Лабораторна робота №2
Основи web – програмування.
Короткі теоретичні відомості.

1. Поняття маски та поділ IP-адреси на адресу мережі та адресу вузла.

Для IP-адрес виділяється 4 байти і адреса записується у вигляді чотирьох чисел, що представляють значення кожного байта в десятковій формі, які розділених точками. Комп'ютер насправді працює з двійковими представленнями адрес. Наприклад, 215.17.125.177 - десяткова форма представлення адреси, а 11010111.00010001.01111101.10110001 - двійкова форма представлення цієї адреси.

IP-адреса — унікальна мережева адреса вузла в мережі, що побудована на основі стеку протоколів TCP/IP.

Вузлу в мережі може бути призначена будь яка IP-адреса, але якщо призначена адреса суперечить прийнятим домовленостям то це призведе до проблем у роботі мережі. Згідно RFC3330 адреси мають наступні призначення
Special-Use IPv4 Addresses

Address Block	Present Use	Reference
0.0.0.0/8	"This" Network	[RFC1700, page 4]
10.0.0.0/8	Private-Use Networks	[RFC1918]
14.0.0.0/8	Public-Data Networks	[RFC1700, page 181]
24.0.0.0/8	Cable Television Networks	--
39.0.0.0/8	Reserved but subject to allocation	[RFC1797]
127.0.0.0/8	Loopback	[RFC1700, page 5]
128.0.0.0/16	Reserved but subject to allocation	--
169.254.0.0/16	Link Local	--
172.16.0.0/12	Private-Use Networks	[RFC1918]
191.255.0.0/16	Reserved but subject to allocation	--
192.0.0.0/24	Reserved but subject to allocation	--
192.0.2.0/24	Test-Net	
192.88.99.0/24	6to4 Relay Anycast	[RFC3068]
192.168.0.0/16	Private-Use Networks	[RFC1918]
198.18.0.0/15	Network Interconnect Device Benchmark Testing	[RFC2544]
223.255.255.0/24	Reserved but subject to allocation	--
224.0.0.0/4	Multicast	[RFC3171]
240.0.0.0/4	Reserved for Future Use	[RFC1700, page 4]

Деколи вказують що IP-адреса належить до певного класу. Можна знайти наступну інформацію про класи IP-адрес, хоча у справжніх мережах ця інформація використовується рідко.

Клас A: 1.0.0.0—126.0.0.0, маска 255.0.0.0
Клас B: 128.0.0.0—191.255.0.0, маска 255.255.0.0
Клас C: 192.0.0.0—223.255.255.0, маска 255.255.255.0
Клас D: 224.0.0.0—239.255.255.255, маска 255.255.255.255
Клас E: 240.0.0.0—247.255.255.255, маска 255.255.255.255

У попередніх таблицях та у налаштуваннях мережевих інтерфейсів можна звернути увагу на ще одне 4-х байтне число – маску. В першій таблиці маска вказана як префікс — кількість значущих бітів у 4-х байтовому числі. В другій як 4-х байтне число де кожен байт представлений десятковим числом.

Насправді, IP - адреса складається із двох логічних частин — номера мережі й номери вузла в мережі. Яка частина адреси належить до номера мережі, а яка — до номера вузла, визначається

маскою. Маска — це число (бітова маска), що використовується в парі з IP-адресою; двійковий запис маски містить одиниці в тих розрядах, які повинні в IP-адресі інтерпретуватися як номер мережі. Оскільки номер мережі це нерозривна частина адреси, одиниці в масці повинні становити безперервну послідовність. Наприклад, часто використовуються маски з наступними значеннями:

255. 0.0.0 - 11111111. 00000000. 00000000. 00000000;
255.255.0.0 - 11111111. 11111111. 00000000. 00000000;
255.255.255.0 - 11111111. 11111111. 11111111. 00000000.

Мережа це та частина IP адреси, яка не змінюється у всій мережі й всі адреси пристроїв починаються з цього номера мережі. Вузол – це змінна частина IP адреси. Кожен пристрій має свою унікальну адресу в мережі й це адреса вузла (ідентифікатор вузла).

Для визначення номера мережі та номера вузла потрібно виконати додаткові обчислення. Для обчислення номера мережі за заданою IP-адресою та маскою необхідно виконати побітове “І” (AND) між адресою та маскою. Таку операцію називають накладанням маски на адресу. Для обчислення номера вузла за заданою IP-адресою та маскою необхідно виконати “І” (AND) між адресою та результатом виконання побітового “НІ” (NOT) до маски.

Наприклад, якщо IP адреса 215.17.125.177 а маска 255.255.255.240, то в двійковій формі це:.

IP-адреса: 215.017.125.177 (11010111.00010001.01111101.10110001)
Маска: 255.255.255.240 (11111111.11111111.11111111.11110000)
Номер мережі: 215.17.125.176 (11010111.00010001.01111101.10110000)
Номер вузла: 0.0.0.1 (00000000.00000000.00000000.00000001)

Якщо IP адреса 67.38.173.245 а маска 255.255.240.0, то в двійковій формі це;

IP-адреса: 67.038.173.245 (01000011.00100110.10101101.11110101)
Маска: 255.255.240.0 (11111111.11111111.11110000.00000000)
Номер мережі: 67.38.160.0 (01000011.00100110.10100000.00000000)
Номер вузла: 0.0.13.245 (00000000.00000000.00001101.11110101)

Кількість вузлів в мережі, а точніше в підмережі визначається як 2^{32-N}-2, де N — довжина маски (кількість значущих бітів). Чим довша маска, тем менше в підмережі вузлів.

Маска підмережі		Розмір ідентифікатора вузла	Максимальна кількість вузлів	
8 біт	255.0.0.0	24 біт	$2^{24} – 2$	16777214
16 біт	255.255.0.0	16 біт	$2^{16} – 2$	65534
24 біт	255.255.255.0	8 біт	$2^{8} – 2$	254
29 біт	255.255.255.248	3 біт	$2^{3} – 2$	6

При обчисленні максимальної кількості вузлів обчислюється скільки доступних IP-адрес є в мережі. Це число зменшується на 2 бо адреса мережі не присвоюється вузлам. Так само широкомовна адреса (broadcast) також не використовується як адреса вузла. Широкомовна адреса (broadcast) мережі сприймається кожним вузлом, як додатка власна адреса й пакет на цю адресу отримають усі вузли мережі, так як би вони були адресовані їм особисто. Широкомовна адреса (broadcast) мережі обчислюється згідно наступного виразу:

широкомовна_адреса_мережі = IP_будь-якого_комп'ютера_цієї_мережі OR NOT (MASK) (https://en.wikipeda.orig/wiki/Broadcast_address)

Для того щоб префіксовану форму маски перевести у маску підмережі можна скористатися наступним способом (вартує його уважно дослідити):

```
"""
x << y
    Returns x with the bits shifted to the left by y places
    (and new bits on the right-hand-side are zeros).
    This is the same as multiplying x by 2**y.
"""

prefix = 32
mask = [0, 0, 0, 0]

for i in range(prefix):
    mask[i // 8] += 1 << (7 - i % 8)

print(mask)
```

Наступний приклад містить всі обчислення які потрібно буде реалізувати під час виконання лабораторної роботи:

IP у десятковому вигляді: 172.20.0.0/14
IP у двійковому вигляді: 10101100.00010100.00000000.00000000
маска підмережі: 11111111.11111100.00000000.00000000 (255.252.0.0)
---------- [Logical AND (&)]
адреса мережі: 10101100.00010100.00000000.00000000 ---> 172.20.0.0

маска вузла: 00000000.00000011.11111111.11111111 (0.3.255.255)
(інвертована маска мережі)
IP у двійковому вигляді: 10101100.00010100.00000000.00000000
(Усі біти вузла у ---------- [Force host bits]
IP адресі робимо одиницями)
широкомовна адреса: 10101100.00010111.11111111.11111111 -> 172.23.255.255

IP у двійковому вигляді: 10101100.00010100.00000000.00000000
маска вузла: 00000000.00000011.11111111.11111111 (0.3.255.255)
---------- [Logical AND (&)]
адреса вузла: 00000000.00000000.00000000.00000000 -> 0.0.0.0

адреса мережі: 172.20.0.0
адреса вузла: 0.0.0.0
широкомовна адреса: 172.23.255.255
кількість біт, які відповідають за адресу вузла: 32-14=18
загальна кількість вузлів у мережі: 2**18-2 = 262144-2=262142

За наступним посиланням можна знайти прості формули для вищеперелічених обчислень, але потрібно контролювати чи ці спрощення не призводять до помилок (https://www.geeksforgeeks.org/ipv4-classless-subnet-equation/).

2. HTML.

HTML (*HyperText Markup Language* — **Мова розмітки гіпертекстових документів**) — стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.
HTML разом із каскадними таблицями стилів (CSS) та вбудованими скриптами (мова Python також належить до скриптових мов) — це три основні технології побудови веб-сторінок.
HTML надає засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- додавання зображень, звуку, відео, та інших об'єктів до тексту.

3. CSS.

Каскадні таблиці стилів (*Cascading Style Sheets* або скорочено **CSS**) — спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних. Найчастіше CSS використовують для візуальної презентації сторінок, створених за допомогою HTML та XHTML, але формат **CSS** може застосовуватися до інших видів XML-документів.

Більше інформації та пояснення до прикладів простих веб-сторінок можна знайти у лекції з курсу CS50 (https://prometheus.org.ua/cs50/week7w.html). Конспект лекції у доданому файлі Лекція 7-2.html.

4. Побудова web – карти.

Для побудови карти у веб браузері потрібно створити html файл, який буде містити карту. Побудову карти можна здійснити за допомогою бібліотек pandas та folium.

pandas — програмна бібліотека, написана на мові програмування Python для маніпулювання даними та їхнього аналізу.

folium – бібліотека, яка дозволяє будувати інтерактивні карти з використанням Python та бібліотеки Leaflet.js.

Для роботи з даними використовується Python, а їх візуалізація здійснюється в Leaflet.js за допомогою folium.

Бібліотеки pandas та folium не входять у стандартну бібліотеку Python і їх потрібно додатково встановити за допомогою програми pip.

```
pip install pandas
pip install folium
```

Розглянемо приклад такої карти (файл map_volcanoes.html). На карті зображено земну кулю, межі країн, населення країн та вулкани США. Карта інтерактивна і можна змінювати її масштаб та вимикати/вмикати шари цієї карти (власне карта, населення, вулкани).

Для побудови web - карти за допомогою folium потрібно виконати два основні кроки:
1. Створити об’єкт – карту.
2. Перетворити цей об’єкт в html файл.

Для створення об’єкту - карти потрібно скористатися класом Map бібліотеки folium.

```
import folium
map = folium.Map()
```

Для перетворення об’єкту в html файл потрібно скористатися методом save() цього класу.

```
map.save('Map_1.html')
```

В результаті отримано html файл. Якщо відкрити цей файл у браузері то можна побачити що це карта, яку можна масштабувати. Іншими словами це карта, яка містить єдиний шар і цей шар це карта з проекту OpenStreetMap. Якщо потрібно щоб цим базовим шаром була інша карта то потрібно при створенні карти вказати відповідне значення для параметра tiles. Наприклад,

```
map = folium.Map(tiles="Mapbox Bright")
```

Для зручності карту можна сформувати так, що при її відкритті на екрані в центрі карти буде певне місце і карта буде зображатися у певному початковому масштабі. Ці та інші параметри можна вказати, як параметри при створенні карти. Повний перелік параметрів доступний в документації та файлі допомоги.

```
map = folium.Map(tiles="Mapbox Bright", location=[49.817545, 24.023932], zoom_start=17)
```

Карта, яку буде створено з цими значеннями параметрів location та zoom_start буде показувати місце де ми зараз знаходимось (академічний корпус на вал. Козельницькій).
Додавання розширеної інформації для її зображення на карті можна здійснювати різними способами. Додавати інформацію безпосередньо до цієї базової карти можна, наприклад за допомогою методу add_child.

```
map.add_child(folium.Marker(location=[49.817545, 24.023932],
                            popup="Хіба я тут!",
                            icon=folium.Icon()))
```

На карті з’явився маркер з написом, який з’являється при наведенні курсора мишки на об'єкт і може зникати. Такий самий результат можна отримати якщо скористатися класом FeatureGroup і саме цьому способу потрібно надавати перевагу, якщо потрібно щоб інформація на карті зображалася на декількох шарах (складалася з декількох шарів). Якщо повторити у програмі цей рядок декілька разів то на карті будуть зображатися декілька маркерів. Якщо створити список координат точок, які потрібно позначити маркером то простий цикл дозволить уникнути дублювання однакових рядків у програмі. Якщо потрібно позначити на карті багато точок то доцільно зберегти координати цих точок у файлі й використовувати при потребі.

Наступний приклад містить фрагмент програми для обробки та зображення на карті інформації про декілька населених пунктів Івано-франківської області. У файлі Stan_1900.csv знаходиться інформація про три населені пункти де вказано рік, назва населеного пункту, назви церков, склад населення, інформація про школу та вказані координати населеного пункту (інформація взята з шематизму Станіславської єпархії за 1900 р.). Бібліотека pandas надає засоби для доступу до цих даних шляхом створення об’єкту типу DataFrame, який буде містити всю інформацію з csv файлу, а до фрагментів цієї інформації можна отримати доступ за назвою стовпчика. Працювати з csv файлами можна також за допомогою засобів модуля csv стандартної бібліотеки, але бібліотека pandas зараз набула більшого поширення.

```
import folium
import pandas
data = pandas.read_csv("Stan_1900.csv", error_bad_lines=False)
lat = data['lat']
lon = data['lon']
map = folium.Map(location=[48.314775, 25.082925],
                 zoom_start=10)
fg = folium.FeatureGroup(name="Kosiv map")
for lt, ln in zip(lat, lon):
  fg.add_child(folium.Marker(location=[lt, ln],
                             popup="1900 рік"
                             icon=folium.Icon()))
map.add_child(fg)
map.save('Map_5.html')
```

За потреби у вікні popup можуть бути показані будь які дані csv файлу. Наприклад,

```
churches = data['церкви']
for lt, ln, ch in zip(lat, lon, churches):
  fg.add_child(folium.Marker(location=[lt, ln],
                             popup="1900 рік"+ ch,
                             icon=folium.Icon()))
```

в цьому випадку на карті буде інформація про церкви в цих населених пунктах.
У бібліотеці folium реалізовані й інші засоби за допомогою яких на карті можна використовувати позначення різного типу, форми та кольору. Наступний фрагмент демонструє як зобразити на карті позначення населених пунктів не за допомогою стандартних маркерів, а кружків різного кольору, який визначається в залежності від кількості населення. Для цього використовується інший клас CircleMarker та відповідні значення для його параметрів.

```
import folium
import pandas
data = pandas.read_csv("Stan_1900.csv")
lat = data['lat']
lon = data['lon']
churches = data['церкви']
hc = data['гр-кат.']

def color_creator(population):
  if population < 2000:
    return "green"
  elif 2000 <= population <= 3500:
    return "yellow"
  else:
    return "red"

map = folium.Map(location=[48.314775, 25.082925],
                 zoom_start=10)
fg = folium.FeatureGroup(name="Kosiv_map")
for lt, ln, ch, hc in zip(lat, lon, churches, hc):
  fg.add_child(folium.CircleMarker(location=[lt, ln],
                                   radius=10,
                                   popup="1900 рік"+"\n" + ch,
                                   fill_color=color_creator(hc),
                                   color='red',
                                   fill_opacity=0.5))
map.add_child(fg)
map.save('Map_6.html')
```

Карта може складатися з довільної кількості шарів. Кожен шар можна створювати окремо і різними способами. Наступний приклад демонструє додавання до карти ще одного шару на основі геоінформації про країни та населення земної кулі. Інформація про країни та населення знаходиться у файлі world.json в форматі json (geojson). Для додавання цих даних як окремого шару до карти потрібно скористатися класом GeoJson з відповідними параметрами. Серед параметрів потрібно звернути увагу на style_function за допомогою якого країни з різною кількістю населення зображаються різними кольорами. Для керування шарами карти потрібно створити і додати до карти ще один елемент LayerControl.

```
map = folium.Map(location=[48.314775, 25.082925],
          zoom_start=10)
fg_hc = folium.FeatureGroup(name="Greek Catholic")
for lt, ln, ch, hc in zip(lat, lon, churches, hc):
  fg_hc.add_child(folium.CircleMarker(location=[lt, ln],
                        radius=10,
                        popup="1900 рік"+"\n" + ch,
                        fill_color=color_creator(hc),
                        color='red',
                        fill_opacity=0.5))
fg_pp = folium.FeatureGroup(name="Population")
fg_pp.add_child(folium.GeoJson(data=open('world.json', 'r',
                encoding='utf-8-sig').read(),
```

```
                style_function=lambda x: {'fillColor':'green'
  if x['properties']['POP2005'] < 10000000
  else 'orange' if 10000000 <= x['properties']['POP2005'] < 20000000
  else 'red'}))
map.add_child(fg_hc)
map.add_child(fg_pp)
map.add_child(folium.LayerControl())
map.save('Map_7.html')
```

В попередніх прикладах для розміщення на карті маркерів використовуються координати (широта, довгота) населених пунктів, які вказані у файлах. У випадку якщо б координати були відсутні було би потрібно їх визначити. Для визначення координат можна скористатися спеціалізованими сервісами наприклад google map, або однією з багатьох Python бібліотек, які розроблені з цією метою.

Бібліотека geopy дозволяє встановити координати населених пунктів місцевостей та окремих адрес. Для використання цієї бібліотеки її потрібно додатково встановити, або якщо її назву вказати у файлі requirements.txt проекту то процес встановлення автоматично виконає PyCharm. Для отримання координат та іншої додаткової інформації потрібно виконати наступні дії.

```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="specify_your_app_name_here")
location = geolocator.geocode("Старі Кути")
print(location.address)
print((location.latitude, location.longitude))
print(location.raw)
```

В результаті буде отримано:
Старі Кути, Косівський район, Івано-Франківська область, Україна
(48.287312, 25.1738)
{'place_id': '242217085', 'licence': 'Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright', 'osm_type': 'relation', 'osm_id': '8839103', 'boundingbox': ['48.26637', '48.2921912', '25.13844', '25.1852525'], 'lat': '48.287312', 'lon': '25.1738', 'display_name': 'Старі Кути, Косівський район, Івано-Франківська область, Україна', 'class': 'boundary', 'type': 'administrative', 'importance': 0.465228110551882, 'icon': 'https://nominatim.openstreetmap.org/images/mapicons/poi_boundary_administrative.p.20.png'}

Якщо потрібно отримати координати великої кількості локацій то можливе виникнення помилки `Too Many Requests 429 error.` Для її уникнення потрібно використовувати клас RateLimiter, який дозволяє керувати процесом викликів з відповідними параметрами.

```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="specify_your_app_name_here")
from geopy.extra.rate_limiter import RateLimiter
geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1)

for point in ["Львів","Старі Кути","Кути","Брустурів"]:
  location = geolocator.geocode(point)
  print(location.address)
  print((location.latitude, location.longitude))
```

Виклики `geolocator.geocode` будуть виконуватися із затримкою `min_delay_seconds`, а при виникненні винятків виклики будуть виконуватися до завершення вказаного проміжку часу (`max_retries`).

Література:
1. https://www.geeksforgeeks.org/ipv4-classless-subnet-equation/
2. https://www.calculator.net/ip-subnet-calculator.html/
3. Курс CS50
4. Курс The Python Mega Course: Build 10 Real World Applications.