Лабораторная работа №4:

Комплект 1: Алгоритмы на Python. Начало.

1.1: Написать функцию two_sum, которая возвращает кортеж из двух индексов элементов списка lst, таких что сумма элементов по этим индексам равна переменной target, Элемент по индексу может быть выбран лишь единожды, значения в списке могут повторяться. Если в списке встречается больше чем два индекса, подходящих под условие - вернуть наименьшие из всех. Элементы находятся в списке в произвольном порядке. Алгоритм на двух циклах, сложность O(n 2). Пример использования:

lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]

target = 8

result = two_sum(lst, target)

print(result)

Результат:

(0, 6)

Код программы:

1.1.py > ...

```python
def two_sum(lst, target):
    pairs = (len(lst), len(lst))
    for i in range(len(lst)):
        for j in range(len(lst)):
            if (lst[i] + lst[j] == target and (i, j) < pairs):
                pairs = (i, j)
    return pairs

lst = list(range(1,10))
target = 8
result = two_sum(lst, target)
print(result)
```

Результат:

```
(0, 6)
```

1.2: Усовершенствуйте предыдущую задачу ??, добавив функцию two_sum_hashed(lst, target) так, чтобы сложность алгоритма была ниже: O(n) или O(n · log(n)).

Код программы:

```python
def two_sum(lst, target):
    pairs = (len(lst), len(lst))
    for i in range(len(lst)):
        if ((target - lst[i]) in lst and (i, lst.index(target - lst[i])) < pairs):
            pairs = (i, lst.index(target - lst[i]))
    return pairs

lst = list(range(1,10))
target = 8
result = two_sum(lst, target)
print(result)
```

(variable) target: Literal[8]

Результат:

```
(0, 6)
```

1.3: Усовершенствуйте предыдущую задачу 1.2, добавив функцию , которая возвращает все наборы индексов, удовлетворяющих условию суммы target. Пример использования:

lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]

target = 8

result = two_sum_hashed_all(lst, target)

print(result)

Результат:

[(0,6), (1,5), (2,4)]

Код программы:

```python
from random import shuffle

def two_sum(lst, target):
    pairs = (len(lst), len(lst))
    for i in range(len(lst)):
        if ((target - lst[i]) in lst and (i, lst.index(target - lst[i])) < pairs):
            pairs = (i, lst.index(target - lst[i]))
    return pairs

def two_sum_hashed_all(lst, target):
    pairs = []
    for i in range(int(len(lst)/2)):
        if ((target - lst[i]) in lst and i != lst.index(target - lst[i]) and (lst.index(target - lst[i]), i) not in pairs):
            pairs.append((i, lst.index(target - lst[i])))
    return pairs

lst = list(range(1,100))
target = 8
result = two_sum_hashed_all(lst, target)
print(result)
```

Результат:

```
[(0, 6), (1, 5), (2, 4)]
```

1.4: Повторите или изучите понятие мемоизации в Python. Реализуйте с помощью мемоизации и рекурсии вычисление чисел Фибоначчи сначала рукаки с помощью вручную добавленого к рекурсивной функции словаря с ранее вычисленными числами Фибоначчи, а затем с помощью декоратора @cache из стандартного модуля Python functools.

Код программы:

1.4.py > ...

```python
from functools import cache
import time

@cache
def fibonacci_cache(n):
    if n <= 1:
        return n
    return fibonacci_cache(n-1) + fibonacci_cache(n-2)


my_dict = {0: 0, 1: 1}
def fibonacci_dict(n):
    if n not in my_dict:
        my_dict[n] = fibonacci_dict(n-1) + fibonacci_dict(n-2)
    return my_dict[n]


start = time.time()
result = fibonacci_dict(30)
time_taken_dict = time.time() - start
print(f"dict result:{result}, time {time_taken_dict:.6f}")


start = time.time()
result = fibonacci_cache(30)
time_taken_cache = time.time() - start
print(f"cache result:{result}, time {time_taken_cache:.6f}")
```

Результат:

```
dict result:832040, time 0.000008
cache result:832040, time 0.000028
```

Комплект 2: Начало использования библиотечных модулей.

2.1: Отправка почты через smtplib.

Код программы:

Не выполнил

Результат:

2.2: Парсинг сайта погоды (wheather HTML parsing) на google.com и/или на простом сайте wttr□in с помощью BeautifulSoup (v4).

Код программы:

2.2.py > ...

```python
import requests
from bs4 import BeautifulSoup

url = "https://yandex.com.am/weather"
response = requests.get(url)

bs = BeautifulSoup(response.text,"lxml")
temp = bs.find('span', 'temp__value temp__value_with-unit')
summary = bs.find('div', 'title-icon__text');
print(f"Температура: {temp.text}")
print(f"{summary.text}")
```

Результат:

```
Температура: 0
По данным гидрометцентра России: гололедица
```

2.3: С помощью бибилиотеки matplotlib вывести два окна с графиками функций по личному выбору. В одном окне два графика двух разных функций. В другом окне - один график ещё одной функции.

Код программы:

2.3.py > get_weather_data

```python
import matplotlib.pyplot as plt
import numpy as np
import requests

def f1(x):
    return 1/x

def f2(x):
    return np.cos(x)

def get_weather_data(format):
    url = f"https://wttr.in/?format={format}"
    try:
        response = requests.get(url)
        response.raise_for_status()
        return response.json()
    except requests.RequestException as e:
        print(f"Ошибка при запросе к серверу: {e}")
        return None


fig1, ax1 = plt.subplots(figsize=(10, 6))
x = np.linspace(-np.pi, np.pi, 400)
ax1.plot(x, f1(x), label='y = 1/x')
ax1.plot(x, f2(x), label='y = cos(x)')
ax1.set_title('Графики двух функций')
ax1.legend()
```

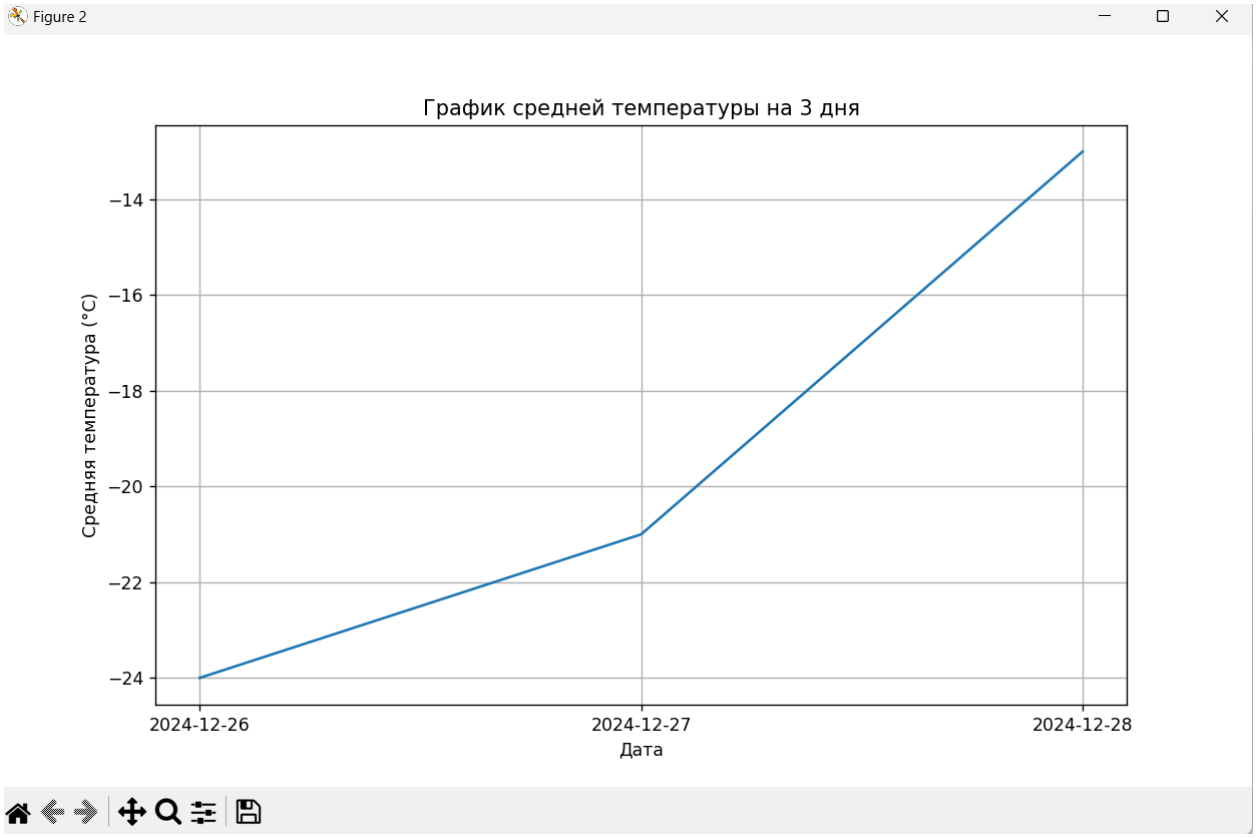
```python
weather_data= get_weather_data("j2")
dates = list()
temps = list()

for weather_info in weather_data['weather']:
    dates.append(weather_info['date'])
    temps.append(int(weather_info['avgtempC']))

fig2, ax2 = plt.subplots(figsize=(10, 6))
ax2.plot(dates, temps)
ax2.set_title('График средней температуры на 3 дня')
ax2.set_xlabel('Дата')
ax2.set_ylabel('Средняя температура (°C)')
ax2.grid(True)

plt.show()
```

Результат:



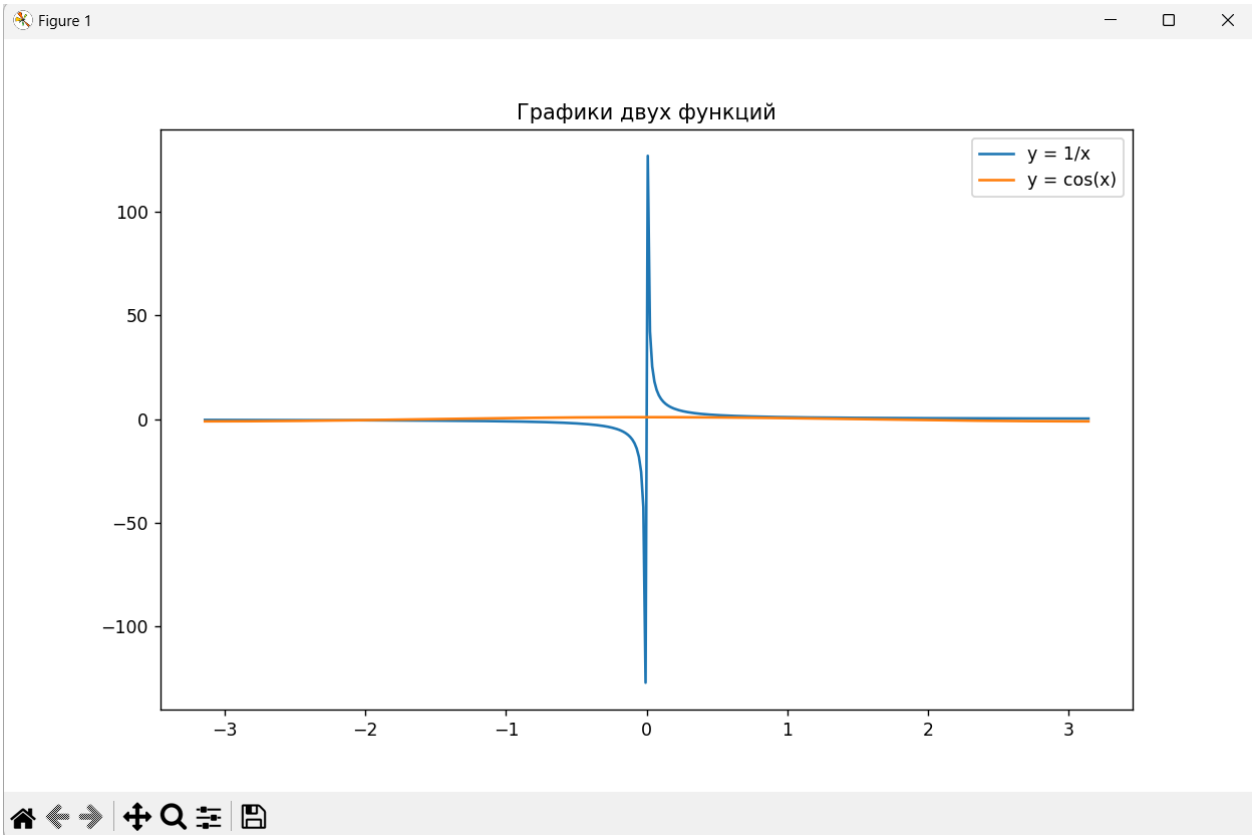


Figure 1
Графики двух функций
y = 1/x
y = cos(x)
100
50
0
−50
−100
−3
−2
−1
0
1
2
3