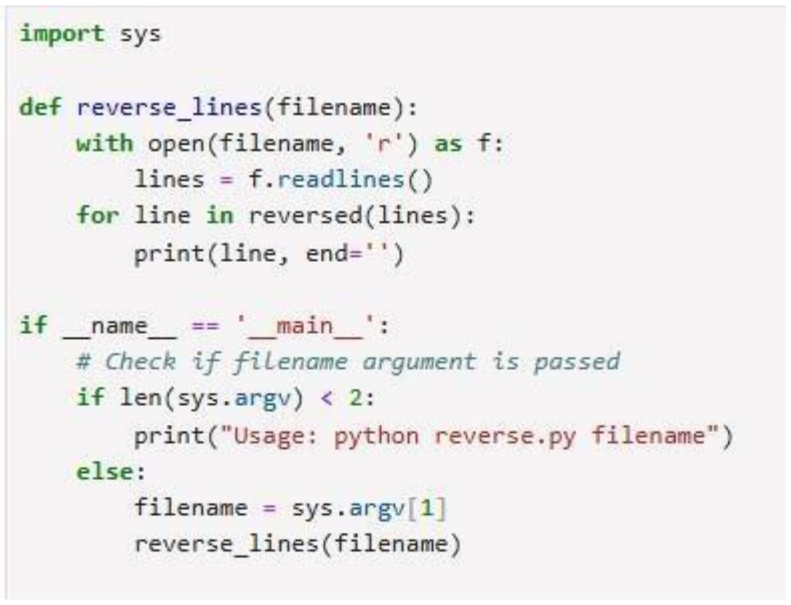
7.



```python
def file_stats(filename):
    with open(filename, 'r') as f:
        lines = f.readlines()
    num_lines = len(lines)
    num_chars = sum(len(line) for line in lines)
    num_words = sum(len(line.split()) for line in lines)
    return num_chars, num_words, num_lines

# Usage example:
chars, words, lines = file_stats('example.txt')
print(f"Chars: {chars}, Words: {words}, Lines: {lines}")
```

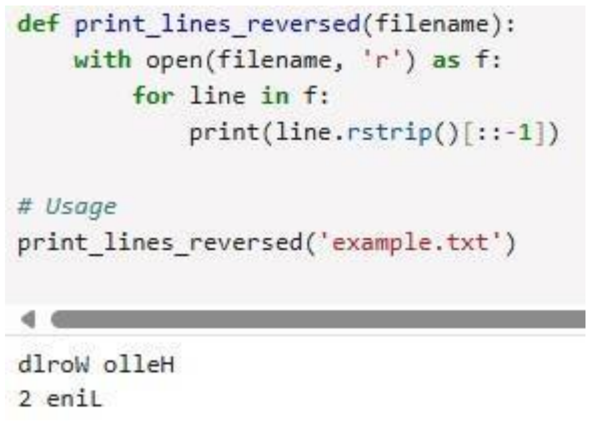
```
Chars: 26, Words: 6, Lines: 3
```

8.



```python
import sys

def reverse_lines(filename):
    with open(filename, 'r') as f:
        lines = f.readlines()
    for line in reversed(lines):
        print(line, end='')

if __name__ == '__main__':
    # Check if filename argument is passed
    if len(sys.argv) < 2:
        print("Usage: python reverse.py filename")
    else:
        filename = sys.argv[1]
        reverse_lines(filename)
```

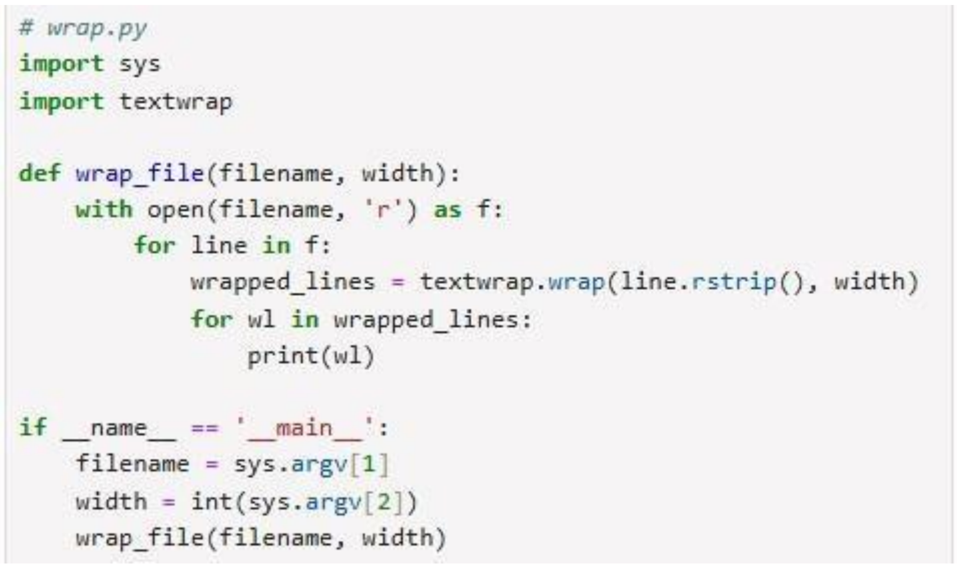
9.



```python
def print_lines_reversed(filename):
    with open(filename, 'r') as f:
        for line in f:
            print(line.rstrip()[::-1])

# Usage
print_lines_reversed('example.txt')
```

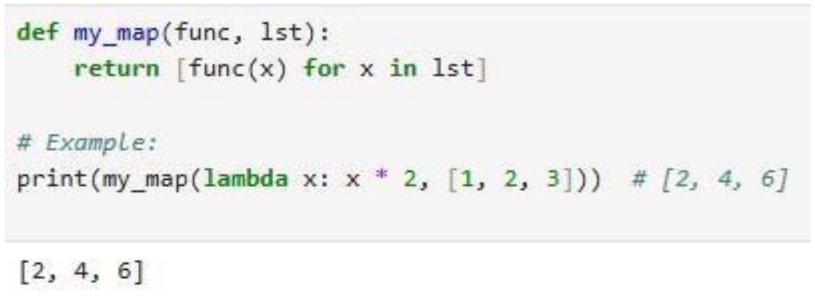
```
dlroW olleH
2 eniL
```

10.



```python
# wrap.py
import sys
import textwrap

def wrap_file(filename, width):
    with open(filename, 'r') as f:
        for line in f:
            wrapped_lines = textwrap.wrap(line.rstrip(), width)
            for wl in wrapped_lines:
                print(wl)

if __name__ == '__main__':
    filename = sys.argv[1]
    width = int(sys.argv[2])
    wrap_file(filename, width)
```

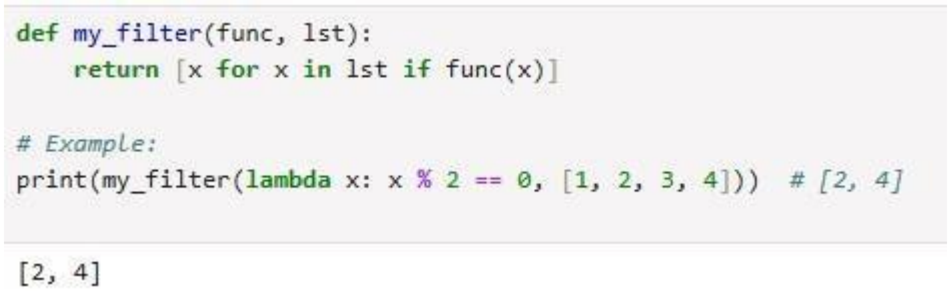
11.



```python
def my_map(func, lst):
    return [func(x) for x in lst]

# Example:
print(my_map(lambda x: x * 2, [1, 2, 3]))  # [2, 4, 6]
```

```
[2, 4, 6]
```

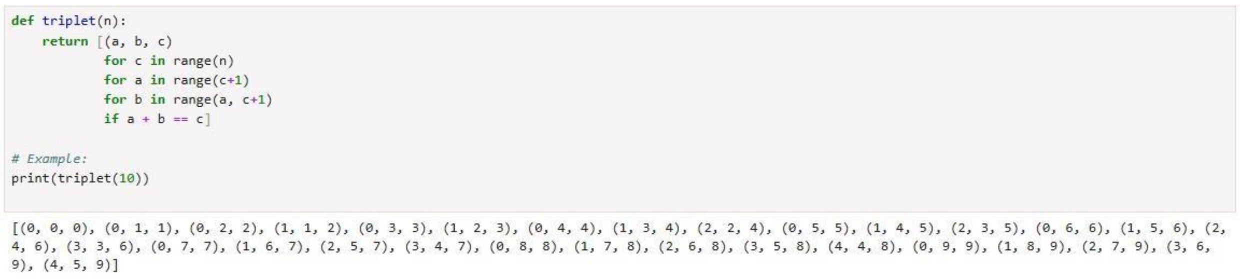
12.



```python
def my_filter(func, lst):
    return [x for x in lst if func(x)]

# Example:
print(my_filter(lambda x: x % 2 == 0, [1, 2, 3, 4]))  # [2, 4]
```

```
[2, 4]
```

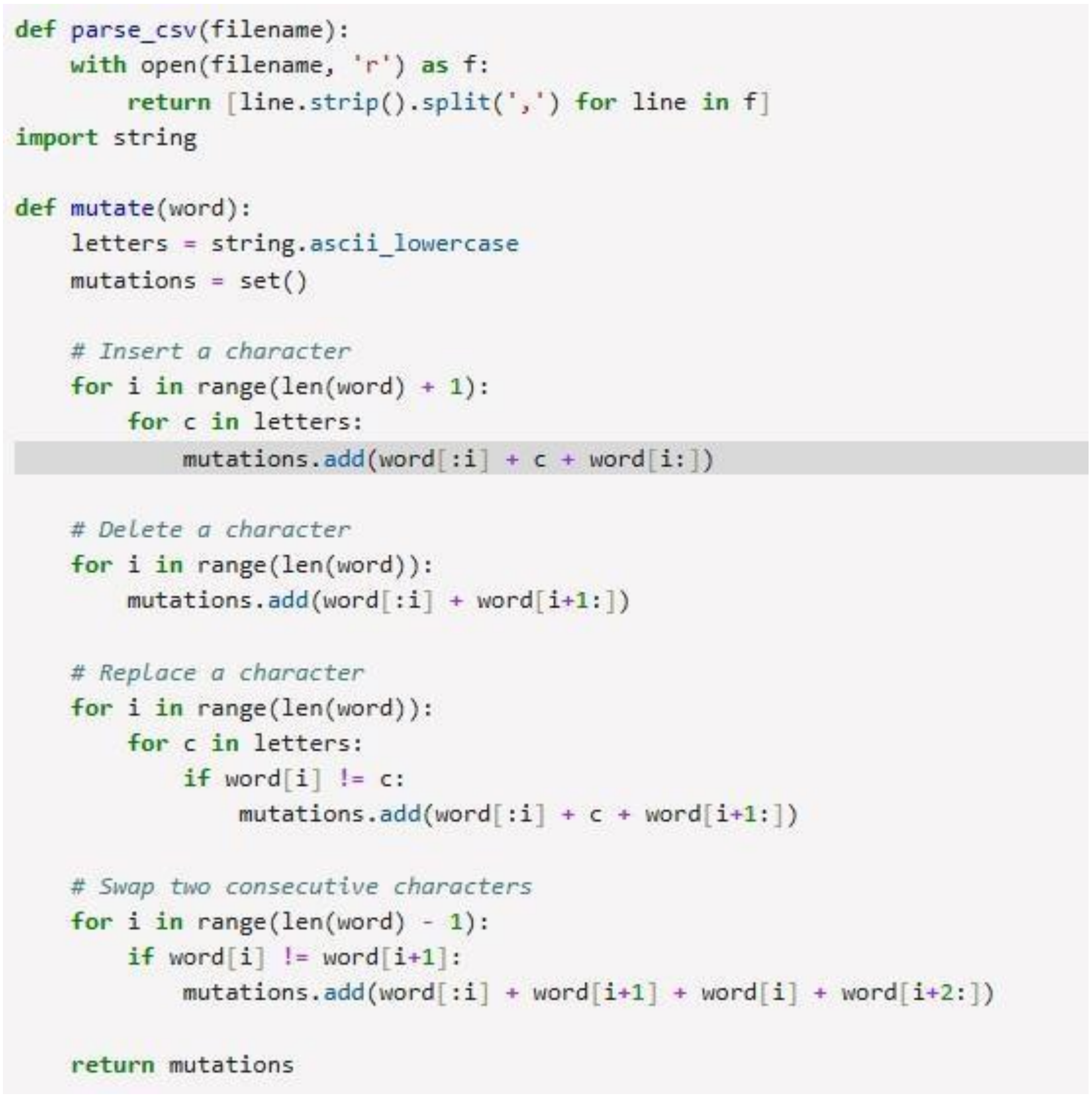
13.



```python
def triplet(n):
    return [(a, b, c)
            for c in range(n)
            for a in range(c+1)
            for b in range(a, c+1)
            if a + b == c]

# Example:
print(triplet(10))
```

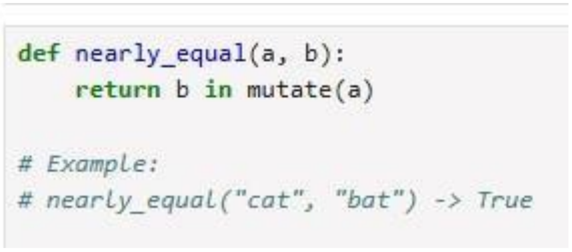
```
[(0, 0, 0), (0, 1, 1), (0, 2, 2), (1, 1, 2), (0, 3, 3), (1, 2, 3), (0, 4, 4), (1, 3, 4), (2, 2, 4), (0, 5, 5), (1, 4, 5), (2, 3, 5), (0, 6, 6), (1, 5, 6), (2,
4, 6), (3, 3, 6), (0, 7, 7), (1, 6, 7), (2, 5, 7), (3, 4, 7), (0, 8, 8), (1, 7, 8), (2, 6, 8), (3, 5, 8), (4, 4, 8), (0, 9, 9), (1, 8, 9), (2, 7, 9), (3, 6,
9), (4, 5, 9)]
```

14.



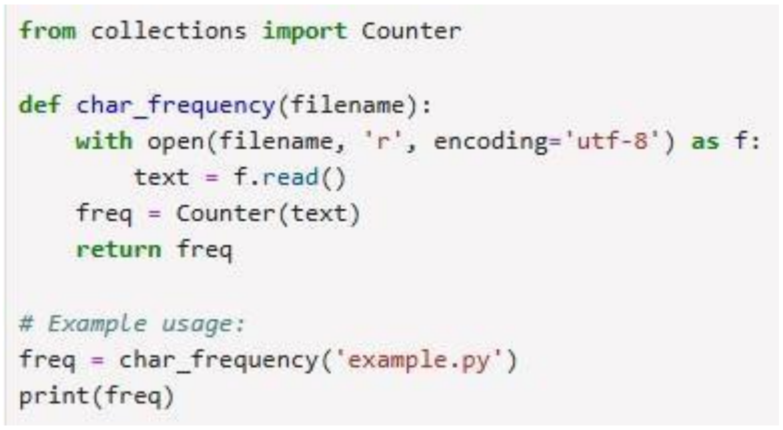
```python
def parse_csv(filename):
    with open(filename, 'r') as f:
        return [line.strip().split(',') for line in f]
import string

def mutate(word):
    letters = string.ascii_lowercase
    mutations = set()

    # Insert a character
    for i in range(len(word) + 1):
        for c in letters:
            mutations.add(word[:i] + c + word[i:])

    # Delete a character
    for i in range(len(word)):
        mutations.add(word[:i] + word[i+1:])

    # Replace a character
    for i in range(len(word)):
        for c in letters:
            if word[i] != c:
                mutations.add(word[:i] + c + word[i+1:])

    # Swap two consecutive characters
    for i in range(len(word) - 1):
        if word[i] != word[i+1]:
            mutations.add(word[:i] + word[i+1] + word[i] + word[i+2:])

    return mutations
```

15.



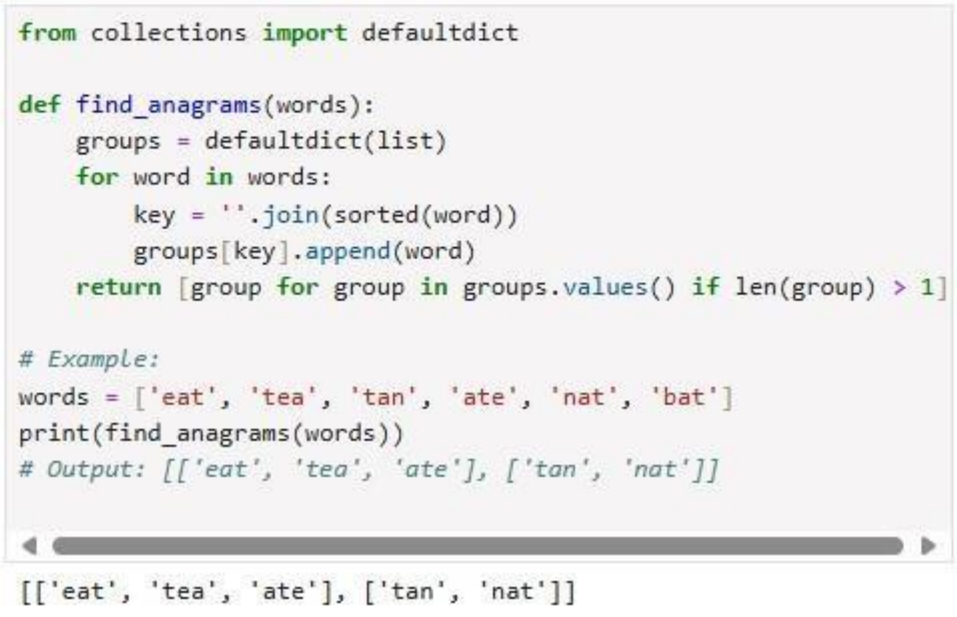
```python
def nearly_equal(a, b):
    return b in mutate(a)

# Example:
# nearly_equal("cat", "bat") -> True
```

16.



```python
from collections import Counter

def char_frequency(filename):
    with open(filename, 'r', encoding='utf-8') as f:
        text = f.read()
    freq = Counter(text)
    return freq

# Example usage:
freq = char_frequency('example.py')
print(freq)
```

17.



```python
from collections import defaultdict

def find_anagrams(words):
    groups = defaultdict(list)
    for word in words:
        key = ''.join(sorted(word))
        groups[key].append(word)
    return [group for group in groups.values() if len(group) > 1]

# Example:
words = ['eat', 'tea', 'tan', 'ate', 'nat', 'bat']
print(find_anagrams(words))
# Output: [['eat', 'tea', 'ate'], ['tan', 'nat']]
```

```
[['eat', 'tea', 'ate'], ['tan', 'nat']]
```