Homework No. 03

Due: 23:59, 07 May, 2023

Max points: 100

Rules

- **No late homeworks.** A penalty of 10 points is applied for each day.
- **No plagiarism.** Collaboration is encouraged, but copying someone else's work without proper attribution is not admitted and invalidates the submission. A penalty is applied to all parties included.

Submission procedure

- Each problem solution should be saved in a separate file. The following naming convention should be used: `problem{number}.{extension}`. For example, `problem1.py` or `problem1.pdf`.
- At the start of each file, homework number, student full name and problem number should be mentioned. For example:

```
"""
Homework 3
Name: John Doe
Problem 1
"""
```

- Solution files should be uploaded to YSU Moodle. Alternatively, you can commit your solutions to a Git repository and provide the repository URL on Moodle.

Problem 1 [10 points]

Two strings or sequences in general are considered anagrams if we can rearrange the letters / elements of one to obtain the other. Write a Python function to check if two sequences are anagrams or not.

Example

```python
def are_anagrams(seq1, seq2):
    pass

print(are_anagrams("Tom Marvolo Riddle", "I am Lord Voldemort")) # True
print(are_anagrams("1234", "2140")) # False
```

Problem 2 [10 points]

Write a Python function to remove duplicate values from a dictionary.

Example

```python
def remove_duplicates(dictionary):
    pass

print(remove_duplicates({"a": 1, "b": 1, "c": 2})) # {"a": 1, "c": 2}
```

Problem 3 [10 points]

Write a Python function to get the common elements from two dictionaries.

Example

```python
def get_common_elements(dictionary1, dictionary2):
    pass

print(get_common_elements({"a": 1, "b": 1, "c": 2, "d": 3}, {"a": 1, "b": 2, "c": 2, "e": 5
```

Problem 4 [10 points]

Write a Python recursive function to generate all possible combinations of a set of elements.

Note: This will be your implementation of `itertools.combinations` function.
Note: It is not required, but this function can be a generator function.

Example

```python
def generate_combinations(elements, k):
    pass

elements = [1, 2, 3, 4]
k = 3
combinations = generate_combinations(elements, k)
print(combinations) # [(1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4)]
```

Problem 5 [10 points]

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For example, 6 is a perfect number.

Write a Python generator function that generates all the perfect numbers up to a given limit.

Example

```python
def generate_perfect_numbers(limit):
    pass

for num in perfect_numbers(100):
    print(num, end=" ") # 6 28
```

Problem 6 [10 points]

An Armstrong number is a number that is the sum of its own digits each raised to the power of the number of digits. For example, 153 is an Armstrong number as $153 = 1^3 + 5^3 + 3^3$.

Write a Python generator function that generates all the Armstrong numbers up to a given limit.

Example

```python
def generate_armstrong_numbers(limit):
    pass

for num in generate_armstrong_numbers(1000):
    print(num, end=" ") # 1 2 3 4 5 6 7 8 9 153 370 371 407
```

Problem 7 [10 points]

Note: The following problem should be used using generator functions in the Python standard library.

Write a Python function that takes a list of numbers and returns a list of all the triples of numbers in the list that form a Pythagorean triplet.

Example

```python
def pythagorean_triplets(numbers):
    pass

print(pythagorean_triplets([3, 4, 5, 6, 7, 8, 9, 10])) # [(3, 4, 5), (6, 8, 10)]
```

Problem 8 [10 points]

Note: The following problem should be used using generator functions in the Python standard library.

Write a Python function that takes a list of numbers and returns a list of all the subsets of numbers in the list that add up to a given target.

Example

```python
def find_subsets(numbers, target):
    pass

print(find_subsets([1, 2, 3, 4, 5], 7)) # [(2, 5), (3, 4), (1, 2, 4)]
```

Problem 9 [10 points]

Write a Python decorator function that caches the output of a function. It should return the cached value if the function is called again with the same arguments. Provide an example usage of the decorator.

Example

```python
def cache(func):
    pass

@cache
def fibonacci(n):
    pass

print(fibonacci(10)) # 55
print(fibonacci(10)) # 55 (this is a cached value)
```

Problem 10 [10 points]

Write a Python decorator function that limits the number of times a function can be called. Provide an example usage of the decorator.

Example

```python
def limit_calls(max_calls):
    pass

@limit_calls(3)
def greet():
    print("Hello world!")

greet() # Hello world!
greet() # Hello world!
greet() # Hello world!
greet() # Function `greet` can only be called 3 times.
```