

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет
«Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з дисципліни
“Розробка прикладних програм”
для студентів спеціальностей
121 “Інженерія програмного забезпечення” та
122 “Комп'ютерні науки”
(денної форми навчання)

2023

Методичні вказівки до виконання лабораторних робіт з дисципліни
“Розробка прикладних програм” для студентів спеціальностей 121
“Інженерія програмного забезпечення” та 122 “Комп'ютерні науки”
(денної форми навчання) / В.М. Льовкін. – Запоріжжя : НУ
«Запорізька політехніка», 2023. – 120 с.

Автор: В.М. Льовкін, канд. техн. наук, доцент

Рецензент: А.О. Олійник, д. т. н., професор

Відповіdalний
за випуск: С.О. Субботін, д. т. н., професор

Затверджено
на засіданні кафедри
“Програмні засоби”

Протокол № 8
від “30” березня 2023 р.

ЗМІСТ

Вступ	5
Лабораторна робота № 1	6
Розроблення програмного забезпечення з графічним інтерфейсом на основі об'єктно-орієнтованого програмування мовою Python..	6
1.1. Мета роботи.....	6
1.2. Короткі теоретичні відомості	6
1.2.1 Об'єктно-орієнтоване програмування у мові Python	6
1.2.2 Розроблення графічних інтерфейсів користувача програм мовою Python.....	9
1.3. Завдання на лабораторну роботу.....	19
1.4. Зміст звіту.....	22
1.5. Контрольні запитання.....	23
Лабораторна робота № 2	24
Використання бібліотек Python для високопродуктивних наукових обчислень	24
2.1. Мета роботи.....	24
2.2. Короткі теоретичні відомості	24
2.2.1 Бібліотека NumPy	24
2.2.2 Бібліотека SciPy	33
2.2.3 Бібліотека matplotlib	48
2.3. Завдання на лабораторну роботу.....	55
2.4. Зміст звіту.....	63
2.5. Контрольні запитання.....	63
Лабораторна робота № 3	65
Розроблення вебдодатків та реалізація доступу до систем керування базами даних через програмні інтерфейси	65
3.1. Мета роботи.....	65
3.2. Короткі теоретичні відомості	65
3.2.1 Програмний інтерфейс для роботи з системою керування базами даних MySQL	65
3.2.2 Фреймворк Django для розроблення вебдодатків.....	67
3.3. Завдання на лабораторну роботу.....	78
3.4. Зміст звіту.....	83
3.5. Контрольні запитання.....	83
Лабораторна робота № 4	84

Оброблення природньої мови	84
4.1. Мета роботи.....	84
4.2. Короткі теоретичні відомості	84
4.3. Завдання на лабораторну роботу.....	91
4.4. Зміст звіту.....	94
4.5. Контрольні запитання.....	94
Лабораторна робота № 5	95
Машинне навчання.....	95
5.1. Мета роботи.....	95
5.2. Короткі теоретичні відомості	95
5.2.1 Бібліотека Pandas	95
5.2.2 Бібліотека scikit-learn	100
5.3. Завдання на лабораторну роботу.....	111
5.4. Зміст звіту.....	115
5.5. Контрольні запитання.....	116
Література.....	118

ВСТУП

Дане видання призначене для отримання студентами денної форми навчання фундаментальних знань з конструювання та розроблення інтелектуально-орієнтованого програмного забезпечення на основі використання сучасних засобів програмування у різних прикладних галузях (оброблення природньої мови, наукові обчислення, розроблення вебдодатків) та практичного засвоєння отриманих знань.

Відповідно до графіка студенти перед виконанням лабораторної роботи повинні ознайомитися з конспектом лекцій та рекомендованою літературою. Дані методичні вказівки містять тільки основні, базові теоретичні відомості, необхідні для виконання лабораторних робіт, тому для виконання лабораторної роботи та при підготовці до її захисту необхідно ознайомитись з конспектом лекцій та опрацювати весь необхідний матеріал, наведений у переліку рекомендованої літератури, використовуючи також статті у інтернет-виданнях та актуальних наукових журналах з проблем штучного інтелекту.

Для одержання заліку з кожної роботи студент повинен у відповідності зі всіма наведеними вимогами розробити програмне забезпечення та оформити звіт, після чого продемонструвати на комп'ютері розроблене програмне забезпечення з виконанням усіх запропонованих викладачем тестів. Звіт виконують на білому папері формату А4. Текст розміщують з однієї сторони аркуша. Поля сторінки з усіх боків – 20 мм. Аркуші вміщують у канцелярський файл.

Усі завдання повинні виконуватись студентами індивідуально і не містити ознак піглату як в оформленому звіті так і в розробленому програмному забезпеченні. Під час співбесіди при захисті лабораторної роботи студент повинен виявити знання щодо мети роботи, теоретичного матеріалу, методів виконання кожного етапу роботи, змісту основних розділів звіту з демонстрацією результатів на конкретних прикладах, практичних прийомів використання теоретичного матеріалу в розробленому програмному забезпеченні. Студент повинен вміти обґрунтувати всі прийняті ним проектні рішення та правильно аналізувати і використовувати на практиці отримані результати. Для базової самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

ЛАБОРАТОРНА РОБОТА № 1

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ НА ОСНОВІ ОБ'ЄКТНО- ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ МОВОЮ PYTHON

1.1. Мета роботи

1.1.1 Ознайомитись з принципами реалізації об'єктно-орієнтованого програмування у мові Python та навчитись використовувати його для розроблення програмного забезпечення.

1.1.2 Навчитися розробляти сучасні графічні інтерфейси користувача для програм, написаних мовою Python.

1.2. Короткі теоретичні відомості

1.2.1 Об'єктно-орієнтоване програмування у мові Python

Для визначення класів у Python [9] використовується наступна загальна форма:

Python

```
class ClassName:  
    <statement-1>  
    .  
    .  
    .  
    <statement-N>
```

Приклад оголошення класу:

Python

```
class MyClass:  
    """Приклад класу"""""  
    i = 12345
```

Python

```
def f(self):
    return 'hello world'
```

Метод для створення екземпляра-об'єкта може оголошуватись наступним чином:

Python

```
def __init__(self):
    self.data = []
```

У такий метод може передаватися більша кількість параметрів, якщо це необхідно:

Python

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(1.5, -2.3)
```

Оголошення класу та роботу з атрибутами об'єктів можна представити наступним прикладом:

Python

```
class Dog:

    kind = 'canine'

    def __init__(self, name):
```

Python

```
self.name = name
```

```
>>> d = Dog('Chuck')
>>> e = Dog('Brovko')
>>> d.kind
'canine'
>>> e.kind
'canine'
>>> d.name
'Chuck'
>>> e.name
'Brovko'
```

Для організації наслідування використовується наступний синтаксис:

Python

```
class DerivedClassName(BaseClassName):
    <statement-1>
    .
    .
    .
    <statement-N>
```

Можливість зберігати об'єкти надають 3 модуля стандартної бібліотеки:

- pickle – перетворює довільні об'єкти в рядок байтів та навпаки;
- dbm – реалізує збереження рядків у файлах, забезпечуючи можливість звернення за ключем;
- shelve – використовує перші два модулі, дозволяючи зберігати об'єкти в файлах-сховищах, що забезпечують можливість звернення за ключем.

Приклад збереження об'єктів у сховище:

Python

```
import shelve
db = shelve.open('persondb')
for object in (bob, sue, tom):
    db[object.name] = object
db.close()
```

При підготовці матеріалів даного пункту використано документацію [19] (зокрема розділ [<https://docs.python.org/3/tutorial/classes.html>]).

1.2.2 Розроблення графічних інтерфейсів користувача програм мовою Python

`tkinter` – бібліотека для розроблення графічних інтерфейсів, що розповсюджується з відкритим вихідним кодом, яка стала стандартом для розроблення переносних графічних інтерфейсів мовою Python. Сценарії мовою Python, що використовують `tkinter` для побудови GUI, виконуються у Windows, Linux та Macintosh, створюючи на кожній з даних платформ графічний інтерфейс з властивим їм зовнішнім виглядом. Бібліотека може легко розширятись програмним кодом мовою Python, а також має багато додаткових пакетів: `Pmw` (стороння бібліотека віджетів), `Tix` (бібліотека віджетів, стандартна частина Python), `PIL` (розширення для оброблення зображень) і `ttk` (бібліотека віджетів Tk, що підтримує теми оформлення та стала стандартною частиною Python).

Існують і інші способи створення інтерфейсів програм, написаних мовою Python. Серед них зокрема і `PyQt` – інтерфейс Python до бібліотеки Qt.

Клас `Tk` є базовим і викликається за допомогою конструктора `tkinter.Tk(screenName=None, baseName=None, className='Tk', useTk=1)`, що створює віджет верхнього рівня, який зазвичай є головним вікном додатку.

`tkinter` є бібліотекою, орієнтованою на події, в яких є головний цикл оброблення подій. Для запуску такого циклу використовується метод `mainloop`, а для виходу – метод `quit`.

До віджетів бібліотеки `tkinter` належать:

- `Toplevel` – вікно верхнього рівня, що зазвичай використовується під час створення багаторівнісних програм;
- `Button` – кнопка;
- `Label` – мітка (напис без можливості редагування);
- `Entry` – віджет, що дозволяє ввести один рядок тексту;
- `Text` – віджет для введення багаторядкового тексту;
- `Listbox` – перелік, з якого можна обрати один або декілька елементів;
- `Frame` – рамка для організації віджетів всередині вікна;
- `Checkbutton` – віджет, що дозволяє вибрати деякий пункт у вікні;
- `Radiobutton` – дозволяє обрати тільки один пункт у вікні;
- `Scale` – віджет, що дозволяє вибрати значення з деякого діапазону;
- `Scrollbar` – віджет, що дозволяє прокручувати інший відмет;
- `Menu` – віджет для створення меню, що спливає або випадає;
- `Menubutton` – кнопка з меню;
- `Canvas` – основа для виведення графічних примітивів;
- `Message` – віджет аналогічний `Label`, що дозволяє розташовувати багаторядкові тексти;
- `Spinbox` – віджет, який дозволяє вибирати значення з заданої множини (поле та пара стрілок);
- `PanedWindow` – віджет, який дозволяє розділити простір та є контейнером для віджетів-нащадків;
- `LabelFrame` – прямокутна ділянка з написом, яка може містити інші віджети;
- `OptionMenu` – віджет, який надає фіксовану множину варіантів вибору у випадаючому меню.

Створення віджетів відбувається наступним чином (приклад для кнопки):

Python

```
w = tkinter.Button(parent, option=value, ...)
```

Перелік налаштувань для кнопки зокрема включає наступні:

- command – функція або метод, які викликаються, коли натискають на кнопку (використовується також для відповідних дій у наступних віджетах: Checkbutton, Radiobutton, Spinbox, Scrollbar, Scale);

- height – висота кнопки;
- width – ширина кнопки;
- text – текст, який відображається на кнопці;
- image – зображення, яке відображається на кнопці;
- bg – колір фону;
- fg – колір шрифту тексту на кнопці;
- font – шрифт тексту на кнопці.

Усі віджети мають наступні спільні методи:

- configure, config – метод, який використовується для конфігурування під час виконання програми;
- cget – дозволяє отримувати інформацію про конфірмацію віджета;

- destroy – видалення віджета та його нащадків;
- grab_* – сімейство методів для керування потоком подій (віджет, що захопив потік, отримує всі події вікна або додатку);
- focus_* – сімейство методів для керування фокусом введення з клавіатури (віджет, що має фокус, отримує всі події з клавіатури);
- after, after_idle i after_cancel – методи таймери, за допомогою яких можна відкласти виконання коду на деякий час;
- update i update_idletasks – методи для роботи з чергою задач, виконання яких викликає оброблення відкладених задач;

- winfo_* – сімейство методів, які повертають значення відповідних властивостей віджетів (winfo_children(), winfo_class(), winfo_height(), winfo_name(), winfo_parent(), winfo_id(), winfo_width()), winfo_x(), winfo_y()).

Віджет Toplevel має зокрема наступні методи:

- title(text=None) – встановлює або повертає заголовок вікна;

- iconify() – згорнути вікно;
- deiconify() – розгорнути вікно;
- withdraw() – сховати вікно;
- state(newState=None) – повертає або встановлює стан вікна (normal, iconic (згорнуте), withdrawn (сховане));
- resizable(width=None, height=None) – чи може користувач змінювати розмір вікна;
- geometry(newGeometry=None) – встановлює розташування та розмір вікна (рядок у вигляді ширинахвисота+x+y);
- transient(parent=None) – робить вікно залежним від іншого вікна.

У складі бібліотеки tkinter є набір готових діалогів, які забезпечуються трьома модулями:

- tkMessageBox забезпечує набір стандартних діалогів для простих завдань: askokcancel(title, message, options), askquestion(title, message, options), askretrycancel(title, message, options), askyesno(title, message, options), showerror(title, message, options), showinfo(title, message, options), showwarning(title, message, options);
- tkFileDialog дозволяє користувачу виконувати пошук файлів;
- tkColorChooser дозволяє користувачу вибрати колір.

Більш детально з повним переліком методів та властивостей кожного віджета можна зокрема ознайомитись у [25].

Для розміщення віджетів у вікні використовуються менеджери розташування (пакувальники). У Tkinter є 3 пакувальники: pack, place, grid.

Пакувальник pack() є найбільш актуальним. За допомогою властивості side ("left"/"right"/"top"/"bottom") можна вказати, до якої сторони батьківського віджета даний віджет повинен прилягати, а за допомогою властивості fill (None/"x"/"y"/"both") – чи необхідно розширювати простір, наданий віджету. Властивість expand (True/False) визначає, чи необхідно розширювати віджет, щоб він зайняв весь необхідний йому простір. Властивість in_ визначає, у який батьківський віджет помістити даний віджет.

Пакувальник grid() представляє собою таблицю з комірками, в яких розташовуються віджети. Має наступні аргументи:

- row – номер рядка, в якому необхідно розташувати віджет;
- rowspan – кількість рядків, яку займає віджет;

– column – номер стовпця, в якому необхідно розташувати віджет;

- columnspan – кількість стовпців, яку займає віджет;

- padx / pady – розмір зовнішнього бордюра;

- ipadx / ipady – розмір внутрішнього бордюра;

- sticky – визначає, яким чином розподілити зайвий простір;

- in_ – визначає, в який батьківський віджет помістити даний.

Додаткові функції для роботи з пакувальником grid включають зокрема grid_location (x, y), яка повертає номер рядка та стовпця, в які потрапляють задані координати, та grid_size (), яка повертає розмір сітки в рядках та стовпцях.

Найпростішим пакувальним є place(), який дозволяє розміщувати віджети у фіксованих місцях з фіксованим розміром. Має зокрема наступні аргументи:

- anchor – кут або сторона, відносно яких визначаються координати розташування віджета;

- in_ – визначає, в який батьківський віджет даний віджет необхідно розташувати;

- x і y – абсолютні координати розташування віджета;

- width і height – ширина та висота віджета;

- relx і rely – відносні координати (від 0.0 до 1.0) розташування віджета.

Ще одним способом прив'язування подій окрім параметра command є метод bind, який дозволяє прив'язувати подію до дії:

Python

```
tkinter.bind(sequence=None, func=None, add=None)
```

Аргумент sequence визначає подію, func – функцію, яка буде викликана за настання відповідної події, а add може дорівнювати '+', якщо прив'язка додається до існуючої.

Загальна форма події представляється наступним шаблоном:

Python

<[modifier-]...type[-detail]>

Опція *modifier* включає: Alt (має утримуватись клавіша alt), Any (узагальнює тип події), Control (має утримуватись клавіша control), Double (дві події, що трапляються поспіль), Lock (має бути натиснута клавіша shift), Shift (має утримуватись клавіша shift), Triple (три події, що трапляються поспіль).

Типи подій *type* включають Activate, Button, ButtonRelease, Configure (зміна положення або розміру віджета), Deactivate, Destroy, Enter (курсор входить у видиму частину віджета), Expose (вікно або його частина перемальовуються), FocusIn, FocusOut, KeyPress, KeyRelease, Leave (курсор виходить з вікна або віджета), Map (показ вікна), Motion, MouseWheel, Unmap (прибирання вікна), Visibility.

Опція *detail* описує конкретну клавішу (Alt_L, Alt_R, BackSpace, Cancel, Caps_Lock тощо) або кнопку миші (1, 2, 3).

Функція *func* повинна приймати один аргумент, об'єкт класу Event, в якому описано подію та яке приймає наступні атрибути.

У бібліотеку віджетів tk включені наступні віджети, які можна використовувати замість відповідних віджетів tk: Button, Checkbutton, Entry, Frame, Label, LabelFrame, Menubutton, PanedWindow, Radiobutton, Scale та Scrollbar.

У ttk включені також наступні нові віджети:

- Combobox – текстове поле з випадаючим переліком значень;
- Notebook – віджет, який керує колекцією вікон та відображає одне за раз. Будь-яке дочірнє вікно зв'язано з вкладкою, яку користувач може вибрати, щоб змінити вікно, яке відображається;
- Progressbar – віджет, який відображає статус довготривалих операцій;
- Separator – вертикальна або горизонтальна полоса розподілу;
- Sizegrip – віджет, який дозволяє користувачу змінити розмір вікна верхнього рівня;
- Treeview – ієрархічна колекція пунктів.

Усі ці класи віджетів є підкласами класу Widget.

Клас Widget має наступні методи:

- identify(x, y) – повертає назву елементу на заданій позиції;
- instate(statespec, callback=None, *args, **kw) – перевіряє стан віджета;

– state([statespec=None]) – змінює або читає стан віджета.

Для налаштування зовнішнього вигляду віджетів використовується клас Style. Основні методи даного класу:

- configure(style, query_opt=None, **kw) – встановлює налаштування зовнішнього вигляду віджетів;

Python

```
from tkinter import ttk
import tkinter

root = tkinter.Tk()
ttk.Style().configure("TButton", padding=6, relief="flat",
background="#ccc")
```

- map(style, query_opt=None, **kw) – конфігурування зовнішнього вигляду віджетів в залежності від їх стану (active, pressed, disabled тощо):

Python

```
ttk.Style().map('C.TButton', foreground=[('pressed',
'red'), ('active', 'blue')], background=[('pressed', '!disabled',
'black'), ('active', 'white')])
```

- lookup(style, option, state=None, default=None) – повертає відповідну опцію конфігурування:

Python

```
ttk.Style().lookup("TButton", "font")
```

– layout – змінює схему віджету (віджети складаються з елементів, опцій конфігурування та вкладених layouts):

Python

```
ttk.Style().layout("TMenubutton",
[("Menubutton.background", None),
("Menubutton.button", {"children": [
[("Menubutton.focus", {"children": [
[("Menubutton.padding", {"children": [
[("Menubutton.label", {"side": "left", "expand": "1})]
}], "side": "left", "expand": "1})]
}], "side": "left", "expand": "1}),
]), "side": "left", "expand": "1}])
```

– element_create (elementname, etype, *args, **kw) – створює новий елемент поточної теми заданого типу etype (“image”, “from” для клонування з іншої теми заданого елементу, “vsapi”);
 – element_names () – повертає перелік елементів поточної теми;
 – element_options (elementname) – повертає перелік опцій заданого елементу;
 – theme_create (themename, parent=None, settings=None) – створює нову тему;
 – theme_settings – тимчасово змінює поточну тему на задану, застосовує вказані налаштування та повертається до попередньої теми (аргументи: назва теми та словник, ключами якого є назви стилів, а значеннями – layout відповідного стилю):

Python

```
ttk.Style().theme_settings("default", {
    "TCombobox": {
        "configure": {"padding": 5},
        "map": {
```

- theme_names () – повертає перелік доступних тем;
 - theme_use ([themename]) – змінює поточну тему на задану.

Більш детально з повним переліком методів та властивостей розглянутих віджетів можна ознайомитись у [24].

Розглянемо приклад побудови графічного інтерфейсу для програми:

Python

```
from tkinter import *
from tkinter import ttk

root = Tk()
root.title('Example')

content = ttk.Frame(root, padding=(3,3,12,12))
frame = ttk.Frame(content, borderwidth=5,
relief="sunken", width=200, height=100)
namelbl = ttk.Label(content, text="Name")
name = ttk.Entry(content)

onevar = BooleanVar()
twovar = BooleanVar()
threevar = BooleanVar()

one = ttk.Checkbutton(content, text="One",
variable=onevar, onvalue=True)
```

Python

```

two = ttk.Checkbutton(content, text="Two",
variable=twovar, onvalue=True)
three = ttk.Checkbutton(content, text="Three",
variable=threevar, onvalue=True)
ok = ttk.Button(content, text="OK")
cancel = ttk.Button(content, text="Cancel")

content.grid(column=0, row=0, sticky=(N, S, E, W))
frame.grid(column=0, row=0, columnspan=3, rowspan=2,
sticky=(N, S, E, W))
namelbl.grid(column=3, row=0, columnspan=2, sticky=(N,
W), padx=5)
name.grid(column=3, row=1, columnspan=2, sticky=(N, E,
W), pady=5, padx=5)
one.grid(column=0, row=3)
two.grid(column=1, row=3)
three.grid(column=2, row=3)
ok.grid(column=3, row=3)
cancel.grid(column=4, row=3)

root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)
content.columnconfigure(0, weight=3)
content.columnconfigure(1, weight=3)
content.columnconfigure(2, weight=3)
content.columnconfigure(3, weight=1)
content.columnconfigure(4, weight=1)
content.rowconfigure(1, weight=1)

root.mainloop()

```

На рис. 1.1 наведено інтерфейс, що відповідає даній програмі.

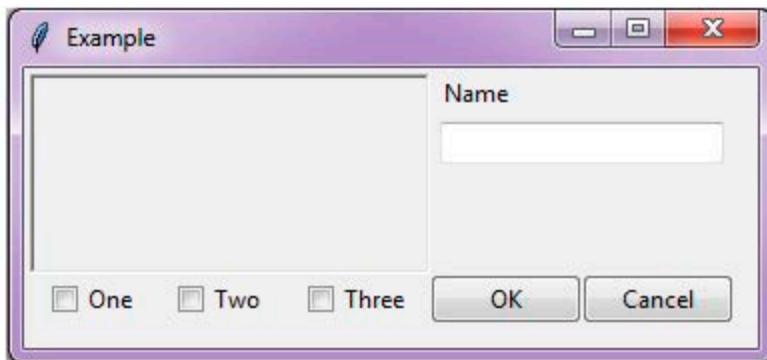


Рисунок 1.1 – Приклад графічного інтерфейсу

При підготовці матеріалів даного пункту використано документацію [21] (зокрема розділ [<https://docs.python.org/3/library/tkinter.ttk.html>]), [22] (зокрема розділ [<https://tkdocs.com/tutorial/grid.html>]), [23] (зокрема розділ [<https://documentation.help/Python-3.6.3/tkinter.ttk.html>]), [24], [31], [32] (зокрема розділи [<https://www.tcl.tk/man/tcl8.5/TkCmd/place.html>], [<https://www.tcl.tk/man/tcl8.5/TkCmd/pack.html>]), а також документацію [<https://docs.python.org/2/library/tk.html>].

1.3. Завдання на лабораторну роботу

1.3.1. Ознайомитись з теоретичними відомостями, необхідними для виконання роботи.

1.3.2. Розробити програмне забезпечення з графічним інтерфейсом користувача та ієархією об'єктів у відповідності з індивідуальним завданням, узгодженим з викладачем.

Оголошення об'єктів винести в окрімий модуль. Виконати документування коду для кожного модуля.

1.3.2.1. Керування бібліотекою. Клас Бібліотека повинен дозволяти додавати, вилучати та шукати книги та читачів. Читач може брати не більше п'яти книжок. Якщо читач хоче взяти книгу, а наявних екземплярів немає, то він автоматично додається в чергу. Після того, як читач повертає книгу, то вона автоматично надається першому читачу з черги, якщо він її може взяти. У разі використання книги довше заданого періоду часу нараховується пеня.

1.3.2.2. Доставка вантажів. Користувачам надається можливість оформлення замовлень на доставку товарів, відстеження поточного стану посилки. Оператори можуть змінювати стан замовлення. Для кожного замовлення обирається відповідний автомобіль для доставки. Якщо всі місця в автомобілі зайняті, то користувач має бути проінформований про це.

1.3.2.3. Облік надання послуг користування Інтернетом. Представники інтернет-провайдера можуть визначати наявні тарифи. Абоненти можуть обирати необхідний тариф. Оператори визначають дані про трафік кожного абонента. Абоненти можуть переглядати власну історію користування. Програма обчислює суму до сплати для кожного користувача за кожний місяць.

1.3.2.4. Підтримка процесу тестування. Адміністратор може створювати нові тестування, додавати запитання до визначеного тестування. Для кожного запитання адміністратор визначає кількість варіантів відповідей і вірних варіантів, безпосередньо самі варіанти відповідей, а також вірні відповіді. Звичайні користувачі можуть проходити тестування за вибраним тестом.

1.3.2.5. Контроль за школолярами. Кожний школляр (як і всі користувачі) ідентифікується системою і додається директором, результати його навчання (присутність, оцінки) визначаються вчителями. Батьки можуть переглядати інформацію тільки про власних дітей.

1.3.2.6. Надання доступу до інформації про власників нерухомості в місті Запоріжжя. Має надаватися можливість внесення відповідних даних, редактування та виконання пошуку. Для кожного власника має розраховуватися податок, який він має сплатити за рік. Податок залежить від площі кожного об'єкту, від кількості об'єктів одного власника. Після додання нових об'єктів податок має перераховуватися.

1.3.2.7. Оренда автомобілів. Адміністратори визначають перелік наявних автомобілів, їх вартість тощо. Орендар може виконувати пошук за необхідними параметрами, брати авто в оренду. Вартість послуги розраховується автоматизовано перед початком оренди та перераховується після повернення автомобіля.

1.3.2.8. Оформлення патентів на винаходи. Надається можливість виконувати пошук зареєстрованих патентів за критеріями.

Звичайні користувачі можуть подавати заявку на реєстрацію патенту, ідентифікуючи його відповідним чином, визначаючи відповідального патентного повіреного, переглядати статус заявки та редагувати її. Патентний повірений може переглядати надіслані йому заявки, редагувати їх та реєструвати в системі в якості патентів.

1.3.2.9. Відбір студентів для участі в освітніх проектах в університеті. Інформація про кандидатів включає зокрема його бал сертифікату зі знання англійської мови, середній бал навчання та спеціальність конкурсу. Адміністратор вносить в систему інформацію про кількість місць за кожною спеціальністю.

1.3.2.10. Замовлення таксі. Клієнту надається можливість обрати автомобіль або водія, отримати інформацію про час, через який має з'явитися таксі, орієнтовну вартість. Клієнт може інформувати про запізнення автомобіля – дана інформація накопичується для кожного водія та відображається іншим користувачам. Кожне замовлення зберігається разом з фактичною вартістю.

1.3.2.11. Підтримка навчання студентів, які можуть підписуватися на проходження курсів. У межах кожного курсу студенти можуть переглядати лекції, отримувати і виконувати завдання. Кожен курс та окремі його лекції доступні протягом заданого викладачем проміжку часу.

1.3.2.12. Керування рахунками в банку. Менеджер може керувати (переглядати, редагувати, зберігати) рахунками, послідовно переміщуючись між ними за PIN-кодом (а також на перший та останній код з можливістю створення нового) або виконуючи пошук за номером телефону, а клієнт може керувати тільки власним рахунком та змінювати пароль.

1.3.2.13. Футбольний тоталізатор. Визначаються футбольні матчі та варіанти результатів. Користувачі роблять ставки, визначаючи результат та суму грошей. Результати визначаються випадковим чином після настання відповідної дати. За отриманими результатами визначається виграш кожного учасника.

1.3.2.14. Підтримка спільніх поїздок автомобілем. Власники автомобілів визначають маршрут поїздки, дати, кількість наявних місць, вартість тощо. Звичайні користувачі можуть виконувати пошук та бронювати місця.

1.3.2.15. Порівняння цін на ліки. Користувачі можуть виконувати пошук наявності ліків у різних аптеках шляхом введення назви ліків та необхідної кількості. За заданими параметрами виконується пошук з сортуванням за збільшенням вартості ліків. Пошук відбувається за заданим користувачем поштовим індексом, що обмежує аптеки, які переглядаються.

1.3.2.16. Підтримка процесу винаймання робітників. Менеджери визначають посади, на які необхідно винайняти працівників, вимоги до них, переглядають послідовно резюме кандидатів на кожну посаду та оцінюють їх. Звичайні користувачі можуть подавати власні резюме та переглядати результати розгляду резюме.

1.3.2.17. Підтримка роботи туристичного агентства. Клієнти можуть переглядати доступні пропозиції та оформляти замовлення з врахуванням можливої знижки (в залежності від історії замовлень). Менеджери можуть переглядати замовлення, змінювати їх статус та переглядати статистику.

1.3.2.18. Менеджер фінансів. Користувачі можуть керувати власними фінансами шляхом додавання записів про доходи та видатки за групами, переглядати послідовно власні записи, переглядати найбільші витрати та доходи, аналізувати дані за часовими інтервалами тощо.

1.3.2.19. Продаж квитків у кінотеатр з можливістю переглядати сеанси, переглядати доступні та зайняті місця для перегляду заданого сеансу у відповідному залі, бронювання та звільнення місць.

1.3.2.20. Оформлення послуг. Користувачі можуть визначати власні послуги, період їх доступності, вартість, географічну зону їх доступності. Замовники послуг можуть виконувати пошук, задаючи власне розташування, оформлювати замовлення послуг.

1.3.3. Виконати тестування розробленого програмного забезпечення.

1.3.4. Оформити звіт.

1.3.5. Відповісти на контрольні запитання.

1.4. Зміст звіту

1.4.1. Мета роботи.

1.4.2. Завдання до роботи.

- 1.4.3. Текст розробленого програмного забезпечення.
- 1.4.4. Результати тестування: вхідні дані, скрипти та результати роботи програми.
- 1.4.5. Інтерфейс роботи з програмою в декількох режимах.
- 1.4.6. Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

1.5. Контрольні запитання

- 1.5.1. Що таке модуль та пакет?
- 1.5.2. Яким чином файли трансформуються в простори імен?
- 1.5.3. Які існують шаблони проєктування?
- 1.5.4. Які пакети використовуються для побудови графічних інтерфейсів у Python?
- 1.5.5. Які засоби має бібліотека tkinter?
- 1.5.6. Яким чином визначається клас у Python?
- 1.5.7. Яким чином реалізуються принципи об'єктно-орієнтованого програмування в Python?

ЛАБОРАТОРНА РОБОТА № 2

ВИКОРИСТАННЯ БІБЛІОТЕК РУТНОН ДЛЯ

ВИСОКОПРОДУКТИВНИХ НАУКОВИХ ОБЧИСЛЕНИЬ

2.1. Мета роботи

2.1.1 Ознайомитись з основними можливостями бібліотеки NumPy та навчитися використовувати їх на практиці.

2.1.2 Ознайомитись з основними можливостями бібліотеки SciPy та навчитися використовувати їх на практиці для виконання високопродуктивних наукових обчислень.

2.2. Короткі теоретичні відомості

2.2.1 Бібліотека NumPy

Бібліотека NumPy забезпечує в якості основного типу даних N-мірний масив – ndarray. Об'єкт ndarray містить елементи однакового типу та розміру. Основні атрибути об'єктів ndarray:

- ndarray.ndim – кількість вимірів масиву (ранг);
- ndarray.shape – розмір масиву, кортеж натуральних чисел, що представляють довжину масиву за кожною віссю;
- ndarray.size – кількість елементів масиву;
- ndarray.dtype – об'єкт, що описує тип елементів масиву;
- ndarray.itemsize – розмір кожного елементу в байтах;
- ndarray.data – буфер, що містить фактичні елементи масиву.

Масив можна утворити за допомогою функції *array* зі звичайних переліків або кортежів:

Python

```
>>> a=array([[2,3,4],[1,2,3]])
```

Окрім того можна утворити масив, заповнений нулями, за допомогою функції zeros, передаючи їй кількість елементів за кожною віссю. Для заповнення масиву одиницями призначена відповідна функція ones.

Функція `arange` призначена для заповнення масиву значеннями в заданому інтервалі з заданим кроком:

- `arange (x)` – значення в інтервалі від 0 до $x-1$;
- `arange (x, y)` – значення в інтервалі від x до y ;
- `arange (x, y, z)` – значення в інтервалі від x до y з кроком z .

Функція `linspace` створює масив з заданою кількістю елементів, розподілених рівномірно між значеннями x і y : `linspace (x, y, num)`.

Функція `indices` створює множину масивів, кожний з яких представляє відповідну вісь та варіації за нею.

Функція `empty` повертає масив заданого розміру без ініціалізації значень.

Арифметичні операції, функції `sin`, `cos`, `exp` тощо над масивами виконуються поелементно.

Функція `diag()` добуває діагональ або конструює діагональний масив.

Операції, які застосовуються до багатомірних масивів як до списків, можна застосувати за окремою віссю за допомогою параметра `axis`, наприклад:

Python

```
>>> b.sum(axis= 0)
```

Об'єкти типу `ndarray` мають велику кількість методів, частина з яких представлена в таблиці 2.1.

Таблиця 2.1 – Методи об'єктів типу `ndarray`

Метод	Опис
<code>ndarray.item(*args)</code>	Копіює елемент масиву в стандартний скаляр та повертає його.
<code>ndarray.tolist()</code>	Повертає масив у вигляді списку.
<code>ndarray.itemset(*args)</code>	Вставляє скалярне значення у масив.
<code>ndarray.tofile(fid[, sep, format])</code>	Записує масив у файл в текстовому або двійковому вигляді.
<code>ndarray.dump(file)</code>	Виконує дамп масиву у вказаний файл
<code>ndarray.dumps()</code>	Виконує дамп масиву в рядок.

Продовження таблиці 2.1

Метод	Опис
ndarray.astype(dtype[, order, casting, ...])	Повертає копію масиву, змінюючи тип.
ndarray.byteswap(inplace)	Змінює місцями байти елементів масиву.
ndarray.copy([order])	Повертає копію масиву.
ndarray.view([dtype, type])	Перетворює масив заданим чином, не змінюючи дані, та створює новий масив.
ndarray.getfield(dtype[, offset])	Повертає представлення масиву зазначеним чином (можлива зміна даних).
ndarray.setflags([write, align, uic])	Встановлює флаги масиву WRITEABLE, ALIGNED та UPDATEIFCOPY відповідно.
ndarray.fill(value)	Заповнює масив скалярними значеннями.
ndarray.reshape(shape[, order])	Повертає масив зі зміненими розмірами.
ndarray.resize(new_shape[, refcheck])	Змінює розмір масиву на місці.
ndarray.transpose(*axes)	Повертає транспонований масив.
ndarray.swapaxes(axis1, axis2)	Міняє місцями дві вісі масиву.
ndarray.flatten([order])	Повертає копію масиву, перетворену в одновимірний вигляд.
ndarray.ravel([order])	Повертає одномірний масив.
ndarray.take(indices[, axis, out, mode])	Повертає масив, сформований з елементів за заданими індексами.
ndarray.put(indices, values[, mode])	Встановлює $a.flat[n] = values[n]$ для всіх індексів.
ndarray.repeat(repeats[, axis])	Повторює елементи масиву.
ndarray.choose(choices[, out, mode])	Використовує масив індексів для створення нового масиву з множини choices.

Продовження таблиці 2.1

Метод	Опис
ndarray.sort([axis, kind, order])	Сортує масив, виконуючи операцію на місці.
ndarray.argsort([axis, kind, order])	Повертає індекси елементів за розташуванням у відсортованому масиві.
ndarray.partition(kth[, axis, kind, order])	Змінює порядок елементів у масиві таким чином, що елемент зі вказаним індексом розташовується таким чином, що перед ним розташовані елементи з меншим значенням, а правіше – з рівним або більшим.
ndarray.argpartition(kth[, axis, kind, order])	Повертає індекси елементів масиву аналогічно ndarray.partition.
ndarray.searchsorted(v[, side, sorter])	Повертає індекси, куди мають бути вставлені елементи, щоб зберегти порядок.
ndarray.nonzero()	Повертає індекси ненульових елементів.
ndarray.compress(condition[, axis, out])	Повертає вибрані зрізи масиву вздовж заданої вісі.
ndarray.trace([offset, axis1, axis2, dtype, out])	Повертає суму елементів вздовж діагоналей.
ndarray.sum([axis, dtype, out])	Повертає суму елементів вздовж заданої вісі.
ndarray.cumsum([axis, dtype, out])	Повертає кумулятивну суму елементів вздовж заданої вісі.
ndarray.mean([axis, dtype, out])	Повертає середнє елементів масиву вздовж заданої вісі.
ndarray.var([axis, dtype, out, ddof])	Повертає дисперсію елементів масиву вздовж заданої вісі.
ndarray.std([axis, dtype, out, ddof])	Повертає стандартне відхилення елементів масиву вздовж даної вісі.
ndarray.prod([axis, dtype, out])	Повертає добуток елементів за заданою віссю.

Продовження таблиці 2.1

Метод	Опис
ndarray.cumprod([axis, dtype, out])	Повертає кумулятивний добуток елементів за заданою віссю.
ndarray.all([axis, out])	Повертає True, якщо всі елементи дорівнюють True.
ndarray.any([axis, out])	Повертає True, якщо будь-який елемент дорівнює True.

Функція `numpy.mat(data, dtype=None)` представляє введені дані у вигляді матриці. Об'єкт матриця має зокрема наступні атрибути, що спрощують роботу з ними:

- A – повертає результат у вигляді об'єкту ndarray;

- T – повертає транспоновану матрицю;

Створити такий об'єкт можна наступним чином:

Python

```
>>> A = numpy.matrix('7.5 2.3; 9.1 1.5')
```

Об'єкт типу даних є екземпляром класу `numpy.dtype` і описує, яким чином інтерпретуються байти у блоці пам'яті фіксованого розміру.

Для створення таких об'єктів використовується конструктор:

Python

```
dt = numpy.dtype(numpy.int32)
```

Створений об'єкт має серед інших наступні атрибути:

- `dtype.type` – тип об'єкту;

- `dtype.char` – символічний код типу;

- `dtype.num` – унікальне число для кожного з типів;

- `dtype.itemsize` – розмір елементу об'єкту даного типу;

- `dtype.byteorder` – порядок байтів.

За допомогою модуля `numpy.ma` можна створити маски масивів, які дозволяють виділити пропущені або невірні значення. Наприклад, один зі способів, яким такий масив можна створити:

Python

```
import numpy.ma as ma
m_x = ma.masked_array(x, mask=[0, 0, 0, 1, 0])
```

Можна також створити такий масив за допомогою маскування інтервалу значень (`masked_inside(x, v1, v2[, copy])` або протилежна їй за суттю `masked_outside(x, v1, v2[, copy])`) або за обмеженнями (`masked_greater(x, value[, copy])`, `masked_greater_equal(x, value[, copy])`, `masked_less(x, value[, copy])`, `masked_less_equal(x, value[, copy])`, `masked_equal(x, value[, copy])`, `masked_not_equal(x, value[, copy])`).

Змінити маску можна за допомогою відповідного атрибуту.

Для роботи з датами використовується відповідний тип даних:

Python

```
numpy.datetime64('2022-05-11')
numpy.datetime64('2022-07-21T02:50')
```

Даний тип підтримується багатьма загальними функціями бібліотеки NumPy, зокрема створення діапазонів, порівняння, віднімання, складання тощо.

Тип `Timedelta` дозволяє визначити інтервал часу. Існують спеціальні функції, які дозволяють виконувати операції з бізнес-днями (наприклад, перевіряє дату на відповідну умову функція `numpy.is_busday()`.

Функція `copyto(dst, src, casting='same_kind', where=None)` дозволяє копіювати дані з одного масиву в інший, виконуючи необхідні перетворення.

Для об'єднання масивів використовуються наступні функції:

- `hstack(tup)` – для горизонтального об'єднання масивів у

послідовність;

– `vstack(tup)` – для вертикального об'єднання масивів у послідовність;

– `concatenate((a1, a2, ...), axis=0)` – приєднує послідовність масивів;

– `column_stack(tup)` – об'єднує 1-D масиви у вигляді колонок у 2-D масиви.

Для роз'єднання масивів використовуються наступні функції:

– `array_split(ary, indices_or_sections[, axis])` – розділяє масив на підмасиви;

– `dsplit(ary, indices_or_sections)` – розділяє масив на підмасиви вздовж третьої вісі (вглиб);

– `hsplit(ary, indices_or_sections)` – розділяє масив на підмасиви горизонтально;

– `split(ary, indices_or_sections[, axis])` – розділяє масив на підмасиви;

– `vsplit(ary, indices_or_sections)` – розділяє масив на підмасиви вертикально.

Для додавання та вилучення елементів призначені наступні функції:

– `delete(arr, obj[, axis])` – вилучає елементи з масиву вздовж заданої вісі;

– `insert(arr, obj, values[, axis])` – вставляє значення вздовж заданої вісі перед заданими індексами;

– `append(arr, values[, axis])` – додає значення в кінець масиву;

– `trim_zeros(filter[, trim])` – вилучає нулі на початку і/або в кінці одномірного масиву або послідовності;

– `unique(ar[, return_index, return_inverse, ...])` – виділяє унікальні елементи масиву.

Для модифікації масивів можуть використовуватися наступні функції:

– `fliplr(m)` – горизонтальне дзеркальне відображення;

– `flipud(m)` – вертикальне дзеркальне відображення;

– `roll(a, shift[, axis])` – повертає елементи масиву навколо заданої вісі;

– `rot90(m[, k])` – повертає масив на 90 градусів проти годинникової стрілки.

Функції для виконання поелементних побітових операцій: `bitwise_and(x1, x2[, out])`, `bitwise_or(x1, x2[, out])`, `bitwise_xor(x1, x2[, out])`, `invert(x[, out])`, `left_shift(x1, x2[, out])`, `right_shift(x1, x2[, out])`.

Логічні функції включають логічні операції (`logical_and(x1, x2[, out])`, `logical_or(x1, x2[, out])`, `logical_not(x[, out])`, `logical_xor(x1, x2[, out])`), перевірки типу елементів масиву (`iscomplex()`, `isreal()`), порівняння масивів (`array_equal(a1, a2)`, `greater(x1, x2[, out])`, `greater_equal(x1, x2[, out])`, `less(x1, x2[, out])`, `less_equal(x1, x2[, out])`).

Бібліотека містить набір функцій лінійної алгебри (модуль `numpy.linalg`):

- `dot(a, b[, out])` – матричний добуток двох масивів;
- `vdot(a, b)` – добуток двох векторів;
- `matrix_power(M, n)` – підносить квадратну матрицю у заданий ступінь;
- `eig(a)` – власні значення та вектори квадратної матриці;
- `norm(x, ord=None, axis=None, keepdims=False)` – норма матриці;
- `det(a)` – детермінант матриці;
- `solve(a, b)` – розв'язує лінійне матричне рівняння або систему лінійних рівнянь.

Бібліотека підтримує також набір функцій для функціонального програмування:

- `apply_along_axis(func1d, axis, arr, *args, ...)` – застосовує функцію до одномірних зрізів вздовж даної вісі;
- `apply_over_axes(func, a, axes)` – застосовує функцію, повторюючи її для набору осей;
- `vectorize(pyfunc[, otypes, doc, excluded, cache])` – визначає векторизовану функцію;
- `piecewise(x, condlist, funclist, *args, **kw)` – визначає функцію, значення якої визначається умовою.

Для генерації масивів використовуються зокрема наступні функції:

- `c_` – виконує конкатенацію вздовж другої вісі;
- `r_` – виконує конкатенацію вздовж першої вісі;
- `s_` – повертає кортежі індексів для масивів;
- `nonzero(a)` – повертає індекси елементів, що не дорівнюють нулю;
- `where(condition, [x, y])` – повертає елементи з `x` або `y` залежно

від умови;

– `ix_(*args)` – створює кортеж з послідовностей.

Функція `covcoef(x[, y, rowvar, bias, ddof])` повертає коефіцієнт кореляції, а функція `cov(m[, y, rowvar, bias, ddof])` оцінює коваріаційну матрицю.

Для створення та оброблення багаточленів використовується пакет `numpy.polynomial`, який дозволяє розрахувати багаточлен в заданій точці (`polyval(x, c[, tensor])`) або двовимірний багаточлен (`polyval2d(x, y, c)`), диференціювати багаточлен (`polyder(c[, m, scl, axis])`), виконувати арифметичні операції над багаточленами (`polyadd(c1, c2), polypow(c, pow[, maxpower])`).

Підтримку виконання тестування забезпечує модуль `numpy.testing`, що містить функції, які в разі нездоволення умови, що перевіряється, повертають `AssertionError` (наприклад, `assert_almost_equal(actual, desired[, ...])` – порівняння з заданою точністю).

Для виведення масиву в файл використовується функція `save(file, arr)`, а для завантаження масиву з файлу – `load(file[, mmap_mode])`. Для аналогічної роботи з текстовими файлами призначенні функції `loadtxt(fname[, dtype, comments, delimiter, ...])` та `savetxt(fname, X[, fmt, delimiter, newline, ...])`.

Окрім того бібліотека містить модуль для виконання дискретного перетворення Фур'є (`numpy.fft`), набір фінансових функцій, функції для роботи з рядками, математичні функції, функцій визначення випадкових значень та отримання розподілів (`numpy.random`).

При підготовці матеріалів даного пункту використано документацію [11] (зокрема [<https://numpy.org/doc/stable/reference/arrays.ndarray.html>], [<https://numpy.org/doc/stable/reference/routines.matlib.html>], [<https://numpy.org/doc/stable/reference/arrays.dtypes.html>], [<https://numpy.org/doc/stable/reference/routines.linalg.html>], [<https://numpy.org/doc/stable/reference/routines.array-manipulation.html>], [<https://numpy.org/doc/stable/reference/routines.bitwise.html>], [<https://numpy.org/doc/stable/reference/routines.testing.html>], [<https://numpy.org/doc/stable/reference/routines.logic.html>], [<https://numpy.org/doc/stable/reference/routines.io.html>], [<https://numpy.org/doc/stable/reference/routines.ma.html>], [<https://numpy.org/doc/stable/reference/routines.polynomials.html>]),

`poly1d.html]`, [<https://numpy.org/doc/stable/reference/arrays.indexing.html>], [<https://numpy.org/devdocs/user/basics.creation.html>], [<https://numpy.org/devdocs/reference/arrays.datetime.html>]), [33].

2.2.2 Бібліотека SciPy

Бібліотека SciPy побудована на основі NumPy та надає потужний науковий інструментарій.

Бібліотека SciPy зокрема містить наступні пакети:

- спеціальних функцій (`scipy.special`);
- чисельного розв'язання звичайних диференційних рівнянь (`scipy.integrate`);
- оптимізації (`scipy.optimize`);
- інтерполяції (`scipy.interpolate`);
- перетворення Фур'є (`scipy.fftpack`);
- оброблення сигналів (`scipy.signal`);
- лінійної алгебри (`scipy.linalg`);
- розріджених графів (`scipy.sparse.csgraph`);
- просторові структури даних та алгоритми (`scipy.spatial`);
- випадкових чисел та статистики (`scipy.stats`);
- багатовимірного оброблення зображень (`scipy.ndimage`);
- файлового вводу-виводу (`scipy.io`);
- включення коду C/C++ (`scipy.weave`);
- алгоритмів кластеризації (`scipy.cluster`);
- розріджених матриць (`scipy.sparse`).

Бібліотека `scipy.linalg` містить функції лінійної алгебри:

- `inv(a[, overwrite_a, check_finite])` – обчислює зворотну матрицю;
- `solve(a, b[, sym_pos, lower, overwrite_a, ...])` – розв'язує рівняння $a x = b$ для x ;
- `solve_banded(l_and_u, ab, b[, overwrite_ab, ...])` – розв'язує рівняння $a x = b$ для x за умови, що a – стрічкова матриця;
- `solveh_banded(ab, b[, overwrite_ab, ...])` – розв'язує рівняння $a x = b$;
- `solve_triangular(a, b[, trans, lower, ...])` – розв'язує рівняння $a x = b$ для x за умови, що a – трикутна матриця;
- `det(a[, overwrite_a, check_finite])` – обчислює детермінант

матриці;

- `norm(a[, ord])` – матрична або векторна норма;
- `ltsq(a, b[, cond, overwrite_a, ...])` – обчислює розв'язання рівняння $Ax = b$ методом найменших квадратів;
- `pinv(a[, cond, rcond, return_rank, check_finite])` – обчислює псевдозворотню матрицю;
- `kron(a, b)` – кронекерівський добуток;
- `tril(m[, k])` і `triu(m[, k])` – копіюють матрицю, встановлюючи значення над і під k -ою діагональю відповідно рівними нулю;
- `eig(a[, b, left, right, overwrite_a, ...])` – розв'язує просту або узагальнену задачу про власні числа квадратної матриці;
- `eigvals(a[, b, overwrite_a, check_finite])` – обчислює власні числа з простої або узагальненої задачі про власні числа;
- `lu(a[, permute_l, overwrite_a, check_finite])` – обчислює LU-декомпозицію матриці;
- `lu_solve(lu_and_piv, b[, trans, ...])` – обчислює систему рівнянь, $ax = b$, за умови LU-факторизації a ;
- `svd(a[, full_matrices, compute_uv, ...])` – розкладає матрицю за сингулярними числами;
- `svdvals(a[, overwrite_a, check_finite])` – обчислює сингулярні значення матриці;
- `diagsvd(s, M, N)` – створює сигма-матрицю в сингулярному розкладі матриці з сингулярних чисел та розміру M, N ;
- `orth(A)` – створює ортогональний нормований базис для діапазону A , використовуючи сингулярний розклад матриці;
- `cholesky(a[, lower, overwrite_a, check_finite])` – обчислює декомпозицію Холеського матриці;
- `cholesky_banded(ab[, overwrite_ab, lower, ...])` – декомпозиція Холецького стрічкової ермітової позитивно-візначеної матриці;
- `cho_factor(a[, lower, overwrite_a, check_finite])` – обчислює декомпозицію Холеського для матриці з метою використання в `cho_solve`;
- `cho_solve(c_and_lower, b[, overwrite_b, ...])` – розв'язує ліній-ні рівняння $Ax = b$ за умови факторизації Холеського для A ;
- `polar(a[, side])` – обчислює полярне розвинення;
- `qr(a[, overwrite_a, lwork, mode, pivoting, ...])` і `rq(a[, overwrite_a, lwork, mode, check_finite])` – обчислює QR- і RQ-декомпозиції матриці;

- `qr_multiply(a, c[, mode, pivoting, ...])` – обчислює QR-декомпозицію та перемножує Q з матрицею;
- `qz(A, B[, output, lwork, sort, overwrite_a, ...])` – QZ-декомпозиція для узагальнених власних чисел пари матриць;
- `schur(a[, output, lwork, overwrite_a, sort, ...])` – обчислює декомпозицію Шура матриці;
- `hessenberg(a[, calc_q, overwrite_a, ...])` – обчислює форму Хесенберга матриці;
- `expm(A[, q])` – обчислює матричний експоненціал, використовуючи апроксимацію Паде;
- `logm(A[, disp])` – обчислює матричний логарифм;
- `cosm(A), sinm(A), tanm(A)` – обчислюють матричний косинус, синус і тангенс відповідно;
- `coshm(A), sinhm(A), tanhm(A)` – обчислюють гіперболічний матричний косинус, синус і тангенс відповідно;
- `signm(A[, disp])` – матрична знакова функція;
- `sqrtm(A[, disp, blocksize])` – матричний квадратний корінь;
- `funm(A, func[, disp])` – обчислює матричну функцію, задану під час виклику;
- `expm_frechet(A, E[, method, compute_expm, ...])` – похідна Фреше матричного експоненціала A у напрямку E ;
- `expm_cond(A[, check_finite])` – відносне число обумовленості матричного експоненціала у нормі Фробеніуса;
- `fractional_matrix_power(A, t)` – обчислює дробний ступінь матриці;
- `solve_sylvester(a, b, q)` – розв'язує (X) рівняння Сильвестра ($AX + XB = Q$);
- `solve_continuous_are(a, b, q, r)` – розв'язує неперервне алгебраїчне рівняння Ріккаті, визначене як $(A'X + XA - XBR^{-1}B'X + Q = 0)$, використовуючи метод декомпозиції Шура;
- `solve_discrete_are(a, b, q, r)` – розв'язує дискретне алгебраїчне рівняння Ріккаті, визначене як $(X = A'XA - (A'XB)(R+B'XB)^{-1}(B'XA)+Q)$, використовуючи метод декомпозиції Шура;
- `solve_discrete_lyapunov(a, q[, method])` – розв'язує дискретне рівняння Ляпунова ($A'XA - X = -Q$);
- `solve_lyapunov(a, q)` – розв'язує неперервне рівняння Ляпунова

$(AX + XA^H = Q)$ за заданих значень A і Q , використовуючи алгоритм Бартелса-Стюарта;

– `block_diag(*arrs)` – створює блочну діагональну матрицю з переданих масивів;

– `circulant(c)`, `companion(a)`, `dft(n[, scale])` – створює циркулянтну, супроводжуючу матрицю, матрицю дискретного перетворення Фур'є;

– `hadamard(n[, dtype])`, `hankel(c[, r])`, `hilbert(n[, exact])` – створює матрицю Адамара, Ханкеля, Гілберта порядка n або зворотну відповідно;

– `tri(N[, M, k, dtype])` – створює (N, M) матрицю, заповнену нулями на і нижче k -ої діагоналі.

Підпакет `scipy.integrate` зокрема включає наступні функції:

– `quad(func, a, b[, args, full_output, ...])` – обчислює визначений інтеграл;

– `dblquad(func, a, b, gfun, hfun[, args, ...])` – обчислює подвійний інтеграл;

– `tplquad(func, a, b, gfun, hfun, qfun, rfun)` – обчислює потрійний інтеграл;

– `nquad(func, ranges[, args, opts])` – інтегрування за декількома змінними;

– `romb(y[, dx, axis, show])` – інтеграція Ромберга, використовуючи вибірки функції;

– `odeint(func, y0, t[, args, Dfun, col_deriv, ...])` – інтегрування системи звичайних диференційних рівнянь.

Пакет оптимізації містить функції локальної, глобальної оптимізації, Розенброка, лінійного програмування тощо.

Функції локальної оптимізації включають:

– `minimize(fun, x0[, args, method, jac, hess, ...])` – мінімізує скалярну функцію однієї або більше змінних одним з методів, представлених параметром `method` ('Nelder-Mead', 'Powell', 'CG', 'BFGS', 'Newton-CG', 'Anneal', 'L-BFGS-B', 'TNC', 'COBYLA', 'SLSQP', 'dogleg', 'trust-nocg', 'custom');

– `minimize_scalar(fun[, bracket, bounds, ...])` – мінімізує скалярну функцію однієї змінної за допомогою методу, визначеного параметром `method` ('Brent', 'Bounded', 'Golden', 'custom');

– `OptimizeResult` – представляє результати оптимізації.

Функції глобальної оптимізації включають:

- `basinhopping(func, x0[, niter, T, stepsize, ...])` – знаходить глобальний мінімум функції, використовуючи стохастичний алгоритм Basin-Hopping;
- `brute(func, ranges[, args, Ns, full_output, ...])` – мінімізує функцію в заданому діапазоні за допомогою повного перебору;
- `differential_evolution(func, bounds[, args, ...])` – знаходить глобальний мінімум багатовимірної функції за допомогою методу диференційної еволюції.

Функції Розенброка включають:

- `rosen(x)` – повертає значення функції Розенброка;
- `rosen_der(x)` – повертає градієнт функції Розенброка;
- `rosen_hess(x)` – повертає матрицю Гессе функції Розенброка.
- `rosen_hess_prod(x, p)` – повертає добуток матриці Гессе функції Розенброка та вектора.

Функція `curve_fit(f, xdata, ydata[, p0, sigma, ...])` використовує нелінійний метод найменших квадратів для мінімізації відхилень функції від даних.

Функція `linprog(c[, A_ub, b_ub, A_eq, b_eq, bounds, ...])` мінімізує лінійну цільову функцію з обмеженнями у вигляді нерівностей та лінійних рівностей на основі лінійного програмування.

Функція `leastsq(func, x0[, args, Dfun, full_output, ...])` мінімізує суму квадратів множини рівнянь.

Функції корисності включають наступний набір:

- `approx_fprime(xk, f, epsilon, *args)` – кінцево-різницева апроксимація градієнта скалярної функції;
- `check_grad(func, grad, x0, *args, **kwargs)` – перевіряє вірність градієнтої функції шляхом порівняння її з кінцево-різницею апроксимацією градієнта;
- `line_search(f, myfprime, xk, pk[, gfk, ...])` – знаходить `alpha`, що задовільняє підсиливим умовам Вольфе;
- `show_options([solver, method])` – показує документацію за додатковими налаштуваннями оптимізаторів.

Бібліотека інтерполяції `scipy.interpolate` містить сплайн-функції і класи, одновимірні і багатовимірні інтерполяційні класи, поліноміальні інтерполятори Лагранжа `lagrange(x, w)` і Тейлора `approximate_taylor_polynomial(f, x, degree, ...)`, оболонки для функцій

FITPACK і DFITPACK.

Одномірна інтерполяція містить наступні функції:

– `interp1d(x, y[, kind, axis, copy, ...])` – дозволяє інтерполювати одновимірну функцію;

– `BarycentricInterpolator(xi[, yi, axis])` і `KroghInterpolator(xi, yi[, axis])` – інтерполяційний багаточлен для множини точок;

– `PchipInterpolator(x, y[, axis, extrapolate])` – одновимірна монотонна кубічна інтерполяція методом кускового кубічного інтерполяційного багаточлена Ерміта;

– `barycentric_interpolate(xi, yi, x[, axis]), krogh_interpolate(xi, yi, x[, der, axis])` – допоміжна функція для поліноміальної інтерполяції;

– `pchip_interpolate(xi, yi, x[, der, axis])` – допоміжна функція для інтерполяції методом кускового кубічного інтерполяційного багаточлена Ерміта;

– `Akima1DInterpolator(x, y)` – інтерполятор Акіма;

– `PPoly(c, x[, extrapolate])` і `BPoly(c, x[, extrapolate])` – кусковий багаточлен на основі коефіцієнтів та контрольних точок.

Багатомірна інтерполяція включає наступні функції:

– `griddata(points, values, xi[, method, ...])` – інтерполювати неструктуровані D-вимірні дані;

– `LinearNDInterpolator(points, values[, ...])` – кусково-лінійний інтерполянт у N вимірах;

– `NearestNDInterpolator(x, y[, rescale, ...])` – інтерполяція методом найближчого сусіда в N вимірах;

– `CloughTocher2DInterpolator(points, values[, tol])` – кусковий кубічний, C1-згладжуючий, мінімізуючий кривизну інтерполянт у 2D;

– `Rbf(*args)` – клас для апроксимації/інтерполяції радіально-базисною функцією n-вимірних розсіяних даних;

– `interp2d(x, y, z[, kind, copy, ...])` – інтерполювати по двовимірній сітці;

– `interpn(points, values, xi[, method, ...])` – багатовимірна інтерполяція на регулярних сітках;

– `RegularGridInterpolator(points, values[, ...])` – інтерполяція на регулярній сітці довільних розмірностей;

– `RectBivariateSpline(x, y, z[, bbox, kx, ky, s])` – апроксимація двомірним сплайном по прямокутній сітці.

Одновимірні сплайні включають:

- `UnivariateSpline(x, y[, w, bbox, k, s, ext])` – одновимірний згладжуючий сплайн, що відповідає даній множині точок даних;
- `InterpolatedUnivariateSpline(x, y[, w, ...])` – одновимірний інтерполяційний сплайн для даної множини точок даних;
- `LSQUnivariateSpline(x, y, t[, w, bbox, k, ext])` – одновимірний сплайн з явними внутрішніми вузловими точками.

Вищеперелічені класи одновимірних сплайнів мають наступні методи:

- `UnivariateSpline.__call__(x[, nu, ext])` – оцінює сплайн в точці x ;
- `UnivariateSpline.derivatives(x)` – повертає всі похідні сплайна в точці x ;
- `UnivariateSpline.integral(a, b)` – повертає визначений інтеграл сплайна між двома даними точками;
- `UnivariateSpline.roots()` – повертає нулі сплайна;
- `UnivariateSpline.derivative([n])` – створює новий сплайн, що представляє похідну даного сплайна;
- `UnivariateSpline.antiderivative([n])` – створює новий сплайн, що представляє антипохідну даного сплайна;
- `UnivariateSpline.get_coefficients()` – повертає коефіцієнти сплайна;
- `UnivariateSpline.get_knots()` – повертає позиції вузлових точок (границьких і внутрішніх) сплайна;
- `UnivariateSpline.get_residual()` – повертає зважену суму квадратних різниць сплайн-апроксимації: $\text{sum}((w[i] * (y[i]-spl(x[i])))**2, \text{axis}=0)$;
- `UnivariateSpline.set_smoothing_factor(s)` – продовжує обчислення сплайна з заданим фактором згладжування s із вузловими точками, знайденими під час останнього виклику.

Функціональний інтерфейс до функцій FITPACK:

- `splrep(x, y[, w, xb, xe, k, task, s, t, ...])`, `splprep(x[, w, u, ub, ue, k, task, s, t, ...])` – знаходить представлення В-сплайна одновимірної та N-вимірної кривої відповідно;
- `splev(x, tck[, der, ext])` – обчислює В-сплайн або його похідні;
- `splint(a, b, tck[, full_output])` – обчислює визначений інтеграл В-сплайна;
- `sproot(tck[, mest])` – знаходить корені кубічного В-сплайна;
- `spalde(x, tck)` – обчислює всі деривативи В-сплайна;

– `splder(tck[, n])` – обчислює представлення сплайнів похідної даного сплайна;

– `splantider(tck[, n])` – обчислює сплайн для антипохідної заданого сплайна.

Функції для 2-D сплайнів:

– `RectBivariateSpline(x, y, z[, bbox, kx, ky, s])` – двовимірна сплайн-апроксимація по прямокутній сітці;

– `BivariateSpline()` – клас для двовимірних сплайнів;

– `SmoothBivariateSpline(x, y, z[, w, bbox, ...])` – згладжуюча двовимірна сплайн-апроксимація;

– `SmoothSphereBivariateSpline(theta, phi, r[, ...])` – згладжуюча двовимірна сплайн-апроксимація у сферичних координатах;

– `LSQBivariateSpline(x, y, z, tx, ty[, w, ...])` – двовимірна сплайн-апроксимація за зваженим методом найменших квадратів та її варіант у сферичних координатах `LSQSphereBivariateSpline(theta, phi, r, tt, tp)`.

Пакет алгоритмів кластеризації містить два модуля: `vq`, який підтримує векторне квантування і алгоритм k -середніх, та `hierarchy`, який містить функції для ієрархічної кластеризації.

Модуль `scipy.cluster.vq` включає наступні функції:

– `whiten(obs)` – нормалізує групу спостережень;

– `kmeans(obs, k_or_guess[, iter, thresh])` – виконує алгоритм k -середніх на множині векторів спостережень, формуючи k кластерів;

– `kmeans2(data, k[, iter, thresh, minit, missing])` – класифікує множину спостережень на k -кластерів, використовуючи алгоритм k -середніх.

Модуль `scipy.cluster.hierarchy` включає наступні функції:

– `fcluster(Z, t[, criterion, depth, R, monocrit])` – формує плоскі кластери з ієрархічної кластеризації, визначеної з'єднуючою матрицею Z ;

– `fclusterdata(X, t[, criterion, metric, ...])` – кластеризує дані спостережень, використовуючи задану метрику;

– `leaders(Z, T)` – повертає кореневі вузли в ієрархічній кластеризації;

– `linkage(y[, method, metric])` – виконує ієрархічну/агломеративну кластеризацію на ущільненій матриці відстаней y ;

– `single(y)` – виконує встановлення зв'язків (просте/мінімальне/найближче) на ущільненій матриці відстаней y ;

- `complete(y)` – виконує точкове повне/максимальне/ найвіддаленіше з'єднання на стиснених матрицях відстаней;
- `average(y), weighted(y)` – виконує з'єднання методами незваженого та зваженого попарного арифметичного середнього відповідно на стиснених матрицях відстаней;
- `centroid(y), median(y)` – виконують з'єднання за допомогою методів незваженого та зваженого попарного центроїдного усереднення відповідно;
- `ward(y)` – виконує з'єднання методом Варда на стиснених або збиткових матрицях відстаней;
- `cophenet(Z[, Y])` – обчислює кофенетичні відстані між кожним спостереженням в ієрархічній кластеризації, визначеній з'єднанням Z ;
- `inconsistent(Z[, d])` – обчислює статистику несумісності на з'єднаннях;
- `maxinconsts(Z, R)` – повертає максимальний коефіцієнт несумісності для кожного неодноелементного кластера та його нащадків;
- `maxdists(Z)` – повертає максимальну відстань між будь-якими неодноелементними кластерами;
- `maxRstat(Z, R, i)` – повертає максимальну статистику для кожного неодноелементного кластера та його нащадків;
- `dendrogram(Z[, p, truncate_mode, ...])` – створює деревоподібну діаграму ієрархічної кластеризації;
- `ClusterNode(id[, left, right, dist, count])` – клас вузла дерева для представлення кластера;
- `leaves_list(Z)` – повертає перелік ідентифікаторів листів;
- `to_tree(Z[, rd])` – ковертує ієрархічну кластеризацію, закодовану в матриці Z у об'єкт дерево;
- `is_valid_im(R[, warning, throw, name])` – істинне, якщо передана матриця несумісності вірна;
- `is_valid_linkage(Z[, warning, throw, name])` – перевіряє вірність матриці з'єднань;
- `is_isomorphic(T1, T2)` – визначає, чи еквівалентні два виділення кластерів;
- `is_monotonic(Z)` – повертає істинне значення, якщо передана матриця з'єднань монотонна;
- `correspond(Z, Y)` – перевіряє відповідність між матрицею

з'єднань та стисненою матрицею відстаней;

– `num_obs_linkage(Z)` – повертає кількість початкових спостережень переданої матриці з'єднань.

Модуль `scipy.sparse.csgraph` містить швидкі алгоритми роботи з графами, що ґрунтуються на представленні у вигляді розріджених матриць:

– `connected_components(csgraph[, directed, ...])` – аналізує зв'язані компоненти розріджених графів;

– `laplacian(csgraph[, normed, return_diag, ...])` – повертає матрицю Кірхгофа орієнтовного графу;

– `shortest_path(csgraph[, method, directed, ...])` – виконує пошук найкоротшого шляху в позитивно направленому або неорієнтовному графі;

– `dijkstra(csgraph[, directed, indices, ...])` – алгоритм Дейкстри, використовуючи Фібоначчієві кучі;

– `floyd_marshall(csgraph[, directed, ...])` – обчислює довжину найкоротших шляхів, використовуючи алгоритм Флойда-Воршала;

– `bellman_ford(csgraph[, directed, indices, ...])` – обчислює довжину найкоротших шляхів, використовуючи алгоритм Беллмана-Форда;

– `johson(csgraph[, directed, indices, ...])` – обчислює довжину найкоротших шляхів, використовуючи алгоритм Джонсона;

– `breadth_first_order(csgraph, i_start[, ...])` – повертає перелік вузлів, впорядкованих за пошуком в ширину, починаючи з заданого вузла;

– `depth_first_order(csgraph, i_start[, ...])` – повертає перелік вузлів, впорядкованих за пошуком в глибину, починаючи з заданого вузла;

– `breadth_first_tree(csgraph, i_start[, directed])` – повертає дерево, згенероване методом пошуку в ширину;

– `depth_first_tree(csgraph, i_start[, directed])` – повертає дерево, згенероване методом пошуку в глибину;

– `minimum_spanning_tree(csgraph[, overwrite])` – повертає мінімальне зв'язуюче дерево неорієнтовного графа.

Модуль `scipy.fftpack`, який реалізує перетворення Фур'є, містить зокрема наступний набір функцій:

– `fft(x[, n, axis, overwrite_x])` – повертає дискретне перетво-

рення Фур'є дійсних або комплексних послідовностей (для оберненого перетворення – `ifft(x[, n, axis, overwrite_x])`);

– `fft2(x[, shape, axes, overwrite_x])` – 2-D дискретне перетворення Фур'є;

– `ifft2(x[, shape, axes, overwrite_x])` – 2-D дискретне обернене перетворення Фур'є дійсних або комплексних послідовностей;

– `fftn(x[, shape, axes, overwrite_x])` – повертає багатомірне дискретне перетворення Фур'є;

– `ifftn(x[, shape, axes, overwrite_x])` – повертає обернене багатомірне дискретне перетворення Фур'є послідовності x довільного типу;

– `rfft(x[, n, axis, overwrite_x])` – дискретне перетворення Фур'є дійсних послідовностей (для оберненого – `irfft(x[, n, axis, overwrite_x])`);

– `dct(x[, type, n, axis, norm, overwrite_x])` – повертає дискретне косинусне перетворення послідовності x довільного типу (для оберненого перетворення – `idct(x[, type, n, axis, norm, overwrite_x])`, а для синусного – `dst(x[, type, n, axis, norm, overwrite_x])` і `idst(x[, type, n, axis, norm, overwrite_x])` відповідно);

– `diff(x[, order, period, _cache])` – повертає похідну k -го порядку (або інтеграл) періодичної послідовності x ;

– `hilbert(x[, _cache])` – повертає перетворення Гільберта періодичної послідовності x (`ihilbert(x)` – для оберненого перетворення);

– `cs_diff(x, a, b[, period, _cache])` – повертає (a,b) -cosh/sinh псевдопохідну періодичної послідовності (`sc_diff(x, a, b[, period, _cache])`) – (a,b) -sinh/cosh, `ss_diff(x, a, b[, period, _cache])` – (a,b) -sinh/sinh, `cc_diff(x, a, b[, period, _cache])` – (a,b) -cosh/cosh);

– `shift(x, a[, period, _cache])` – змінює періодичну послідовність x за допомогою a : $y(u) = x(u+a)$;

– `fftshift(x[, axes])` – змінює постійну складову на центр спектра (обернений варіант – `ifftshift(x[, axes])`);

– `fftfreq(n[, d])` – повертає частоти дискретного перетворення Фур'є вибірок (`rfftfreq(n[, d])` – для використання з функціями `rfft`, `irfft`);

– функції упакування згорток підмодуля `scipy.fftpack.convolve`: `convolve(x, omega[, swap_real_imag, overwrite_x])`, `convolve_z(x, omega_real, omega_imag[, overwrite_x])`, `init_convolution_kernel(...)`,

`destroy_convolve_cache()`.

Статистичні функції та розподіли ймовірностей містяться в підпакеті `scipy.stats`.

Кожний розподіл є об'єктом класу `rv_continuous`, що має наступні методи зокрема: щільність розподілу ймовірностей `pdf(x, *args, **kwds)`, кумулятивна функція розподілу `cdf(x, *args, **kwds)`, функція надійності ($1 - cdf$) `sf(x, *args, **kwds)`, нецентральний момент розподілу n -ого порядку `moment(n, *args, **kwds)`, деяка статистика `stats(*args, **kwds)`, диференційна ентропія `entropy(*args, **kwds)`, обчислення очікуваних значень функції у відповідності з розподілом `expect([func, args, loc, ...])`.

Відповідно кожний дискретний розподіл є об'єктом класу `rv_discrete`, який має наступні методи: випадкові величини даного типу `rvs(*args, **kwargs)`, функція ймовірності міри `pmf(k, *args, **kwds)`, а також відповідні методи `cdf`, `sf`, `stats`, `moment`, `entropy`, `expect`.

Неперервні розподіли включають неперервні випадкові величини `alpha`, `anglit`, `arcsine`, `beta`, `betaprime`, `bradford`, `burr`, `cauchy`, `chi`, `chi2`, `cosine`, `dgamma`, `dweibull`, `erlang`, `expon`, `exponweib`, `exponpow`, `f`, `fatiguelife`, `fisher`, `foldcauchy`, `foldnorm`, `frechet_r`, `frechet_l`, `A`, `genlogistic`, `genpareto`, `genexpon`, `genextreme`, `gausshyper`, `gamma`, `gengamma`, `genhalflogistic`, `gilbrat`, `gompertz`, `gumbel_r`, `gumbel_l`, `halfcauchy`, `halflogistic`, `halfnorm`, `hypsecant`, `invgamma`, `invgauss`, `invweibull`, `johnsonsb`, `johnsonsu`, `ksone`, `kstwobign`, `laplace`, `logistic`, `loggamma`, `loglaplace`, `lognorm`, `lomax`, `maxwell`, `mielke`, `nakagami`, `ncx2`, `ncf`, `nct`, `norm`, `pareto`, `pearson3`, `powerlaw`, `powerlognorm`, `powernorm`, `rdist`, `reciprocal`, `rayleigh`, `rice`, `recipinvgauss`, `semicircular`, `t`, `triang`, `truncexpon`, `truncnorm`, `tukeylambda`, `uniform`, `vonmises`, `wald`, `weibull_min`, `weibull_max`, `wrapcauchy`.

Багатомірні розподіли включають багатомірні нормальні випадкові величини `multivariate_normal`, випадкові величини Діріхле `dirichlet`.

Дискретні розподіли включають дискретні випадкові величини `bernoulli`, `binom`, `boltzmann`, `dlaplace`, `geom`, `hypergeom`, `logser`, `nbinom`, `planck`, `poisson`, `randint`, `skellam`, `zipf`.

Статистичні функції підпакету включають обчислення декількох описових статистик переданого масиву `describe(a[, axis, ddof])`,

середньогоеметричного вздовж заданої вісі gmean(a[, axis, dtype]), середньогармонічного вздовж заданої вісі hmean(a[, axis, dtype]), експесу (Фішера або Пірсона) множини даних kurtosis(a[, axis, fisher, bias]), моди mode(a[, axis]), перевірку множини даних на нормальний експес kurtosistest(a[, axis]), нормальний розподіл normaltest(a[, axis]), асиметрії на відмінність від нормального розподілу skewtest(a[, axis]), обчислення n-го моменту moment(a[, moment, axis]), асиметрії skew(a[, axis, bias]), коефіцієнту варіації variation(a[, axis]), усічених середнього tmean(a[, limits, inclusive]), дисперсії tvar(a[, limits, inclusive]), мінімуму tmin(a[, lowerlimit, axis, inclusive]), максимуму tmax(a[, upperlimit, axis, inclusive]), обчислення кумулятивної гістограми частот cumfreq(a[, numbins, defaultreallimits, weights]), гістограми з використанням розподілу за стовпчиками histogram2(a, bins), розподілу діапазону на декілька стовпчиків з поверненням кількості об'єктів у кожному histogram(a[, numbins, defaultlimits, ...]), обчислення оцінки на заданій процентилі вхідної послідовності scoreatpercentile(a, per[, limit, ...]), байесівських довірчих інтервалів для середнього, дисперсії і стандартного відхилення bayes_mvs(data[, alpha]), урізання масиву до заданих значень threshold(a[, threshmin, threshmax, newval]), пропорційного відрізання сутностей з обох кінців масиву trimboth(a, proportiontocut[, axis]) або тільки одного trim1(a, proportiontocut[, tail]), виконання однонаправленого дисперсійного аналізу f_oneway(*args), обчислення коефіцієнту кореляції Пірсона pearsonr(x, y), Спірмена spearmanr(a[, b, axis]), рангу Кендалла kendalltau(x, y[, initial_lexsort]), обчислення лінії регресії linregress(x[, y]), Т-тестів для середнього однієї групи оцінок ttest_1samp(a, popmean[, axis]), двох незалежних груп ttest_ind(a, b[, axis, equal_var]), двох залежних груп оцінок ttest_rel(a, b[, axis]), тест Колмогорова-Смірнова kstest(rvs, cdf[, args, N, alternative, mode]), обчислення однонаправленого тесту хі-квадрат chisquare(f_obs[, f_exp, ddof, axis]) тощо.

Функції таблиць спряженості включають хі-квадрат тест незалежних змінних у таблиці спряженості chi2_contingency(observed[, correction, lambda_]), обчислення очікуваних частот з таблиці спряженості contingency.expected_freq(observed), обчислення точного критерію Фішера на таблиці спряженості 2x2 fisher_exact(table[, alternative]).

Модуль `scipy.stats.mstats` окрім містить великий набір функцій, які можуть застосовуватися до замаскованих масивів, більшість з яких аналогічна розглянутим вище в даному пакеті.

Бібліотека `scipy.ndimage` містить функції багатовимірного оброблення зображень: фільтри `scipy.ndimage.filters` (багатовимірний фільтр з гаусівською характеристикою `gaussian_filter(input, sigma[, order, ...])` і одновимірний `gaussian_filter1d(input, sigma[, axis, ...])`), N-вимірний фільтр Лапласа `generic_laplace(input, derivative2[, ...])` тощо), фільтри Фур'є `scipy.ndimage.fourier`, інтерполяції `scipy.ndimage.interpolation`, вимірювання `scipy.ndimage.measurements`, морфології `scipy.ndimage.morphology`.

Підпакет оброблення сигналів `scipy.signal` включає функції згортання (згортання двох N-вимірних масивів `convolve(in1, in2[, mode])`), кроскореляції двох N-вимірних масивів `correlate(in1, in2[, mode])`), бі-сплайнів (бі-сплайни порядку n `bspline(x, n)`, кубічні `cubic(x)`, квадратичні бі-сплайни `quadratic(x)` , обчислення сплайнів у новій множині точок `cspline1d_eval(cj, newx[, dx, x0])`), вейвлетів (комплексний вейвлет Морле `morlet(M[, w, s, complete])`), вейвлет «Мексиканський капелюх» `ricker(points, a)`), спектрального аналізу (обчислення спектральної щільності потужності з використанням періодограм `periodogram(x[, fs, window, nfft, detrend, ...])`), пошуку піків (`find_peaks_cwt(vector, widths[, wavelet, ...])`), фільтрування (медіанний фільтр для N-вимірного масиву `order_filter(a, domain, rank)`), вінерівський фільтр `wiener(im[, mysize, noise])`), згладжуючий фільтр з нескінченною імпульсною характеристикою та дзеркально-симетричними граничними умовами `symiirorder1((input, c0, z1 {, ...}),` вилучення лінійного тренда вздовж вісі `detrend(data[, axis, type, bp])`), лінійних систем з неперервним часом (обчислення частотної характеристики систем з неперервним часом `freqresp(system[, w, n])`), моделювання виходу системи `lsim(system, U, T[, X0, interp])`) та дискретним часом (моделювання виходу `dlsim(system, u[, t, x0])`), імпульсної характеристики `dimpulse(system[, x0, t, n])`), форм хвилі (генератор косинусоїdalьних сигналів `chirp(t, f0, t1, f1[, method, phi, vertex_zero])`), періодичний прямокутний імпульс `square(t[, duty])`, синусоїда Гаусового модулювання `gausspulse(t[, fc, bw, bwr, tpr, retquad, ...])`), вирізаючі функції (`get_window(window, Nx[, fftbins])` , `bartlett(M[, sym])`, `cosine(M[, sym])`, `gaussian(M, std[, sym])`) та функції

проектування фільтрів (`bilinear(b, a[, fs])`), яка повертає цифровий фільтр з аналогічного, використовуючи білінійне перетворення, `findfrequs(num, den, N)`, яка знаходить масив частот для обчислення відгуку фільтра, `frequs(b, a[, worN, plot])`, яка обчислює частотний відгук аналогового фільтра, `freqz(b[, a, worN, whole, plot])`, яка обчислює частотний відгук цифрового фільтра.

Підпакет спеціальних функцій `scipy.special` містить функції Ейрі (`airy(z)`), еліптичні функції і інтеграли (еліптичні функції Якобі `ellipj(u, m)`, обчислення еліптичного інтегралу першого типу `ellipk(m)`), функції Бесселя (функція Бесселя першого типу дійсного порядку `jv(v, z)`, інтеграли функцій Бесселя порядку 0 `itj0y0(x)`), статистичні функції необроблених даних (кумулятивна функція біноміального розподілу `bdtr(k, n, p)`, функція надійності біноміального розподілу `bdtrc(k, n, p)`, функція обчислення середнього нормального розподілу `nrdtrimn(p, x, std)`, функція обчислення середньоквадратичного відхилення нормального розподілу `nrdtrisd(p, x, mn)`, функція надійності Пуассона `pdtrc(k, m)`, комплементарна функція кумулятивного розподілу Колмогорова-Смірнова `smirnov(n, e)`), функції теорії інформації (поелементна функція обчислення ентропії `entr(x)` або відносної ентропії `rel_ent(x, y)`), гамма-функції (`gamma(z)`, логарифм абсолютноного значення гамма-функції `gammaln(z)`, знак гамма-функції `gammasgn(x)`, неповна гамма-функція `gammainc(a, x)`), функція помилок та інтеграли Френеля (`erf(z)`, яка повертає функцію помилок комплексних аргументів, `fresnel(z)`), функції Лежандра (`lpmv(m, v, x)`), еліпсоїdalну гармонічну функцію (`ellip_harm(h2, k2, n, p, s[, signm, signn])`), функції обчислення значень ортогональних поліномів (поліном Лаггера `assoc_laguerre(x, n[, k])`, багаточлени Чебишева `eval_chebyt(n, x[, out])`, `eval_chebyu(n, x[, out])`, `eval_chebyc(n, x[, out])`, `eval_chebys(n, x[, out])`), поліном Якобі `eval_jacobi(n, alpha, beta, x[, out])`), гіпергеометричні функції (Гаусса `hyp2fl(a, b, c, z)`, вироджена функція `hyp1fl(a, b, x)`), функції параболічного циліндра (`pbdv(v, x)`, `pbvv(v,x)`, `pbwa(a,x)`), функцію Маттьє (характеристичні значення `mathieu_a(m,q)` і `mathieu_b(m,q)` функції), сфероїdalні хвильові функції (витягнута сфероїdalна кутова функція першого типу та її похідна `pro_ang1(m,n,c,x)`, характеристичне значення витягнутої сфероїdalної функції `pro_cv(m,n,c)`), функції Кельвіна (`kelvin(x)`), комбінаторні функції

(`comb(N, k[, exact, repetition])` , яка визначає кількість комбінацій N речей, взятих по k за раз, `perm(N, k[, exact])`, яка визначає кількість перестановок N речей, взятих по k за раз), інші спеціальні (середньоарифметичне і середньогоеметричне `agm(a, b)`, масив чисел Бернуллі `bernoulli(n)`, біноміальні коефіцієнти `binom(n, k)`, функція факторіалу `factorial(n[, exact])`) та допоміжні функції (кубічний корінь `cbrt(x)`, `exp10(x)`, `exp2(x)`, перетворення з градусів у радіани `radian(d, m, s)`, косинус, синус, тангенс, котангенс кута, заданого в градусах, `cosdg(x)`, `sindg(x)`, `tandg(x)`, `cotdg(x)`, округлення до найближчого цілого `round(x)`).

При підготовці матеріалів даного пункту використано документацію [12] (зокрема розділи [<https://docs.scipy.org/doc/scipy/reference/linalg.html#module-scipy.linalg>], [<https://docs.scipy.org/doc/scipy/reference/integrate.html>], [<https://docs.scipy.org/doc/scipy/reference/optimize.html>], [<https://docs.scipy.org/doc/scipy/reference/interpolate.html>], [<https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.UnivariateSpline.html>], [<https://docs.scipy.org/doc/scipy/reference/cluster.vq.html>], [<https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html>], [<https://docs.scipy.org/doc/scipy/reference/sparse.csgraph.html>], [<https://docs.scipy.org/doc/scipy/reference/fftpack.html>], [https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.rv_continuous.html], [https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.rv_discrete.html], [<https://docs.scipy.org/doc/scipy/reference/stats.html>], [<https://docs.scipy.org/doc/scipy/reference/stats.mstats.html>], [<https://docs.scipy.org/doc/scipy/reference/ndimage.html>], [<https://docs.scipy.org/doc/scipy/reference/signal.html>], [<https://docs.scipy.org/doc/scipy/reference/special.html>]).

2.2.3 Бібліотека matplotlib

Бібліотека `matplotlib` призначена для побудови 2D графіків, написана мовою Python.

`matplotlib.pyplot` – колекція функцій у командному стилі, що дозволяють працювати в стилі MATLAB.

Функції `xlabel`, `ylabel` додають мітки на відповідні вісі координат (`matplotlib.axes.Axes.set_xlabel()` через API);

Функція `text` додає текст на графік за вказаними координатами

(x, y, text). Функція `figtext` додає текст на графік у відносних координатах (0-1).

Функція `annotate('annotation', xy=(x1, y1), xytext=(x2, y2))` дозволяє вставити анотацію для зазначеної точки графіка.

Функція `legend()` виводить підпис для відповідних ліній на графіку.

Функція `plot()` відображає лінії і/або маркери на графіку. Наприклад:

Python

```
plot(x, y, 'bo')
```

Рядок формату задається символами, що задають стиль лінії (таблиця 2.2) та колір (таблиця 2.3).

Таблиця 2.2 – Представлення стилю ліній у рядку формату

Символ	Опис	Символ	Опис
'-'	суцільна лінія	'3'	маркер у вигляді трієда, направленого вліво
'--'	пунктирна лінія	'4'	маркер у вигляді трієда, направленого вправо
'-.'	штрихпунктирна лінія	's'	квадратний маркер
'.'	пунктирна лінія	'p'	п'ятикутний маркер
'.'	маркер у вигляді крапки	'**'	маркер у вигляді зірочки
'.'	маркер у вигляді пікселя	'h'	маркер у вигляді шестикутника 1
'o'	маркер у вигляді кола	'H'	маркер у вигляді шестикутника 2

Продовження таблиці 2.2

Символ	Опис	Символ	Опис
'v'	маркер у вигляді знаку 'v'	'+'	маркер у вигляді знаку '+'
'^'	маркер у вигляді знаку '^'	'x'	маркер у вигляді знаку 'x'
'<'	маркер у вигляді знаку '<'	'D'	маркер у вигляді ромба
'>'	маркер у вигляді знаку '>'	'd'	маркер у вигляді малого ромба
'1'	маркер у вигляді тріода, направленого вниз	' '	маркер у вигляді вертикальної лінії
'2'	маркер у вигляді тріода, направленого вгору	'. -'	маркер у вигляді горизонтальної лінії

Таблиця 2.3 – Представлення кольору у рядку формату

Символ	Колір	Символ	Колір
'b'	Синій	'm'	Пурпурний
'g'	Зелений	'y'	Жовтий
'r'	Червоний	'k'	Чорний
'c'	Блакитний	'w'	Білий

Функція `hist()` призначена для побудови гістограм та приймає в якості аргументів масив даних і кількість ділянок, на які буде розбито масив (за замовчуванням дорівнює 10).

Для побудови стовпчастих діаграм призначена функція `bar()`, що приймає в якості аргументів послідовності координат x (лівий край стовпця) та y (висота). Ширина прямокутників за замовчуванням дорівнює 0.8. Параметри можна задати за допомогою наступних

ключових слів: width (ширина прямокутника), color (колір прямокутника). Для горизонтального зображення використовується функція barh().

Функція pie() призначена для побудови кругових діаграм та в якості аргументів приймає послідовність даних і параметри, які можна задавати за допомогою ключових слів: colors (кольори сегментів), labels (імена сегментів) тощо.

Функція scatter() призначена для побудови графіків розсіяння та приймає в якості параметрів координати маркерів і параметри, що задаються ключовими словами: s (розмір маркерів), c (колір маркерів), marker (тип маркера).

Для побудови в полярних координатах використовується функція polar(), в аргументах якої задаються кути та радіуси.

Функція triplot() призначена для побудови сітки з комірками у вигляді трикутників.

Функція phase_spectrum() дозволяє побудувати фазовий спектр.

Функція errorbar() дозволяє відобразити розсіювання навколо вірних значень та приймає в якості параметрів набір даних, відповідні значення помилок ует, вид графіка fmt, колір ліній інтервалу ecolor, товщину ліній linewidth, ширину обмежуючих ліній capsizes.

Функція show() відображає побудований графік.

Функції ylim() і xlim() встановлює межі вісі ординат, передаючи мінімальне та максимальні значення або за ключовими словами (ymin, ymax). Без аргументів функція повертає поточні межі вісі.

Функція vlines(x, ymin, ymax, colors='k', linestyles='solid', label='', hold=None, **kwargs) дозволяє будувати вертикальні лінії на графіку з абсцисою x та ординатами від $ymin$ до $ymax$. Для горизонтальних ліній існує відповідна функція hlines().

Функція subplots(nrows=1, ncols=1, sharex=False, sharey=False, squeeze=True, subplot_kw=None, gridspec_kw=None, **fig_kw) дозволяє створити графічне зображення, на якому розташовано декілька графіків. Параметри nrows та ncols задають кількість рядків та стовпців у сітці графіків. Функція повертає значення fig, об'єкт matplotlib.figure.Figure та ax, об'єкти підграфіків у заданій кількості.

Функція subplot() повертає новий підграфік за заданою позицією на сітці графіків.

Для того щоб задати властивості об'єкту використовується

функція `setp()`. Властивості задаються за допомогою ключових слів, а першим параметром задається безпосередньо об'єктом.

Функція `savefig()` дозволяє зберегти поточну фігуру. Перший параметр задає шлях до файлу.

Функція `fill_between(x, y1, y2=0, where=None, interpolate=False, hold=None, **kwargs)` дозволяє заповнити багатокутники між двома кривими. Функція `fill_betweenx()` відповідно заповнює простір між двома горизонтальними кривими. Функція `fill()` відображає заповнені багатокутники.

За допомогою функції `rc()` можна визначити необхідні налаштування.

Функція `figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, **kwargs)` створює нову фігуру.

Функція `close()` закриває вікно з фігурою, а `clf()` очищує поточну фігуру.

`matplotlib` підтримує вирази TeX у будь-яких текстових виразах. Наприклад, щоб внести у якості назви напис $\sigma_i = 15$, необхідно скористатись командою

Python

```
plt.title(r'$\sigma_i=15$')
```

Окрім розглянутої колекції функцій бібліотека `matplotlib` містить також наступні пакети:

- `matplotlib.cm` містить велику кількість карт кольорів та функцій для роботи з ними [https://matplotlib.org/stable/api/cm_api.html];

- `matplotlib.collections` – класи для зображення великих колекцій об'єктів з однаковими властивостями [https://matplotlib.org/stable/api/collections_api.html];

- `matplotlib.figure` забезпечує високорівневі об'єкти, що містять всі елементи графіків [https://matplotlib.org/stable/api/figure_api.html];

- `matplotlib.image` підтримує операції завантаження, масштабування та відображення зображень

- [https://matplotlib.org/stable/api/image_api.html];
 - `matplotlib.lines` містить всі класи 2D-ліній
- [https://matplotlib.org/stable/api/lines_api.html];
 - `matplotlib.text` містить класи для розміщення тексту на фігурах
- [https://matplotlib.org/stable/api/text_api.html];
 - `matplotlib.widgets` забезпечує віджети для роботи з будь-якими GUI тощо [https://matplotlib.org/stable/api/widgets_api.html].

Інструментарій `mpl_toolkits.mplot3d` додає можливості 3D-візуалізації у `matplotlib`, дозволяючи будувати стовпчасті діаграми, графіки розсіювання у трьохмірному вигляді. Для такої візуалізації призначенні спеціальні об'єкти `Axes3D`, які мають відповідні функції `plot`, `scatter`, `bar`, а також `plot_surface` для побудови графіка поверхні тощо.

Розглянемо деякий приклад побудови 3D-моделі:

Python

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib
import numpy as np
from matplotlib import cm
from matplotlib import pyplot as plt

step = 0.04
maxval = 0.8
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
r = np.linspace(0.5,1.15,50)
p = np.linspace(0,2*np.pi,50)
R,P = np.meshgrid(r,p)
X,Y = R*np.cos(P),R*np.sin(P)
Z = ((R**2 - 1)**2)
ax.plot_surface(X, Y, Z, rstride=1, cstride=1)
ax.set_zlim3d(0, 1)

plt.show()
```

На рис. 2.1 наведено результат роботи даної програми.

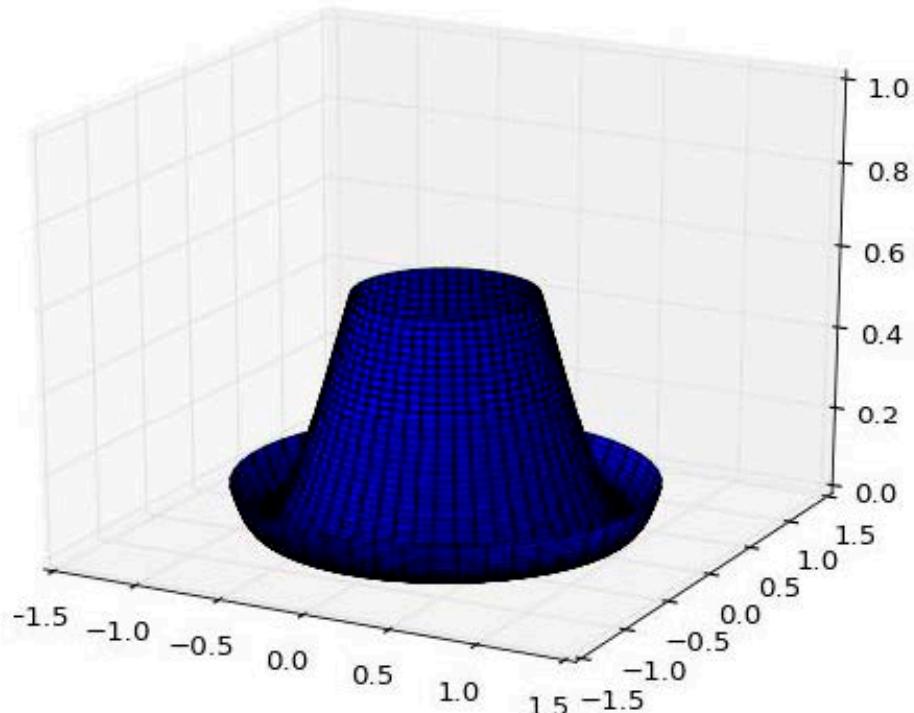


Рисунок 2.1 – Тривимірна модель

При підготовці матеріалів даного пункту використано документацію [20], [34] (зокрема розділи [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot], [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot], [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html], [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.vlines.html#matplotlib.pyplot.vlines], [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html], [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html], [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.pie.html], [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html], [https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.polar.html]),

[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.triplot.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.triplot.html), [\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.phase_spectrum.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.phase_spectrum.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.errorbar.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.errorbar.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.show.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.show.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.ylim.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.ylim.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xlim.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xlim.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplot.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplot.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.setp.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.setp.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.fill_between.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.fill_between.html), [\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.fill_betweenx.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.fill_betweenx.html),
[\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.fill.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.fill.html), [\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.rc.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.rc.html), [\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.figure.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.figure.html), [\[https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.close.html\]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.close.html)).

2.3. Завдання на лабораторну роботу

2.3.1. Ознайомитись з теоретичними відомостями, необхідними для виконання роботи.

2.3.2. Розробити програмне забезпечення у відповідності з індивідуальним завданням, узгодженим з викладачем.

2.3.2.1. Завдання:

а) визначити коефіцієнти функції $f(x) = axe^{bx}$ таким чином, щоб вона задоволяла наступним даним:

x	0.5	1.0	1.5	2.0	2.5
y	0.541	0.398	0.232	0.106	0.052

Обчислити стандартне відхилення.

б) знайти мінімум функції $F = 100(y - x^2)^2 + (1 - x)^2$ за допомогою метода Пауела. Пошук виконувати для різних початкових точок. Зобразити графік функції з необхідними позначеннями. Проаналізувати отримані результати.

2.3.2.2. Завдання:

а) для значень

x	1	2	3	4	5
y	13	15	12	9	13

виконати кубічну інтерполяцію сплайнами та визначити значення функції для заданих у файлі значень аргумента. Побудувати графік функції. Позначити задані точки.

б) знайти значення x , яке мінімізує функцію $f(x) = 1.6x^3 + 3x^2 - 2x$ за умови, що $x \geq 0$.

2.3.2.3. Завдання:

а) обчислити подвійний інтеграл:

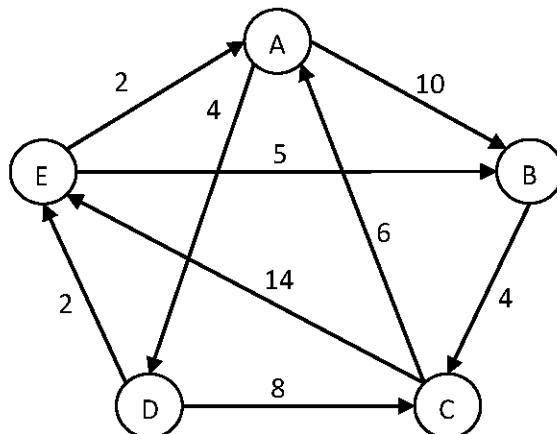
$$\int_0^1 \int_1^2 x^2 \exp(x + \sin(y)) \cos(y) dx dy.$$

б) знайти мінімум функції $F = 100(y - x^2)^2 + (1 - x)^2$ за допомогою метода Пауела. Пошук виконувати для різних початкових точок. Зобразити графік функції з необхідними позначеннями. Проаналізувати отримані результати.

2.3.2.4. Завдання:

а) розв'язати диференційне рівняння $y''' + 15y' = 7y'' + 9y$ та побудувати графік функції на заданому користувачем інтервалі. У межах тих самих вісей координат побудувати графік функції $y = e^{-x^2}$, обравши різні кольори.

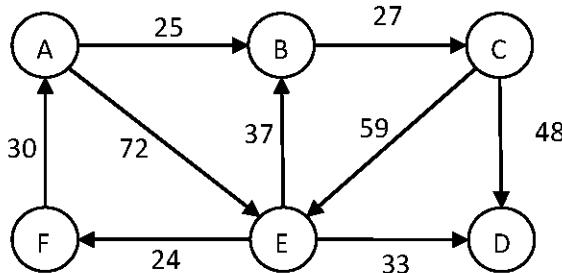
б) знайти найкоротші шляхи між всіма парами вершин графа, використовуючи відповідний алгоритм для роботи з графами.



2.3.2.5. Завдання:

а) розв'язати диференційне рівняння $y''' + 2y'' + y' = 0$, отримати значення загального розв'язку в заданому користувачем інтервалі та зберегти результати в файл формату CSV.

б) знайти найкоротші шляхи між заданого користувачем вершиною та всіма іншими вершинами та визначити сильно зв'язані компоненти в заданому графі, використовуючи відповідний алгоритм для роботи з графами.



2.3.2.6. Завдання:

а) обчислисти визначений інтеграл

$$\int_{\pi/7}^{\pi/4} \frac{\cos(2x) + \sin^2(x)}{\sin(3x)} dx.$$

б) побудувати графік функції та знайти її мінімум методом Нелдера-Міда з точністю до 10 десяткових знаків

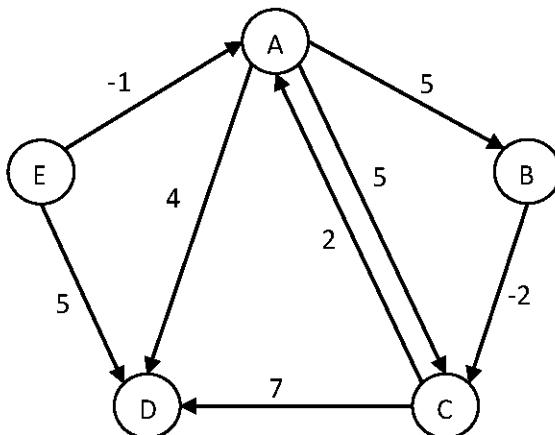
$$f(x, y) = x^2 + y^2 - 4x - y - xy.$$

2.3.2.7. Завдання:

а) побудувати інтерполяційний багаточлен Лагранжа, якщо відомі значення в наступному наборі точок

x	0	0.5	2.5	4.25	6.5
y	10.0	9.2	9.8	10.3	8.6

б) знайти найкоротші шляхи між однією з вершин графа, заданою користувачем, та всіма іншими вершинами, використовуючи відповідний алгоритм для роботи з графами.



Після обчислення вилучити елементи більше заданого значення.

2.3.2.8. Завдання:

- а) визначити коефіцієнти функції $f(x) = axe^{bx}$ таким чином, щоб вона задовольняла наступним даним:

x	0.5	1.0	1.5	2.0	2.5
y	0.541	0.398	0.232	0.106	0.052

Обчислити стандартне відхилення.

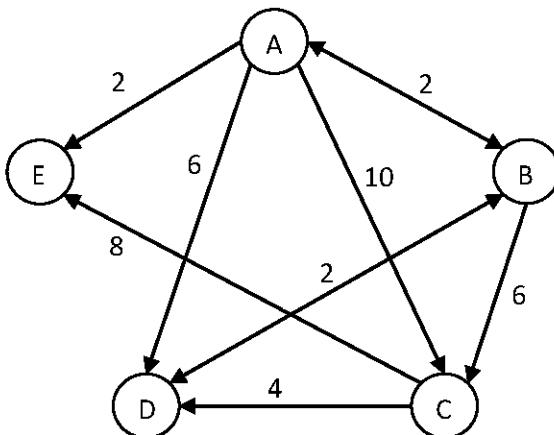
- б) знайти мінімум функції $F = 100(y - x^2)^2 + (1 - x)^2$ за допомогою метода Пауела. Пошук виконувати для різних початкових точок. Зобразити графік функції з необхідними позначеннями. Проаналізувати отримані результати.

2.3.2.9. Завдання:

- а) побудувати функцію та знайти її глобальний екстремум

$$z = 3x^2 + xy + 2y^2 - x - 4y.$$

- б) виконати пошук в глибину та пошук в ширину для заданого графа:



2.3.2.10. Завдання:

а) для значень

x	1	2	3	4	5
y	13	15	12	9	13

виконати кубічну інтерполяцію сплайнами та визначити значення функції для заданих у файлі значень аргумента. Побудувати графік функції. Позначити задані точки.

б) знайти значення x , яке мінімізує функцію $f(x) = 1.6x^3 + 3x^2 - 2x$ за умови, що $x \geq 0$.

2.3.2.11. Завдання:

а) обчислити подвійний інтеграл:

$$\int_0^1 \int_1^2 x^2 \exp(x + \sin(y)) \cos(y) dx dy.$$

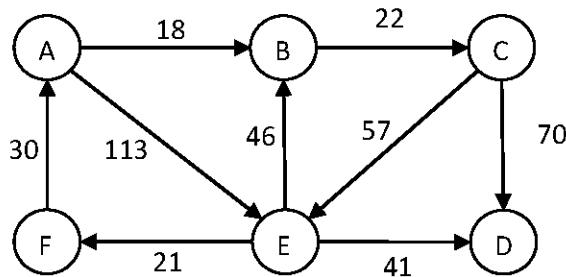
б) знайти мінімум функції $F = 100(y - x^2)^2 + (1 - x)^2$ за допомогою метода Пауела. Пошук виконувати для різних початкових точок. Зобразити графік функції з необхідними позначеннями. Проаналізувати отримані результати.

2.3.2.12. Завдання:

а) розв'язати диференційне рівняння $y''' + 2y'' + 2y' = 0$, отримати значення загального розв'язку в заданому користувачем інтервалі та зберегти результати в файл формату CSV.

б) знайти накоротшиі шляхи між заданою користувачем

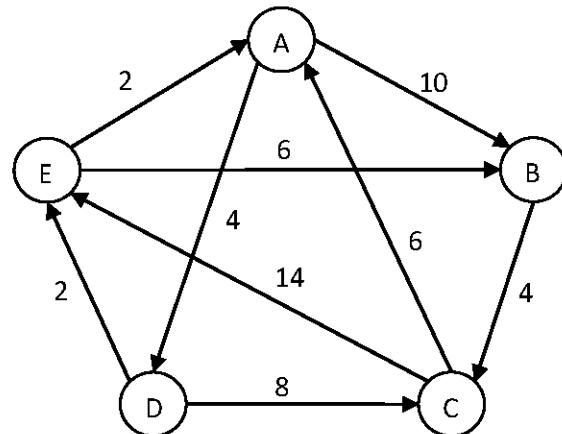
вершиною та всіма іншими вершинами та визначити сильно зв'язані компоненти в заданому графі, використовуючи відповідні алгоритми для роботи з графами.



2.3.2.13. Завдання:

а) розв'язати диференційне рівняння $y''' + 15y' = 7y'' + 9y$ та побудувати графік функції на заданому користувачем інтервалі. У межах тих самих вісей координат побудувати графік функції $y = e^{-x^2}$, обравши різні кольори.

б) знайти найкоротші шляхи між всіма парами вершин графа, використовуючи відповідний алгоритм для роботи з графами.

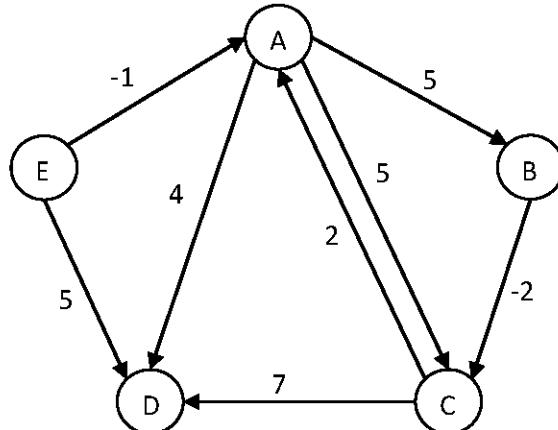


2.3.2.14. Завдання:

а) побудувати інтерполяційний багаточлен Лагранжа, якщо відомі значення в наступному наборі точок

x	0	0.125	0.62	1.06	1.62
y	2.5	2.3	2.85	2.507	2.165

б) знайти найкоротші шляхи між однією з вершин графа, заданою користувачем, та всіма іншими вершинами, використовуючи відповідний алгоритм для роботи з графами.



Після обчислення вилучити елементи більше заданого значення.

2.3.2.15. Завдання:

а) обчислисти визначений інтеграл

$$\int_{\pi/7}^{\pi/4} \frac{\cos(2x) + \sin^2(x)}{\sin(3x)} dx.$$

б) побудувати графік функції та знайти її мінімум методом Неллдера-Міда з точністю до 10 десяткових знаків

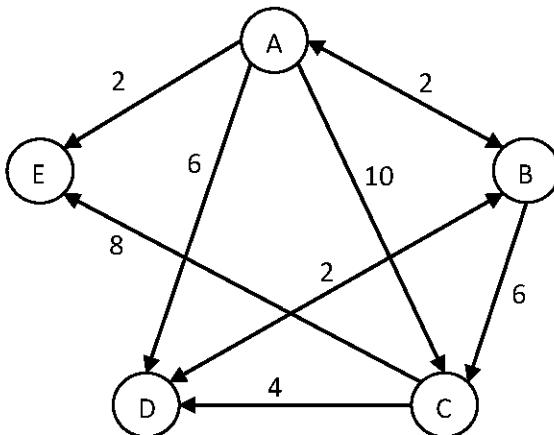
$$f(x, y) = x^2 + y^2 - 4x - y - xy.$$

2.3.2.16. Завдання:

а) побудувати функцію та знайти її глобальний екстремум

$$z = 3x^2 + xy + 2y^2 - x - 4y.$$

б) виконати пошук в глибину та пошук в ширину для заданого графа:



2.3.2.17. Завдання:

а) для значень

x	1	2	3	4	5
y	13	15	12	9	13

виконати кубічну інтерполяцію сплайнами та визначити значення функції для заданих у файлі значень аргумента. Побудувати графік функції. Позначити задані точки.

б) знайти значення x , яке мінімізує функцію $f(x) = 1.6x^3 + 3x^2 - 2x$ за умови, що $x \geq 0$.

2.3.2.18. Завдання:

а) розв'язати диференційне рівняння $y''' + 15y' = 7y'' + 9y$ та побудувати графік функції на заданому користувачем інтервалі. У межах тих самих вісей координат побудувати графік функції $y = e^{-x^2}$, обравши різні кольори.

б) знайти мінімум функції $F = 100(y - x^2)^2 + (1 - x)^2$ за допомогою метода Пауела. Пошук виконувати для різних початкових точок. Зобразити графік функції з необхідними позначеннями. Проаналізувати отримані результати.

2.3.2.19. Завдання:

а) обчислити визначеній інтеграл

$$\int_{\pi/7}^{\pi/4} \frac{\cos(2x) + \sin^2(x)}{\sin(3x)} dx.$$

б) побудувати графік функції та знайти її мінімум методом Нелдера-Міда з точністю до 10 десяткових знаків

$$f(x, y) = x^2 + y^2 - 4x - y - xy.$$

2.3.2.20. Завдання:

а) побудувати функцію та знайти її глобальний екстремум

$$z = 3x^2 + xy + 2y^2 - x - 4y.$$

б) визначити коефіцієнти функції $f(x) = axe^{bx}$ таким чином, щоб вона задоволяла наступним даним:

x	0.5	1.0	1.5	2.0	2.5
y	0.541	0.398	0.232	0.106	0.052

Обчислити стандартне відхилення.

2.3.3. Виконати тестування розробленого програмного забезпечення.

2.3.4. Оформити звіт.

2.3.5. Відповісти на контрольні запитання.

2.4. Зміст звіту

- 2.4.1. Мета роботи.
- 2.4.2. Завдання до роботи.
- 2.4.3. Текст розробленого програмного забезпечення.
- 2.4.4. Результати тестування: вхідні дані та результати роботи програми.
- 2.4.5. Інтерфейс роботи з програмою в декількох режимах.
- 2.4.6. Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

2.5. Контрольні запитання

- 2.5.1. Які засоби надаються бібліотекою NumPy?
- 2.5.2. Які засоби надаються бібліотекою SciPy?
- 2.5.3. Для чого необхідна бібліотека matplotlib?
- 2.5.4. Які засоби містить бібліотека matplotlib?
- 2.5.5. Які графіки можна побудувати за допомогою бібліотеки matplotlib?

2.5.6. Які існують налаштування під час створення графіків у бібліотеці `matplotlib`?

2.5.7. Яким чином побудувати декілька графіків під час одного виведення?

ЛАБОРАТОРНА РОБОТА № 3
РОЗРОБЛЕННЯ ВЕБДОДАТКІВ ТА РЕАЛІЗАЦІЯ
ДОСТУПУ ДО СИСТЕМ КЕРУВАННЯ БАЗАМИ ДАНИХ
ЧЕРЕЗ ПРОГРАМНІ ІНТЕРФЕЙСИ

3.1. Мета роботи

3.1.1 Навчитися використовувати програмні інтерфейси для доступу до баз даних.

3.1.2 Навчитися розробляти вебдодатки за допомогою фреймворка Django.

3.2. Короткі теоретичні відомості

3.2.1 Програмний інтерфейс для роботи з системою керування базами даних MySQL

Пакет MySQLdb призначений для забезпечення програмного інтерфейсу для використання системи керування базами даних MySQL у програмах, написаних мовою Python. Програмний інтерфейс аналогічний даному модулю, написаний на Python та призначений для інтерпретаторів CPython, PyPy, IronPython, Jython, реалізовано в пакеті PyMySQL.

Функція `connect(parameters...)` – конструктор для створення з'єднання до бази даних, який повертає відповідний об'єкт. Параметри функції зокрема включають ім'я хосту `host`, з яким встановлюється з'єднання, логін `user` і пароль `passwd` для аутентифікації, назву бази даних `db`, з якою встановлюється з'єднання, `port` – TCP-порт MySQL сервера, `unix_socket` – розташування сокету UNIX, `conv` – словник перетворення типів, `compress` – протокол стискання, `connect_timeout` – розрив, якщо з'єднання не встановлено за задану кількість секунд, `read_default_file` – конфігураційний файл MySQL для читання, `cursorclass` – клас `cursor`, який використовує функція `cursor()`, поки не перевантажено, `use_unicode`: якщо істинно, то стовпчики `CHAR`, `VARCHAR`, `TEXT` повертаються як рядки Unicode, використовуючи сконфігурковану множину символів, `charset` – задає множину символів.

Приклад з'єднання з базою даних MySQL:

Python

```
from MySQLdb import _mysql
mydb = _mysql.connect(host = 'localhost', user = 'mykyta',
passwd = 'mysecret', db = 'mydb')
```

Об'єкти з'єднання містять наступні методи:

- commit(): якщо база даних та таблиці підтримують транзакції, то фіксує поточну транзакцію;
- rollback(): відхиляє поточну транзакцію;
- cursor([cursorclass]): створює новий курсор для виконання запитів;
- close(self): закриття з'єднання з базою даних;
- ping(self, reconnect=True): пінгувати сервер;
- select_db(self, db): встановлює поточну базу даних.

Об'єкти класу Cursor мають наступні методи:

- callproc(self, procname, args=()) – виконати збережену процедуру з заданими аргументами;
- close(self) – закрити курсор;
- execute(self, query, args=None) – виконати запит;
- executemany(self, query, args) – виконує один запит для декількох наборів даних;
- fetchall(self) – видобуває всі записи;
- fetchmany(self, size=None) – видобуває декілька записів;
- fetchone(self) – видобуває наступний запис;
- nextset(self) – пересуває курсор на наступну множину результатів.

Наприклад, для виконання запиту до таблиці бази даних можна скористатися наступним скриптом:

Python

```
from MySQLdb import _mysql
db=_mysql.connect(passwd="moonpie",db="thangs")
c=db.cursor()
```

Python

```
max_price=5
c.execute("""SELECT spam, eggs, sausage FROM
breakfast
WHERE price < %s""", (max_price,))
```

Для того щоб переглянути один запис з результату запиту, можна виконати наступну команду:

Python

```
>>> c.fetchone()
(3L, 2L, 0L)
```

Клас DatabaseError є класом виключення, яке створюється у випадку проблем з базою даних, а InterfaceError – у випадку проблем зі з'єднанням. Підтримуються наступні типи помилок: DataError, IntegrityError, InternalError, NotSupportedError, OperationalError, ProgrammingError тощо.

При підготовці матеріалів даного пункту використано документацію [18], [27]-[28].

3.2.2 Фреймворк Django для розроблення вебдодатків

Django – популярний потужний фреймворк для розроблення вебдодатків, написаних мовою Python, що використовує шаблон проектування MVC.

Після встановлення Django в системі з'явиться скрипт django-admin.py для оброблення задач скафолдінгу. Для того щоб створити файли проекту, необхідно скористатися командою

CMD

```
django-admin.py startproject mysite
```

Створений проект має наступну структуру:

```
mysite/
    manage.py #посилання на скрипт django-admin з наперед
               #установленими змінними оточення, що вказують
               #на проект
mysite/
    __init__.py
    settings.py #налаштування проекту
    urls.py #містить URL для відображення представлень
    wsgi.py #WSGI-обгортка додатку
```

Для створення додатку використовується команда:

CMD

```
python manage.py startapp polls
```

Створений додаток має наступну структуру:

```
./polls
    __init__.py
    admin.py #містить модель для адміністративного інтерфейсу
    migrations/ #містить файли міграцій
    models.py #містить Django ORM-моделі додатку
    tests.py #містить модульні та інтеграційні тести
    views.py #містить код представлень
```

Django підтримує системи керування базами даних MySQL, PostgreSQL, SQLite. Для того щоб задати відповідну систему, необхідно встановити відповідні значення у секції DATABASES файла settings.py. У даному файлі задається також перелік встановлених додатків (у перелік необхідно додати створений додаток (‘‘polls’’)).

Моделі Django відображають таблиці бази даних та надають місце для інкапсуляції бізнес-логіки. Усі моделі є нащадками базового

класу `django.db.models.Model` та містять поля визначень. Наприклад, файл `polls/models.py`:

Python

```
from django.db import models

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    question = models.ForeignKey(Question)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

В якості типів полів можуть використовуватися: `AutoField`, `IntegerField`, що автоматично збільшується), `BigIntegerField`, `BinaryField`, `BooleanField`, `CharField`, `CommaSeparatedIntegerField`, `DateField`, `DateTimeField`, `DecimalField` (десяtkове число з фіксованою точністю), `EmailField`, `FileField`, `FilePathField`, `FloatField`, `ImageField`, `IntegerField`, `IPAddressField`, `GenericIPAddressField`, `NullBooleanField`, `PositiveIntegerField`, `PositiveSmallIntegerField`, `SlugField`, `SmallIntegerField`, `TextField`, `TimeField`, `URLField`.

Для завдання відношень використовуються `ForeignKey` («множина-один»), `ManyToManyField` («множина-до-множини»), `OneToOneField` («один-до-одного»):

Python

```
class Manufacturer(models.Model):
    # ...
    pass

class Car(models.Model):
    manufacturer = models.ForeignKey(Manufacturer)
```

Для класів можуть бути визначені методи моделі або змінено поведіння бази даних за допомогою методів `save()` та `delete()`. Як і класи в Python моделі в Django можуть наслідуватися одна від одної.

Метод `raw(raw_query, params=None, translations=None)` може використовуватися для того, щоб виконувати запити SQL, які повертають екземпляри. Наприклад:

Python

```
Person.objects.raw('SELECT id, first_name, last_name,
birth_date FROM myapp_person')
```

Стандартне з'єднання бази даних повертається за допомогою об'єкта `django.db.connection`.

Після створення моделі необхідно доповнити базу даних новими таблицями за допомогою команд

CMD

```
python manage.py makemigrations
python manage.py migrate
```

Для того щоб взаємодіяти зі створеною моделлю за допомогою інтерактивної оболонки необхідно скористатися командою

Python

```
python manage.py shell
>>> from polls.models import Question, Choice
>>> Question.objects.all()
[]
>>> from django.utils import timezone
>>> q = Question(question_text="What's new?", pub_date=timezone.now())
>>> q.save()
```

Для того щоб запустити вбудований вебсервер, необхідно скористатися командою

CMD

```
python manage.py runserver
```

Після цього в браузері можна запустити `http://127.0.0.1:8000/` та отримати доступ до сервера.

Представлення Django на отриманий запит повертають відповідь. Будь-який об'єкт мови Python може бути представленням. Єдина жорстка і необхідна вимога полягає в тому, що об'єкт Python, який викликається, повинен приймати об'єкт запиту в якості першого аргументу. Це означає, що мінімальне представлення буде дуже простим:

Python

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, World")
```

Представлення-класи формуються наступним чином:

Python

```
from django.http import HttpResponse
from django.views.generic import View

class MyView(View):

    def get(self, request, *args, **kwargs):
        return HttpResponse("Hello, World")
```

Існують наступні стандартні представлення:

- базові django.views.generic.base (базовий View, TemplateView, який інтерпретує даний шаблон з використанням параметрів з URL, що містять контекст);

- параметризовані django.views.generic (detail.DetailView, що під час виконання у self.object містить об'єкт, над яким виконуються дії, та list.ListView, що є сторінкою, яка представляє перелік об'єктів self.object_list);

- django.views.generic.edit для редагування контенту (FormView, що відображує форму, CreateView, що відображує форму для створення об'єкту, повторного відображення форми та збереження об'єкту, UpdateView, що відображує форму для редагування існуючого об'єкту, повторного відображення форми та збереження змін до об'єкту, DeleteView, що відображує сторінку підтвердження та вилучає існуючий об'єкт);

- прості суміші django.views.generic.base (ContextMixin, що повертає словник, який представляє контекст шаблону, TemplateResponseMixin), суміші для одного об'єкту django.views.generic.detail (SingleObjectMixin, що забезпечує механізм для поліпшення об'єкта, який асоціюється з запитом, SingleObjectTemplateResponseMixin), суміші для декількох об'єктів django.views.generic.list (MultipleObjectMixin, що використовується для відображення переліку об'єктів, MultipleObjectTemplateResponseMixin, що формує відповіді на основі шаблонів), редакуючі суміші django.views.generic.edit (FormMixin, що забезпечує створення і відображення форм).

Пакет django.shortcuts містить деякі допоміжні функції, які охоплюють декілька рівнів MVC:

- render(request, template_name[], dictionary[], context_instance[], content_type[], status[], current_app[], dirs]) – об'єднує даний шаблон з заданим контекстним словником та повертає об'єкт HttpResponseRedirect;

- redirect(to, [permanent=False,]*args, **kwargs) – повертає HttpResponseRedirect на відповідний URL для переданих аргументів.

Конфігурація URL вказує Django, яким чином за адресою запиту знайти Python-код.

Деякі загальні представлення повинні мати шаблони, в загальному випадку розташовані за адресою ім'я_додатку/templates/ім'я_додатку/ім'я_шаблону.

Змінні в шаблонах представляються у вигляді {{ variable }}. Фільтри застосовуються у вигляді {{ variable|filter }}. Керуючий код визначається у вигляді {% tag %}.

Розглянемо детальніше приклад, для якого вище було створено модель і який передбачає створення набору опитувань, кожне з яких складається з деякої кількості варіантів, дозволяючи переглянути опитування, його результати та проголосувати.

Текст файлу /mysite/urls.py, де визначаються всі URL, які розпізнає даний проект:

Python

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('polls/', include('polls.urls')),
    path('admin/', admin.site.urls),
]
```

Текст файлу /mysite/polls/urls.py, який визначає всі URL, що розпізнаються даним додатком:

Python

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('<int:question_id>', views.detail, name='detail'),
    path('<int:question_id>/results/', views.results,
         name='results'),
```

Python

```
    path('<int:question_id>/vote/', views.vote, name='vote'),
]
```

Текст файлу /mysite/polls/views.py, який визначає всі наявні представлення (перегляду переліку опитувань, конкретного опитування, його результатів та голосування):

Python

```
from polls.models import Question, Choice
from django.http import HttpResponseRedirect,
HttpResponse
from django.template import RequestContext, loader
from django.shortcuts import get_object_or_404, render
from django.core.urlresolvers import reverse
from django.views import generic

class IndexView(generic.ListView):
    template_name = 'polls/index.html'
    context_object_name = 'latest_question_list'

    def get_queryset(self):
        """Return the last five published questions."""
        return Question.objects.order_by('-pub_date')[:5]

class DetailView(generic.DetailView):
    model = Question
    template_name = 'polls/detail.html'

class ResultsView(generic.DetailView):
    model = Question
    template_name = 'polls/results.html'

    def vote(request, question_id):
        p = get_object_or_404(Question, pk=question_id)
```

Python

```

try:
    selected_choice =
        p.choice_set.get(pk=request.POST['choice'])
except (KeyError, Choice.DoesNotExist):
    return render(request, 'polls/detail.html', {
        'question': p,
        'error_message': "You didn't select a choice.",
    })
else:
    selected_choice.votes += 1
    selected_choice.save()
    return HttpResponseRedirect(reverse('polls:results',
        args=(p.id,)))

```

Текст файлу index.html:

Template

```

<b>Опитування</b>
<ul>
    {% for question in latest_question_list %}
        <li><a href="{% url 'polls:detail' question.id %}">{{ question.question_text }}</a></li>
    {% endfor %}
</ul>

```

Текст файлу results.html:

Template

```

<h1>{{ question.question_text }}</h1>
<ul>
    {% for choice in question.choice_set.all %}
        <li>{{ choice.choice_text }} -- {{ choice.votes }} vote{{

```

Template

```
choice.votes|pluralize }}</li>
{%
endfor %}
</ul>
<a href="{% url 'polls:detail' question.id %}">Проголосувати знову?</a>
```

Текст файлу detail.html:

Template

```
<h1>{{ question.question_text }}</h1>
{% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
<form action="{% url 'polls:vote' question.id %}" method="post">
    {% csrf_token %}
    {% for choice in question.choice_set.all %}
        <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}"/>
        <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br />
    {% endfor %}
    <br>
    <input type="submit" value="Try to vote" />
</form>
```

На рис. 3.1 представено результат відображення одного з опитувань.

На рис. 3.2 представлені результати голосування, які з'являються на екрані після вибору варіанту та натиснення на кнопку ‘Try to vote’.

Яка команда виграє чемпіонат Іспанії?

- Атлетико Мадрид
- Барселона
- Реал Мадрид
- Валенсія
- Севілья

[Try to vote](#)

Рисунок 3.1 – Інтерфейс розробленого вебдодатку з опитуванням

Яка команда виграє чемпіонат Іспанії?

- Атлетико Мадрид -- 5 votes
- Барселона -- 3 votes
- Реал Мадрид -- 7 votes
- Валенсія -- 0 votes
- Севілья -- 0 votes

[Проголосувати знову?](#)

Рисунок 3.2 – Інтерфейс розробленого вебдодатку з результатами опитування

Django містить пакет `django.contrib.staticfiles` для керування статичними файлами і генерування шляхів до них. Статичні файли, визначені на рівні додатку, зберігаються в піддиректорії `static` директорії додатку.

Пакет `django.contrib.staticfiles` Django включає тег шаблона, який полегшує створення посилань на статичні файли з шаблону, організовуючи в результаті абсолютну URL-адресу статичного файлу. Для цього потрібно використати спочатку тег шаблону:

Template

```
{% load static %}
```

Після завантаження бібліотеки для роботи зі статичними файлами можна посилатись на файл, використовуючи тег `static`:

Template

```
<link rel="stylesheet" type="text/css" href="{% static  
'polls/style.css' %}">
```

Для створення адміністраторського облікового запису для підключення до адміністративної панелі необхідно використовувати команду

CMD

```
python manage.py createsuperuser
```

При підготовці матеріалів даного пункту використано документацію [14], [29] (зокрема розділи [<https://docs.djangoproject.com/en/4.0/intro/tutorial01/>], [<https://docs.djangoproject.com/en/4.0/intro/tutorial02/>], [<https://docs.djangoproject.com/en/4.0/intro/tutorial03/>], [<https://docs.djangoproject.com/en/4.0/intro/tutorial04/>], [<https://docs.djangoproject.com/en/4.0/intro/tutorial05/>], [<https://docs.djangoproject.com/en/4.0/intro/tutorial06/>], [<https://docs.djangoproject.com/en/4.1/topics/http/shortcuts/>], [<https://docs.djangoproject.com/en/4.1/topics/class-based-views/mixins/>]).

3.3. Завдання на лабораторну роботу

3.3.1. Ознайомитись з теоретичними відомостями, необхідними для виконання роботи.

3.3.2. Узгодити з викладачем індивідуальне завдання:

3.3.2.1. Вебдодаток планування завдань. Користувачам надається можливість визначати завдання, задаючи для кожного з них назву, опис, час початку та завершення, відсоток виконання. Для кожного завдання може встановлюватися пріоритет, група завдань, а також ресурси, необхідні для виконання даного завдання. Користувач має можливість обмежувати перегляд завдань відсотком

виконання, переглядати завдання за вказаний день, переглядати ресурси, що використовуються.

3.3.2.2. Вебдодаток обміну речами. Користувачі можуть визначати власні пропозиції наявних речей з описом їх і для кожної речі визначати речі, на які вони згідні поміняти її. Якщо на річ, яку виставляє користувач є пропозиція обміну, то в процесі її оформлення має визначатися перелік таких пропозицій обміну, з яких можна обрати одну для реалізації (у такому випадку обом користувачам надходить відповідне повідомлення). Пропозиції можна скасовувати.

3.3.2.3. Інфраструктура велосервісу включає велостанції, велосипеди та його працівників. Вебдодаток дозволяє переглядати наявні велосипеди (загалом та на обраній велостанції), надаючи інформацію про них, статистику їх використання та поточний стан, додавати нові та вилучати з користування існуючі, брати велосипеди на прокат (з переліку наявних на велостанції), повернати їх, розраховувати вартість користування велосипедом та сплачувати за користування.

3.3.2.4. Вебдодаток підтримки медичного забезпечення. Користувачами вебдодатку є лікарі, пацієнти та аптекарі. Пацієнти можуть переглядати тільки власні медичні картки та вносити інформацію про себе. Лікар може вносити всю необхідну інформацію в медичну картку та формувати рецепти на продаж ліків. Аптекар може отримати доступ до всіх рецептів та продати ліки за відповідним рецептом.

3.3.2.5. Вебдодаток підтримки роботи ресторану. Клієнти можуть переглядати інформацію про наявні в меню позиції, оформляти замовлення та резервувати місця в ресторані. Оформлення замовлень також виконується офіціантами з зазначенням всієї відповідної інформації. Кухарі отримують інформацію про замовлення, виконання яких очікується. Будь-який користувач може відстежувати стан будь-якого замовлення.

3.3.2.6. Вебдодаток новинного агентства, що дозволяє додавати новини менеджерам, переглядати їх звичайним користувачам, коментувати, оцінювати. Менеджери можуть вилучати коментарі. Перегляд новин може виконуватися як за часом публікації, так і за популярністю або середніми оцінками статей (стандартний вибір

може задаватися в налаштуваннях індивідуально). Для кожного менеджера розраховується статистика за його діяльністю.

3.3.2.7. Вебдодаток проведення іспитів на отримання водійського посвідчення. Адміністратор визначає перелік тестових запитань, відповідей на них та правильні відповіді на кожне запитання. Користувачі здають іспит з обмеженнями в часі та обмеженнями на невдалу загальну кількість спроб. Набір запитань для кожного тесту формується випадковим чином. У результаті іспиту користувач отримує повідомлення про те, чи успішно склав іспит, та свій загальний результат.

3.3.2.8. Вебдодаток продажу квитків на потяг. Для кожного потяга визначається його маршрут слідування, вартість проїзду, кількість місць, час та дні слідування. Користувач задає свій маршрут для виконання пошуку або визначає потяг за номером, після чого отримує дані про наявні потяги (разом з їх повним маршрутом слідування) та вільні місця, з яких може обрати необхідні для бронювання.

3.3.2.9. Вебдодаток керування нотатками. Користувач може формувати нотатки: задавати для них текст, назву, теги, колір. Для кожної нотатки може прив'язуватися дата. Вебдодаток повинен надавати календар, за яким можна переглядати нотатки, прив'язані до конкретних дат, або повний перелік нотаток, або перелік нотаток за тегами. Користувач може переглядати тільки власні нотатки. Нотатки можна архівувати та відновлювати з архіву.

3.3.2.10. Вебдодаток планування та обчислення бюджету поїздки. Користувач може планувати бюджет поїздки, визначаючи витрати та дляожної статті витрат групу (проживання, проїзд, їжа, музей, інші розваги тощо). У процесі поїздки всі сплачені суми мають фіксуватися. Має надаватися можливість переглядати відхилення між запланованим бюджетом та фактичними результатами. Поїздки можуть бути груповими і оброблюватися декількома учасниками.

3.3.2.11. Вебдодаток підтримки оренди житла в містах України. Власники житла можуть розміщувати пропозиції надання житла в оренду. Клієнти можуть виконувати пошук житла за заданими параметрами, бронювати житло, залишати відгуки. Власники житла можуть залишати відгуки про клієнтів після виконання бронювання.

Рішення про здавання в оренду житла конкретному клієнту приймає власник на основі поданої заявки на бронювання.

3.3.2.12. Вебдодаток збору коштів на реалізацію проектів. Автори можуть визначати власні проекти та кошти, необхідні на їх фінансування, для кожного з яких встановлюється обмежений період збору коштів, протягом якого користувачі можуть перераховувати кошти на його фінансування. Якщо на момент закінчення періоду збору коштів необхідна сума не зібрана, то кошти повертаються користувачам. Якщо необхідна сума зібрана раніше, то автор може закінчити збір коштів або не зупиняти його.

3.3.2.13. Вебдодаток, що реалізує сплату за комуналні послуги. Адміністратори вносять інформацію в систему про тарифи на користування окремими комунальними послугами. Користувачі вносять інформацію про дані своїх лічильників. Система розраховує суму, яку необхідно сплатити за комуналні послуги. Користувачі можуть переглядати стан власних рахунків, сплачувати їх, переглядати історію.

3.3.2.14. Вебдодаток керування складом, що надає можливість менеджерам реєструвати новий товар, вилучати товари зі складу, змінювати їх наявну кількість, відстежувати наявний товар, місця його розташування на складі та строки його придатності. Клієнтам має надаватися інформація тільки про кількість та вартість товарів. Менеджери мають бути проінформовані про товари, строк придатності яких завершується або завершено.

3.3.2.15. Вебдодаток керування успішністю академічної групи, який повинен надавати можливість відстежувати відвідування занять студентами, їх поточні результати та результати складання сесії. Викладачі можуть виставляти оцінки та вносити дані про відвідуваність тільки за власними предметами, отримувати статистичні дані про кількість боржників, успішність складання сесії тощо.

3.3.2.16. Вебдодаток підтримки роботи з документами на підприємстві. Кожен документ реєструється, при чому повторна реєстрація заборонена. Для кожного документа може визначатися задача, строки її виконання та відповідальні. Деяким працівникам може бути заборонена робота з окремими документами. Працівники можуть працювати з документами, змінювати статус задач. При вході

працівника в систему має відображатися перелік визначених для нього задач. Підтримується пошук документів.

3.3.2.17. Вебдодаток, що дозволяє виконувати пошук транспорту (літаки, автобуси, потяги), яким можна переміститися між деякими двома заданими містами. Для кожного рейсу визначається перелік місць, через які він проходить, для кожного місця задаючи відносний момент часу та вартість такого переміщення. Враховувати маршрути з пересадками. Визначати маршрути для кожного виду транспорту, сортуючи їх за вартістю. Для кожного виду транспорту залишити найдешевший та найшвидший маршрут.

3.3.2.18. Вебдодаток підтримки роботи бібліотеки. Бібліотекарі визначають перелік наявних книг, кількість екземплярів тощо, можуть переглядати інформацію про наявні в поточний момент книги. Читачі можуть резервувати книги для читання та повернати їх через деякий час. Якщо час користування книгою перевищує граничний термін, то з читача стягається пеня.

3.3.2.19. Вебдодаток керування фінансами, який забезпечує виконання одноразових та періодичних платежів клієнтами. Початкова кількість коштів на рахунку кожного клієнта та його кредитні ліміти визначаються менеджером, якому доступна вся інформація про клієнтів. Клієнт може переглядати всю інформацію про власний рахунок (разом з історією транзакцій), вносити кошти, сплачувати за послуги, призначати періодичні платежі.

3.3.2.20. Вебдодаток підтримки аукціону товарів. Продавці виставляють на аукціон товари, визначають граничну вартість та термін проведення аукціону. Учасники аукціонів можуть визначати власну суму коштів, які вони готові сплатити за даний товар або викупати товари за граничну ціну. Перемагає заявка, в якій запропоновано найбільшу суму.

3.3.3. Виконати проектування та реалізувати базу даних MySQL у відповідності з індивідуальним завданням.

3.3.4. Розробити вебдодаток у відповідності з індивідуальним завданням.

3.3.5. Виконати тестування розробленого програмного забезпечення.

3.3.6. Оформити звіт.

3.3.7. Відповісти на контрольні питання.

3.4. Зміст звіту

- 3.4.1. Мета роботи.
- 3.4.2. Завдання до роботи.
- 3.4.3. Концептуальна модель бази даних.
- 3.4.4. Текст програми з основними коментарями.
- 3.4.5. Інтерфейс роботи з програмою в декількох режимах.
- 3.4.6. Результати тестування: вхідні дані та результати роботи програми.
- 3.4.7. Перелік проблем, які були виявлені і розв'язані або не розв'язані під час роботи над проектом, з описом відповідної ситуації. Якщо проблему було розв'язано, то необхідно додатково описати прийняті рішення. Якщо проблему не було розв'язано, ситуацію має бути ідентифіковано максимально детально з поясненням причин.
- 3.4.8. Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

3.5. Контрольні запитання

- 3.5.1 Які існують способи збереження даних в програмах, написаних мовою Python?
- 3.5.2 Для чого призначена бібліотека MySQLdb?
- 3.5.3 Які існують високорівневі функції бібліотеки MySQLdb?
- 3.5.4 Яким чином виконати запит до бази даних та яким чином переглянути результати?
- 3.5.5 Що таке шаблон проєктування MVC?
- 3.5.6 За допомогою яких команд виконується розроблення додатків у бібліотеці Django?
- 3.5.7 З яких файлів складається проект Django?
- 3.5.8 Яку структуру мають додатки Django?
- 3.5.9 Для чого необхідні та яким чином реалізуються моделі Django?
- 3.5.10 Яким чином визначаються представлення Django?
- 3.5.11 Для чого необхідні та яким чином визначаються і підключаються шаблони?
- 3.5.12 Яким чином можна використати статичні ресурси в Django?

ЛАБОРАТОРНА РОБОТА № 4

ОБРОБЛЕННЯ ПРИРОДНОЇ МОВИ

4.1. Мета роботи

4.1.1 Ознайомитись з основними інструментами оброблення природної мови, які входять у склад бібліотеки NLTK мови програмування Python

4.1.2 Навчитися розв'язувати актуальні практичні завдання угалузі оброблення природної мови за допомогою бібліотеки NLTK.

4.2. Короткі теоретичні відомості

Natural Language Toolkit (NLTK) – пакет бібліотек та програм для обробки природної мови.

Для початку роботи з бібліотекою NLTK необхідно встановити даний пакет з вебсайту nltk.org. Окрім програм дана бібліотека включає великий обсяг даних, які можуть використовуватись під час навчання моделей та характеризуються широкою розмаїтістю. Дані для пакету NLTK включають тексти, граматики, навчені моделі тощо. Для того щоб завантажити дані на комп’ютер, необхідно після встановлення пакету скористатися командами:

Python

```
>>> import nltk
>>> nltk.download()
```

Після виконання останньої команди на екрані з’явиться вікно NLTK Downloader (рис. 4.1), з якого можна вибрати дані, які необхідно завантажити.

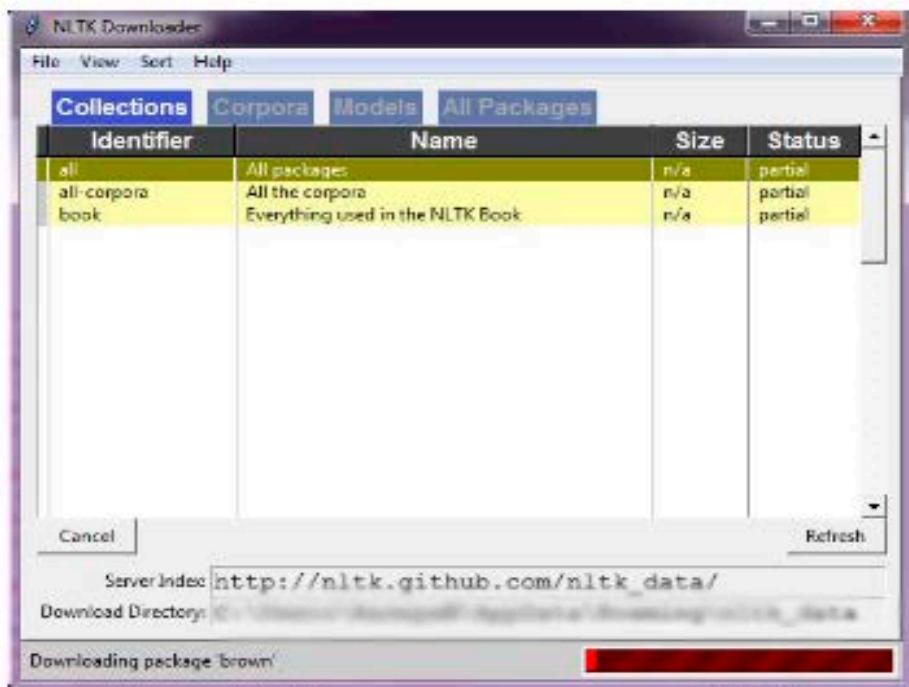


Рисунок 4.1 – Вікно завантаження даних пакету NLTK

Модуль пакету NLTK corpus містить тексти книг електронного архіву Проєкт Gutenberg ([gutenberg](http://gutenberg.org)), вебтексти (webtext), тексти з 500 джерел, розподілені за окремими жанрами (brown), більше 10 000 новин, розподілених за 90 темами, при цьому вони можуть належати одночасно декільком темам (reuters), також є тексти, розподілені в часі, словники (перелік слів, транскрипції для вимовляння, порівняння слів у різних мовах), семантично-орієнтований словник англійської мови (wordnet), що містить визначення, синоніми, ієархічні відносини між поняттями та більш специфічними визначеннями.

У таблиці 4.1 представлена модулі пакету NLTK та завдання процесу оброблення мови та функціональність, що їм відповідає.

Таблиця 4.1 – Модулі NLTK

Завдання оброблення мови	Модуль NLTK	Функціональність
Доступ до бібліотеки документів	corpus	Стандартизовані інтерфейси до документів та словників
Обробка рядків	tokenize, stem	Розбиття на лексеми
Визначення словосполучень	collocations	Визначення п-грам та відповідної статистики
Розмітка за частинами мови	tag	Створення правил та розподіл слів у реченні за частинами мови (послідовний метод, приховані марківські моделі (HMM), статистичний метод Торстена Брантса (TnT), метод Бриля (Brill))
Машинне навчання	classify, cluster, tbl	Дерева рішень, максимальна ентропія, наївні байесівські методи, k-середніх
Чанкінг	chunk	Формування деревоподібної структури на основі відповідності регулярним виразам над тегами
Контекстно-вільні граматики	grammar	Визначення продукцій
Парсинг	parse, ccg	Формування деревоподібної структури, що представляє внутрішню структуру тексту
Семантична інтерпретація	sem, inference	lambda-обчислення, логіка першого порядку, перевірка моделі
Оціночні метрики	metrics	Точність, коефіцієнти узгодженості
Імовірність та оцінка	probability	Розподіл частот, згладжений розподіл імовірностей
Лінгвістичні дослідження	toolbox	Читання, виведення та маніпулювання базами даних Toolbox

Основна функціональність corpus забезпечується за допомогою наступних методів:

- fileids() – файли корпусу текстів;
- fileids([categories]) – файли корпусу, що відповідають заданим категоріям;
- categories() – категорії корпусу;
- categories([fileids]) – категорії корпусу, що відповідають заданим файлам;
- raw() – необроблений вміст корпусу;
- raw(fileids=[f1,f2,f3]) – необроблений вміст заданих файлів;
- words() – слова зі всього корпусу;
- words(categories=[c1,c2]) – слова за заданими категоріями;
- sents() – речення зі всього корпусу;
- encoding(fileid) – кодування файла.

Приклад визначення кількості входжень заданого слова в деякий текст:

Python

```
>>> text3.generate()
>>> text3.count("smote")
5
```

Приклад визначення 50 слів, що найчастіше зустрічаються в деякому тексті:

Python

```
>>> fdist1 = nltk.FreqDist(text1)
>>> fdist1.most_common(50)
[(',', 18713), ('the', 13721), ('.', 6862), ('of', 6536), ('and', 6024),
('a', 4569), ('to', 4542), (';', 4072), ('in', 3916), ('that', 2982),
('"', 2684), ('-', 2552), ('his', 2459), ('it', 2209), ('I', 2124),
('s', 1739), ('is', 1695), ('he', 1661), ('with', 1659), ('was', 1632),
('as', 1620), ('"', 1478), ('all', 1462), ('for', 1414), ('this', 1280),
('!', 1269), ('at', 1231), ('by', 1137), ('but', 1113), ('not', 1103),
```

Python

```
(--, 1070), ('him', 1058), ('from', 1052), ('be', 1030), ('on', 1005),
('so', 918), ('whale', 906), ('one', 889), ('you', 841), ('had', 767),
('have', 760), ('there', 715), ('But', 705), ('or', 697), ('were', 680),
('now', 646), ('which', 640), ('?', 637), ('me', 627), ('like', 624)]
```

Можна також завантажити власну колекцію текстів або отримати доступ до текстів книг, HTML-документів, RSS-стрічок через інтернет, локальних файлів, документів форматів PDF (pypdf), MS Word (pywin32) тощо. Наприклад, доступ до HTML-сторінки можна отримати наступним способом:

Python

```
>>> from urllib import request
>>> url = "http://news.bbc.co.uk/2/hi/health/2284783.stm"
>>> html = request.urlopen(url).read().decode('utf8')
```

Для оброблення даних RSS-стрічки необхідно скористатися пакетом feedparser та наступними командами:

Python

```
>>> import feedparser
>>> llog =
feedparser.parse("http://languagelog.ldc.upenn.edu/nll/?feed=
atom")
```

Розглянемо приклад визначення частин мови:

Python

```
>>> text = word_tokenize("And now for something
completely different")
```

Python

```
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something',
'NN'), ('completely', 'RB'), ('different', 'JJ')]
```

Таким чином, *and* – єднальний сполучник, *now*, *completely* – прислівники, *for* – прийменник, *different* – прикметник, *something* – іменник.

Приклад створення власних регулярних виразів для визначення частин мови, до яких належать слова в тексті:

Python

```
>>> patterns = [
...   (r'.*ing$', 'VBG'),
...   (r'.*ed$', 'VBD'),
...   (r'.*es$', 'VBZ'),
...   (r'.*ould$', 'MD'),
...   (r'.*\$s$', 'NNS'),
...   (r'.*s$', 'NNS'),
...   (r'^-[0-9]+([0-9]+)?$', 'CD'),
...   (r'.*', 'NN')
... ]
>>> regexp_tagger = nltk.RegexpTagger(patterns)
>>> regexp_tagger.tag(brown_sents[3])
[('``', 'NN'), ('Only', 'NN'), ('a', 'NN'), ('relative', 'NN'),
('handful', 'NN'), ('of', 'NN'), ('such', 'NN'), ('reports',
'NNS'), ('was', 'NNS'), ('received', 'VBD'), ('''', 'NN'), (',',
'NN'), ('the', 'NN'), ('jury', 'NN'), ('said', 'NN'), (',',
'NN'), ('``', 'NN'), ('considering', 'VBG'), ('the', 'NN'),
('widespread', 'NN'), ...]
```

Приклад виконання чанкінга для речення «*The little yellow dog barked at the cat.*», який дозволяє побудувати в результаті деревоподібну структуру:

Python

```
>>> sentence = [("the", "DT"), ("little", "JJ"), ("yellow", "JJ"),
   ... ("dog", "NN"), ("barked", "VBD"), ("at", "IN"),
   ("the", "DT"), ("cat", "NN")]

>>> grammar = "NP: {<DT>?<JJ>*<NN>}"

>>> cp = nltk.RegexpParser(grammar)
>>> result = cp.parse(sentence)
>>> print(result)
(S
 (NP the/DT little/JJ yellow/JJ dog/NN)
 barked/VBD
 at/IN
 (NP the/DT cat/NN))
```

У результаті на екран можна вивести структуру, зображену на рис. 4.2.

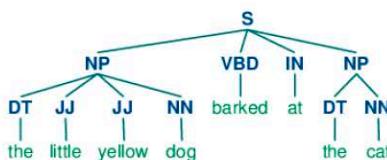


Рисунок 4.2 – Результат виконання чанкінгу

Розглянемо приклад виконання чанкінгу на основі власного регулярного виразу:

Python

```
grammar = r"""
NP: {<DT>?<JJ>*<NN>}
     {<NNP>+}
```

Python

```
cp = nltk.RegexpParser(grammar)
sentence = [("Rapunzel", "NNP"), ("let", "VBD"),
            ("down", "RP"), ("her", "PP$"), ("long", "JJ"),
            ("golden", "JJ"), ("hair", "NN")]

>>> print(cp.parse(sentence))
(S
  (NP Rapunzel/NNP)
  let/VBD
  down/RP
  (NP her/PP$ long/JJ golden/JJ hair/NN))
```

При підготовці матеріалів даного підрозділу використано документацію [30] (зокрема розділи [<https://www.nltk.org/book/ch00.html>], [<https://www.nltk.org/book/ch01.html>], [<https://www.nltk.org/book/ch02.html>], [<https://www.nltk.org/book/ch03.html>], [<https://www.nltk.org/book/ch05.html>], [https://www.nltk.org/book_1ed/ch07.html]).

4.3. Завдання на лабораторну роботу

4.3.1. Ознайомитись з теоретичними відомостями, необхідними для виконання роботи.

4.3.2. Розробити програмне забезпечення у відповідності з індивідуальним завданням, узгодженим з викладачем.

4.3.2.1. Реалізувати програму-бот, яка відповідає на запитання користувача за предметною областю з лабораторної роботи № 4 шляхом формування SQL-запитів та їх виконання.

4.3.2.2. Сформувати і синхронізувати дві часові шкали подій з історії заданої країни та історії її музики на основі аналізу текстів з вікіпедії та виділення основних дат і відповідних фактів зі статей.

4.3.2.3. Визначити події (опис, час та місце проведення) на основі аналізу твіттер-стрічki декількох заданих наперед облікових

записів. Нагадувати про їх наближення та наводити повний перелік подій у хронологічному порядку.

4.3.2.4. Виконати аналіз твіттер-стрічки за заданий період та виділити повідомлення про заданий користувачем музичний колектив, ідентифікувавши серед них позитивні та негативні відгуки. Підрахувати загальну кількість відгуків, кількість відгуків кожного виду та надати можливість переглянути їх.

4.3.2.5. Виконати переклад тексту з української мови на російську. Переклад виконувати на основі визначення тем (на основі набору текстів та слів і їх форм за даними темами), частин мови та статистичного підходу до машинного перекладу.

4.3.2.6. Реалізувати автоматизоване формування анотацій до статей з метою використання їх у RSS-стрічках. Анотація повинна узагальнювати основну інформацію, наведену в статті.

4.3.2.7. Виконати коригування тексту шляхом визначення правильності правопису. Перевіряючи правопис, обов'язково розподіляти слова за їх роллю у реченні. Для знайдених невідповідностей запропонувати варіанти на вибір, ранжуючи їх відповідним чином.

4.3.2.8. Реалізувати чат-бот, який цензурує повідомлення, які відсилаються між користувачами та вилучає з них персоналізовані образи та інші заборонені вирази.

4.3.2.9. Виконати аналіз тексту на наявність у ньому плагіату. Аналіз виконувати на основі банку текстів, визначених користувачем. У процесі аналізу визначати подібність як тексту загалом, так і окремих частин.

4.3.2.10. Розробити бот для діалогового спілкування з користувачем українською мовою. Відповіді повинні формуватися на основі співставлення з існуючими даними. Допускається ставити уточнюючі запитання за необхідності.

4.3.2.11. Реалізувати програмне забезпечення перевірки правильності пунктуації в заданому користувачем тексті. Окрім того перевіряти ситуації, коли одне й те саме слово зустрічається два рази підряд, визначаючи, які з них невірні, а які вірні.

4.3.2.12. Виконати аналіз блогу (статей і коментарів до них) та виділити інформацію про те, які теми майбутніх статей можуть бути популярними в аудиторії даного блогу.

4.3.2.13. Виконати аналіз RSS-стрічок заданого користувачем набору новинних агентств з метою виділення повідомлень про однакові новини та групування їх.

4.3.2.14. Виконати переклад тексту з української мови на англійську. Переклад виконувати на основі визначення тем (на основі набору текстів та слів і їх форм за даними темами), частин мови та статистичного підходу до машинного перекладу.

4.3.2.15. Розробити програмне забезпечення, яке дозволяє визначати мову, якою написаний текст або деякий фрагмент, та автоматично доповнювати текст, який вводить користувач, пропонуючи відповідний набір варіантів, ґрунтуючись зокрема на структурі речення.

4.3.2.16. Виконати аналіз тексту, виділяючи з нього всі займенники та співставляючи їх з відповідними іменниками або відповідними їм фактами. Вивести текст з поясненням іменниками (фактами) займенників.

4.3.2.17. Реалізувати діалогове спілкування з користувачем, під час якого програма відповідає на запитання користувача шляхом реферування. Реферування відбувається на основі текстів книг. Для кожного запитання визначається, якій частині (абзацу, реченню) якої книги відповідає дане запитання.

4.3.2.18. Реалізувати створення звітів щодо роботи підприємства за рік шляхом анотування документів, які оформлювалися на підприємстві протягом даного періоду часу.

4.3.2.19. Реалізувати створення програмних відповідей українською мовою на основі виконання SQL-запитів до бази даних, розробленої в лабораторній роботі № 3.

4.3.2.20. Виконати перетворення заданого користувачем алгоритму (у вигляді кроків, сформульованих однією з природніх мов) на текст програми однією з мов програмування.

4.3.3. Виконати тестування розробленого програмного забезпечення.

4.3.4. Оформити звіт.

4.3.5. Відповісти на контрольні запитання.

4.4. Зміст звіту

- 4.4.1. Мета роботи.
- 4.4.2. Завдання до роботи.
- 4.4.3. Текст програми з основними коментарями.
- 4.4.4. Інтерфейс роботи з програмою в декількох режимах.
- 4.4.5. Результати тестування: вхідні дані та результати роботи програми.
- 4.4.6. Перелік проблем, які були виявлені і розв'язані або не розв'язані під час роботи над проектом, з описом відповідної ситуації. Якщо проблему було розв'язано, то необхідно додатково описати прийняті рішення. Якщо проблему не було розв'язано, ситуацію має бути ідентифіковано максимально детально з поясненням причин.
- 4.4.7. Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

4.5. Контрольні запитання

- 4.5.1. Яким чином виконується оброблення природної мови в бібліотеці NLTK?
- 4.5.2. Які основні модулі входять до складу бібліотеки NLTK?
- 4.5.3. Яким чином можна використати готові тексти для роботи програми на Python?
- 4.5.4. Яким чином розбити текст на лексеми?
- 4.5.5. Яким чином побудувати граматику в NLTK?
- 4.5.6. Що таке біграми та яким чином їх визначити у тексті?
- 4.5.7. Що таке чанкінг та яким чином він виконується?
- 4.5.8. Яким чином та за допомогою яких засобів виконується семантична інтерпретація?
- 4.5.9. Що таке контекстно-вільна граматика? Наведіть приклади.
- 4.5.10. Яким чином визначити частини мови слів тексту?

ЛАБОРАТОРНА РОБОТА № 5

МАШИННЕ НАВЧАННЯ

5.1. Мета роботи

5.1.1 Ознайомитись з основними пакетами, які використовуються для машинного навчання в програмах, написаних мовою Python.

5.1.2. Навчитися розробляти сучасні інтелектуальні системи з використанням методів машинного навчання.

5.2. Короткі теоретичні відомості

5.2.1 Бібліотека Pandas

Пакет pandas забезпечує аналіз даних з реального світу для програм, написаних мовою Python.

Структурами даних, які використовуються в даному пакеті, є:

– Series ([data, index, dtype, name, copy, ...]) – одновимірний проіндексований масив, який може містити дані будь-яких типів (s = Series(data, index=index), де *data* може бути даними типу dict (Series працює подібно словникам), ndarray (Series працює подібно масивам NumPy) або скалярного типу, а *index* – перелік міток для даних) [<https://pandas.pydata.org/docs/reference/api/pandas.Series.html>];

– DataFrame ([data, index, columns, dtype, copy]) – двовимірний проіндексований масив з колонками потенційно будь-якого типу. DataFrame в якості даних може використовувати dict з одновимірних ndarray, двомірні ndarray, структуровані ndarray або типу запис, Series, DataFrame [<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html#pandas.DataFrame>].

Для того щоб визначити категоріальні значення, необхідно використовувати під час створення об'єкту Series атрибут *dtype="category"*.

Приклад створення структури даних типу DataFrame:

Python

```
import pandas as pd
```

Python

```
data = {'start' : pd.Series([2., 7., 9.], index=['e', 'x', 'a']),
        'end' : pd.Series([4., 5., 9., 10.], index=['e', 'x', 'a', 'm'])}
framed = pd.DataFrame(data)
```

У DataFrame стовпці можна вибирати, додавати, вилучати аналогічно тому, як це виконується зі словниками. Індексація описана в таблиці 5.1. Окрім того можна отримувати доступ до значень елементів структур даних типу Series, вказуючи індекс в якості атрибута (sa.c). Більш складні умови вибору можна реалізовувати за допомогою методу map. Для визначення умов можуть також використовуватися метод where та окремо для DataFrame – query (умови визначаються з використанням синтаксису Python). Для виконання зрізів використовується метод select.

Таблиця 5.1 – Операції індексації у DataFrame

Операція	Синтаксис	Результат
Вибрати стовпчик	df[col]	Series
Вибрати рядок за міткою	df.loc[label]	Series
Вибрати рядок за цілочисельною позицією	df.iloc[loc]	Series
Виконати зріз рядків	df[5:10]	DataFrame
Вибрати рядки за вектором логічних значень	df[bool_vec]	DataFrame

Для роботи з даними структурами можуть використовуватися методи head ([n]) і tail ([n]), index (рядки). DataFrame має методи add, sub, mul, div, пов’язані функції radd, rsub для виконання бінарних операцій і методи бінарного порівняння eq, ne, lt, gt, le, ge. Для того щоб застосувати визначену користувачем функцію до двох структур даних типу DataFrame, необхідно використати метод combine, для виконання дій над однією структурою призначений метод apply (func[, axis]), у випадку необхідності роботи з невекторизованими даними використовуються методи applymap (func) і map (arg[, na_action]). Для підбиття статистичних підсумків над даними може використовуватися

метод `describe ([percentile_width, ...])`.

Методи `idxmin ([axis, out, skipna])` і `idxmax ([axis, out, skipna])` повертають індекс з мінімальним/максимальним значенням, а функції `cut (x, bins[, right, labels, retbins, ...])` і `qcut (x, q[, labels, retbins, precision])` розділяють значення на задану кількість інтервалів.

Для зміни структур даних використовується метод `reindex ([index])`, а для зміни індексів – метод `rename ([index])`. Сортування виконується за допомогою методів `sort_index ([ascending])` за індексами та `order ([return_indexer, ascending])` за значеннями.

Об'єкт Series надає методи `nsmallest ([n, take_last])` і `nlargest ([n, take_last])`, що повертають n найменших і відповідно найбільших значень.

Для того щоб ідентифікувати та вилучити дубльовані рядки в DataFrame призначені методи `duplicated ([take_last])` та `drop_duplicates ([take_last, inplace])`.

Для того щоб змінити DataFrame, додаючи новий індекс, можна використати метод `set_index (keys[, drop, append, ...])`, а метод `reset_index ([level, drop, name, inplace])` – для перетворення індексів у стовпці DataFrame.

Для побудови ієрархічних індексів може використовуватися об'єкт MultiIndex, який можна побудувати з переліку масивів (`MultiIndex.from_arrays`), масиву кортежів (`MultiIndex.from_tuples`) або множини ітераторів, що перетинаються (`MultiIndex.from_product`).

Метод `swaplevel (i, j[, axis])` дозволяє змінювати порядок двох рівнів, узагальнює його метод `reorder_levels (order)`.

Набір статистичних функцій включає `pct_change`, що обчислює зміну у відсотках протягом заданого періоду, `cov` – коваріацію, `pearson`, `kendall`, `spearman` – коефіцієнти кореляції тощо.

Для роботи з пропущеними значеннями в даних, які можуть бути представлені як `NaN`, `None`, `inf`, `-inf`, призначені функції: `isnull()`, `notnull()`, для заповнення пропущених значень – `fillna ([value, method, axis, ...])`, для вилучення індексів з пропущеними значеннями – `dropna ([axis, inplace])`, для інтерполяції пропущених значень – `interpolate ([method, axis, limit, ...])`.

Для розподілу об'єктів pandas на групи використовується метод `groupby`, після чого отримати перелік сформованих груп можна за допомогою атрибути `groups`, отримати доступ до окремих груп за

допомогою метода `get_group (name[, obj])` або використанням ітераторів, а обчислити розмір груп за допомогою метода `size`. Метод `aggregate (func, *args, **kwargs)` призначений для виконання обчислень на згрупованих даних та використовує в якості атрибуту функцію, перелік або словник функцій, а метод `transform (func, *args, **kwargs)` – для трансформації згрупованих об'єктів з використанням заданої функції. Для фільтрування даних відповідно до визначеної умови призначений метод `filter ([items, like, regex, axis])`.

Функція `concat(objs[, axis, join, join_axes, ...])` призначена для конкатенації об'єктів вздовж однієї з осей, де `objs` – перелік або словник об'єктів Series, DataFrame, Panel.

Для об'єднання двох DataFrame об'єктів призначена функція `merge(left, right[, how, on, left_on, ...])`, яка за суттю близька до аналогічної функції в SQL. Якщо ключі співпадають, то результат обчислюється у вигляді декартового добутку.

Для об'єднання двох об'єктів DataFrames з різними індексами використовується метод `join (other[, on, how, lsuffix, ...])`.

Для зміни представлення об'єктів призначений метод `pivot(index, columns, values)`, який формує зведену таблицю з трьох стовпців DataFrame, метод `pivot_table(*args, **kwargs)`, який створює зведену таблицю у вигляді об'єкта DataFrame з використанням за необхідності функції агрегації, та метод `melt(frame[, id_vars, value_vars, var_name, ...])`, який перетворює об'єкт DataFrame з широкого формату у довгий.

Pandas надає достатньо ефективний інструментарій роботи з часовими рядами, особливо у галузі аналізу фінансових даних. Функція `date_range([start, end, periods, freq, tz, ...])` дозволяє створити часовий інтервал, який може бути використаний, наприклад, в якості індексу для об'єкту Series. Для перетворення об'єктів Series у часові дані можна використати функцію `to_datetime(arg[, errors, dayfirst, utc, ...])`. Для створення часових міток призначена функція `datetime`, а для створення з них індексу – `DatetimeIndex`. Об'єкти `Timedelta` використовуються для визначення різниці в часі. Для їх створення з існуючих об'єктів призначена функція `to_timedelta(arg[, unit, box, coerce])`. Кожний з перелічених об'єктів має властивості, за допомогою яких можна отримати відповідні показники часу.

API для введення/виведення містить наступний набір функцій:

– `read_csv(filepath_or_buffer[, sep, dialect, ...])` – читає з текстового файлу дані, перетворюючи табличні дані (необхідно задати розділювач *sep*) на об'єкти DataFrame. Використовує зокрема наступні параметри: *dtype* – словник з назвами стовпців та типами даних, *header* – номери рядків з назвами стовпців, *name* – перелік назв стовпців, *na_values*, *true_values*, *false_values* – перелік рядкових значень, які будуть прочитані як NaN, True, False відповідно, *index_col* – номери стовпців, назви стовпців або перелік назв/номерів стовпців, які будуть використані в якості індексів DataFrame, *dialect* – визначає формат файла, *usecols* – підмножина стовпців з файла, які будуть використані, *comment* – задає формат коментарів [https://pandas.pydata.org/docs/dev/reference/api/pandas.read_csv.html#pandas.read_csv];

– `to_csv(path[, index, sep, na_rep, ...])` – метод об'єктів Series і DataFrame, який дозволяє зберігати їх дані в csv-файл;

– `read_html(io[, match, flavor, header, ...])` і `to_html([buf, columns, col_space, ...])` – читає (виводить) HTML рядок/файл/URL та парсить HTML таблиці в перелік об'єктів DataFrames;

– `read_excel(io[, sheetname])` і `to_excel(*args, **kwargs)` – зчитує (виводить) файли Excel 2003 (.xls) and Excel 2007 (.xlsx). За допомогою атрибута *sheet_names* можна задавати назви аркушів;

– `read_json([path_or_buf, orient, typ, dtype, ...])` і `to_json([path_or_buf, orient, ...])` – читає та виводить дані в файли (рядки) формату JSON;

– `read_msgpack(path_or_buf, iterator=False, **kwargs)` і `to_msgpack(path_or_buf=None, **kwargs)` – зчитування та виведення даних в форматі msgpack;

– `read_hdf(path_or_buf, key, **kwargs)` і `to_hdf(path_or_buf, key, **kwargs)` – зчитування та виведення даних в форматі HDF5;

– `read_gbq(query[, project_id, index_col, ...])` і `to_gbq(dataframe, destination_table[, ...])` – зчитування та передачі даних у вебсервіс аналітики BigQuery;

– `read_stata(filepath_or_buffer[, ...])` і `to_stata(fname[, convert_dates, ...])` – зчитування та виведення даних у файл з розширенням .dta;

– `read_clipboard(**kwargs)` і `to_clipboard([excel, sep])` – використовують буфер обміну, зчитуючи з нього або передаючи в

нього дані;

- `read_pickle(path)` і `to_pickle (path)` – зчитує (виводить) структури даних з (на) диску в форматі pickle;
- `read_sql_table(table_name, con[, schema, ...])` – зчитування таблиць бази даних у об'єкт DataFrame;
- `read_sql_query(sql, con[, index_col, ...])` – зчитування результату виконання SQL-запиту в об'єкт DataFrame;
- `read_sql(sql, con[, index_col, ...])` – зчитування результату SQL-запиту або таблиці бази даних у об'єкт DataFrame;
- `to_sql(name, con[, flavor, ...])` – виводить записи, що зберігаються в об'єкті DataFrame, у базу даних SQL.

При підготовці матеріалів даного пункту використано документацію [26] (зокрема розділи [<https://pandas.pydata.org/docs/dev/reference/io.html>]), [35] (зокрема розділи [https://pandas.pydata.org/docs/user_guide/dsintro.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/groupby.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/reshaping.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/advanced.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/computation.html], [https://pandas.pydata.org/pandas-docs/stable/user_guide/basics.html]).

5.2.2 Бібліотека scikit-learn

Бібліотека scikit-learn – бібліотека машинного навчання, яка підтримує алгоритми класифікації, регресії, кластеризації.

Алгоритми машинного навчання реалізовані за допомогою класів та функцій. Класи імплементують метод `fit` для навчання (наприклад, кластерів на навчальній вибірці) та методи `get_params()` і `set_params()` для визначення і встановлення параметрів оцінювача, а функції, наприклад, отримуючи навчальну вибірку, повертають масив цілочисельних міток, що відповідають різним кластерам. Набір методів кожного класу залежить від алгоритму машинного навчання, який він реалізує. Наприклад, для прогнозування може

використовуватись метод `predict(X)`, для оцінювання деяких параметрів – метод `score()`, а метод `transform()` – для перетворення набору даних (наприклад, під час вибору властивостей).

Базовими класами функцій оцінювання є базовий клас `BaseEstimator`, клас для всіх класифікаторів `ClassifierMixin`, клас для всіх оцінювачів кластерів `ClusterMixin`, клас для всіх функцій оцінювання регресійних моделей `RegressorMixin`, клас для всіх перетворювачів `TransformerMixin`. Метод `score(X, y[, sample_weight])` класів `ClassifierMixin` та `RegressorMixin` повертає середню похибку на даний тестовій вибірці. Метод `fit_predict(X[, y])` класу `ClusterMixin` виконує кластеризацію на наборі даних X та повертає мітки класів.

Модуль `sklearn.preprocessing` включає методи масштабування, центрування, нормалізації та перетворення у двійкову форму, зокрема:

- `Binarizer([threshold, copy])` – перетворює у двійкову форму за порогом;
- `LabelBinarizer([neg_label, ...])` і `label_binarize(y, classes[, ...])` – клас і функція, які перетворюють мітки у двійкову форму («один-проти-всіх»);
- `LabelEncoder` – кодує мітки за значеннями між 0 та $n_classes-1$;
- `MinMaxScaler([feature_range, copy])` – стандартизує властивості, зважуючи кожну з них у заданому діапазоні;
- `Normalizer([norm, copy])` – нормалізує вібірки індивідуально до одиничної норми;
- `OneHotEncoder([n_values, ...])` – кодує категоріальні цілочисельні властивості, використовуючи схему «один-з-К»;
- `StandardScaler([copy, ...])` – стандартизує властивості, вилучаючи середнє значення та зважуючи до одиничної дисперсії;
- `binarize(X[, threshold, copy])` – логічне встановлення порогу для матриць типів, подібних до `array` або `scipy.sparse`;
- `label_binarize(y, classes[, ...])` – перетворює мітки у двійкову форму («один-проти-всіх»);
- `normalize(X[, norm, axis, copy])` – нормалізує множину даних вздовж будь-якої вісі;
- `scale(X[, axis, with_mean, ...])` – центрує множину даних на середньому значенні та зважує компоненти взовж будь-якої вісі для отримання одиничної дисперсії.

Модуль `sklearn.feature_extraction` призначений для видобування

властивостей з необроблених даних і на даний момент включає методи для видобування властивостей з тексту та зображень.

Функція `DictVectorizer([dtype, ...])` перетворює перелік відображень властивість-значення у вектори, а функція `FeatureHasher(...)` імплементує гешування властивостей .

Підмодуль `sklearn.feature_extraction.image` містить засоби, які видобувають властивості з зображень.

Підмодуль `sklearn.feature_extraction.text` містить засоби для побудови векторів властивостей з текстових документів: `CountVectorizer(...)`, який перетворює колекцію текстових документів у матрицю підрахунку лексем, `TfidfTransformer(...)`, який перетворює матрицю підрахунку в нормалізоване представлення TF або TF-IDF, `TfidfVectorizer(...)`, який перетворює колекцію необроблених документів у матрицю властивостей TF-IDF.

Модуль `sklearn.feature_selection` імплементує алгоритми вибору властивостей:

- `GenericUnivariateSelect(...)` – одномірний селектор зі стратегією, що конфігурується;
- `SelectPercentile(...)` – вибір властивостей відповідно до процентилі найвищої оцінки;
- `SelectKBest([score_func, k])` – вибір властивостей відповідно до k найвищих оцінок;
- `SelectFpr([score_func, alpha])` – фільтр: вибір p -значень нижче α , ґрунтуючись на FPR-тесті;
- `SelectFdr([score_func, alpha])` – фільтр: вибір p -значень для очікуваної долі хибних відхилень;
- `SelectFwe([score_func, alpha])` – фільтр: вибір p -значень, що відповідають коефіцієнту помилок під час множинного порівняння;
- `RFE(estimator[, ...])` – ранжування властивостей з рекурсивним вилученням властивостей;
- `RFECV(estimator[, step, ...])` – ранжування властивостей з рекурсивним вилученням властивостей та перевіркою найкращої кількості властивостей;
- `VarianceThreshold([threshold])` – відбір властивостей з вилученням всіх властивостей з низькою дисперсією;
- `chi2(X, y)` – обчислює статистику хі-квадрат для кожної комбінації клас/властивість;

– `f_classif(X, y)` – обчислює F-значення дисперсійного аналізу для наданої вибірки;

– `f_regression(X, y[, center])` – перевірка на присутність одномірної лінійної регресійної залежності.

Модуль `sklearn.discriminant_analysis` імплементує дискримінантний аналіз за допомогою функцій `LinearDiscriminantAnalysis(...)`, що представляє лінійний дискримінантний аналіз, та `QuadraticDiscriminantAnalysis(*)`, що представляє квадратичний дискримінантний аналіз, а модуль `sklearn.linear_model` – узагальнені лінійні моделі за допомогою класів для байесівської ARD регресії `ARDRegression([n_iter, tol, ...])`, байесівської регуляризації Тихонова `BayesianRidge([n_iter, tol, ...])`, лінійної регресії з комбінуванням L1 і L2 `ElasticNet([alpha, l1_ratio, ...])`, моделі Elastic Net з ітеративним підбором вздовж шляху регуляризації `ElasticNetCV([l1_ratio, eps, ...])`, моделі регресії найменших кутів `Lars([fit_intercept, verbose, ...])`, моделі регресії найменших кутів з перехресною перевіркою `LarsCV([fit_intercept, ...])`, L1-регуляризації `Lasso([alpha, fit_intercept, ...])`, лінійної моделі L1-регуляризації з ітеративним підбором вздовж шляху регуляризації `LassoCV([eps, n_alphas, ...])`, моделі L1-регуляризації, що відповідає методу найменших кутів, `LassoLars([alpha, ...])`, моделі L1-регуляризації з перехресною перевіркою, що відповідає методу найменших кутів, `LassoLarsCV([fit_intercept, ...])`, моделі L1-регуляризації, що відповідає методу найменших кутів, з використанням ВІС або АІС для вибору моделі `LassoLarsIC([criterion, ...])`, звичайної лінійної регресії на основі методу найменших квадратів `LinearRegression(...)`, класифікатора на основі логістичної регресії `LogisticRegression([penalty, ...])`, моделі багатозадачної L1-регуляризації, навченої з використанням змішаної L1/L2-норми в якості регуляризатора `MultiTaskLasso([alpha, ...])`, багатозадачної моделі `ElasticNet`, навченої за допомогою змішаної L1/L2-норми в якості регуляризатора, `MultiTaskElasticNet([alpha, ...])`, багатозадачної L1/L2-регуляризації з вбудованою перехресною перевіркою `MultiTaskLassoCV([eps, ...])`, багатозадачної моделі `ElasticNet` з вбудованою перехресною перевіркою `MultiTaskElasticNetCV([...])`, моделі ортогонального узгодженого переслідування `OrthogonalMatchingPursuit([...])`, моделі ортогонального узгодженого

переслідування з перехресною перевіркою OrthogonalMatchingPursuitCV(...]), перцептрона Perceptron([penalty, alpha, ...]), випадкової L1-регуляризації RandomizedLasso([alpha, ...]), випадкової логістичної регресії RandomizedLogisticRegression(...]), стабільного методу оцінки параметрів моделі на основі випадкових вибірок RANSACRegressor(...]), лінійної моделі на основі методу найменших квадратів з L2-регуляризацією Ridge([alpha, fit_intercept, ...]), класифікації з використанням регуляризації Тихонова RidgeClassifier([alpha, ...]), класифікатора на основі регуляризації Тихонова з вбудованою перехресною перевіркою RidgeClassifierCV([alphas, ...]), регуляризації Тихонова з вбудованою перехресною перевіркою RidgeCV([alphas, ...]), лінійного класифікатора з навчанням за допомогою методу SGDClassifier([loss, penalty, ...]).

Модуль `sklearn.neighbors` імплементує алгоритм k-найближчих сусідів за допомогою наступних класів: `NearestNeighbors([n_neighbors, ...])`, навчання без учителя для імплементації пошуку сусідів, `KNeighborsClassifier(...)`, класифікатор, що імплементує голосування k-найближчих сусідів, `RadiusNeighborsClassifier(...)`, класифікатор, що імплементує голосування між сусідами в заданому радіусі, `KNeighborsRegressor([n_neighbors, ...])`, регресія на основі методу k-найближчих сусідів, `RadiusNeighborsRegressor([radius, ...])`, регресія, що ґрунтуються на сусідах у заданому радіусі, `kneighbors_graph(X, n_neighbors[, ...])`, який обчислює зважений граф k-сусідів для точок з X, `radius_neighbors_graph(X, radius)`, який обчислює зважений граф сусідів для точок з X.

Модуль `sklearn.naive_bayes` імплементує алгоритм найвної байесівської класифікації, включаючи наступні класи: `GaussianNB`, для поліноміальних моделей `MultinomialNB([alpha, ...])`, для багатомірних моделей Бернуллі `BernoulliNB([alpha, binarize, ...])`.

Модуль `sklearn.multiclass` імплементує наступні алгоритми багатокласової класифікації за допомогою відповідних класів та функцій підбору стратегії і прогнозування:

- `OneVsRestClassifier(estimator[, ...])`, `fit_ovr(estimator, X, y[, n_jobs])`, `predict_ovr(estimators, ...)` – багатокласова стратегія «один-проти-інших»;

- `OneVsOneClassifier(estimator[, ...])`, `fit_ovo(estimator, X, y[, n_jobs])`, `predict_ovo(estimators, ...)` – багатокласова стратегія «один-проти-одного»;

`n_jobs]), predict_ovo(estimators, classes, X)` – багатокласова стратегія «один-проти-одного»;

– `OutputCodeClassifier(estimator[, ...]), fit_ecoc(estimator, X, y[, ...]), predict_ecoc(estimators, classes, ...)` – багатокласова стратегія кодування виходів з корекцією помилок.

Модуль `sklearn.ensemble` включає методи класифікації та регресії на основі використання ансамблів: `AdaBoostClassifier([...]), AdaBoostRegressor([base_estimator, ...]), BaggingClassifier([base_estimator, ...]), BaggingRegressor([base_estimator, ...]), ExtraTreesClassifier([...]), ExtraTreesRegressor([n_estimators, ...]), GradientBoostingClassifier([loss, ...]), GradientBoostingRegressor([loss, ...]), RandomForestClassifier([...]), RandomTreesEmbedding([...]), RandomForestRegressor([...])`.

Модуль `sklearn.cluster` дозволяє виконувати кластеризацію непройндексованих даних.

Функції модуля:

– `estimate_bandwidth(X[, quantile, ...])` – оцінює ширину полоси для використання з алгоритмом здигу середнього;

– `k_means(X, n_clusters[, init, ...])` – алгоритм кластеризації К-середніх;

– `ward_tree(X[, connectivity, ...])` – ієархічна кластеризація Уорда;

– `affinity_propagation(S[, ...])` – виконує кластеризацію даних методом Affinity Propagation («розповсюдження близькості»);

– `dbSCAN(X[, eps, min_samples, ...])` – кластеризація методом DBSCAN з векторного масиву або матриці відстаней;

– `mean_shift(X[, bandwidth, seeds, ...])` – кластеризація методом здигу середнього;

– `spectral_clustering(affinity[, ...])` – застосовує кластеризацію до проекції на нормалізований лапласіан.

Класи модуля: для агломеративної кластеризації `AgglomerativeClustering([...])` і `FeatureAgglomeration([n_clusters, ...])`, а також відповідні розглянутим функціям `KMeans([n_clusters, init, n_init, ...])` і `MiniBatchKMeans([n_clusters, init, ...]), Ward([n_clusters, memory, ...]), AffinityPropagation([damping, ...]), DBSCAN([eps, min_samples, metric, ...]), MeanShift([bandwidth, seeds, ...]), SpectralClustering([n_clusters, ...])`.

Для використання методів бікластеризації, які дозволяють розв'язувати задачі з вимогами збереження об'єктивно-ознакового опису подібності кластерів, призначений модуль `sklearn.cluster.bicluster`.

Модуль `sklearn.tree` включає моделі для класифікації (клас `DecisionTreeClassifier` та його альтернатива `ExtraTreeClassifier([criterion, ...])`) та регресії (клас `DecisionTreeRegressor` та його альтернатива `ExtraTreeRegressor([criterion, ...])`) на основі дерев рішень та функцію `export_graphviz(decision_tree[, ...])` для експорту дерев рішень у формат DOT.

Модуль `sklearn.semi_supervised` імплементує алгоритми часткового навчання, які використовують невелику кількість промаркованих даних та велику кількість не промаркованих даних для завдань класифікації: класифікатор `LabelPropagation([kernel, ...])` і модель для навчання `LabelSpreading([kernel, ...])`.

Модуль `sklearn.metrics` включає функції оцінки, метрики продуктивності, попарні метрики (`metrics.pairwise`) та обчислення відстані.

Класифікаційні метрики включають оцінювання класифікаційної похибки `accuracy_score(y_true, y_pred[, ...])`, обчислення площин під кривою з використанням трапеційального правила `auc(x, y[, reorder])`, обчислення середньої точності оцінок прогнозування `average_precision_score(y_true, y_score)`, побудову текстового звіту з основними класифікаційними метриками `classification_report(y_true, y_pred)`, обчислення матриці неточностей для оцінювання похибки класифікації `confusion_matrix(y_true, y_pred[, ...])`, обчислення оцінки F1 (F-міри) `f1_score(y_true, y_pred[, labels, ...])`, обчислення оцінки F-beta `fbeta_score(y_true, y_pred, beta[, ...])`, обчислення оцінки коефіцієнта подібності Жакара `metrics.jaccard_similarity_score(y_true, y_pred)`, обчислення точності та підтримки для кожного класу `precision_recall_fscore_support(...)`, обчислення точності `precision_score(y_true, y_pred[, ...])` тощо.

Регресійні метрики включають функції отримання оцінки поясненої дисперсії `explained_variance_score(y_true, y_pred)`, середньої абсолютної похибки втрат `mean_absolute_error(y_true, y_pred)`, середньо-квадратичної похибки втрат `mean_squared_error(y_true, y_pred[, ...])`, коефіцієнта детермінації `r2_score(y_true, y_pred[, ...])`,

`sample_weight]).`

Метрики результатів кластерного аналізу включають скориговану повну інформацію між двома кластеризаціями `adjusted_mutual_info_score(...)`, метрику завершеності маркування кластерів `completeness_score(labels_true, ...)`, обчислення однорідності та завершеності показника V-міри одночасно `homogeneity_completeness_v_measure(...)`, однорідку метрику маркування кластерів `homogeneity_score(labels_true, ...)`, повну інформацію між двома кластеризаціями `mutual_info_score(labels_true, ...)`, нормалізовану повну інформацію між двома кластеризаціями `normalized_mutual_info_score(...)`, середній силуетний коефіцієнт всіх вибірок `silhouette_score(X, labels[, ...])`, силуетний коефіцієнт для кожної вибірки `silhouette_samples(X, labels[, metric])`.

Попарні метрики включають: `additive_chi2_kernel(X[, Y])`, що обчислює адитивне хі-квадрат ядро між спостереженнями в X та Y , `chi2_kernel(X[, Y, gamma])`, що обчислює експоненційне хі-квадрат ядро між X та Y , `distance_metrics()`, набір метрик для попарних відстаней, `euclidean_distances(X[, Y, ...])`, що розглядаючи рядки ($Y=X$) як вектори, обчислює матрицю відстаней між кожною парою векторів, `kernel_metrics()`, що повертає набір метрик для попарних ядер, `linear_kernel(X[, Y])`, що обчислює лінійне ядро між X та Y , `manhattan_distances(X[, Y, ...])`, що обчислює L1-відстань між векторами X і Y , `pairwise_kernels(X[, Y, ...])`, що обчислює ядро між масивами X і необов'язковим ядром Y , `polynomial_kernel(X[, Y, ...])`, що обчислює поліноміальне ядро між X і Y , `rbf_kernel(X[, Y, gamma])`, що обчислює гаусівське ядро між X і Y , `pairwise_distances(X[, Y, metric, ...])`, що обчислює матрицю відстаней з векторного масиву X та за необхідності Y , `pairwise_distances_argmin(X, Y[, ...])`, що обчислює мінімальні відстані між однією точкою та множиною точок.

Модуль `sklearn.svm` включає алгоритми методу опорних векторів, зокрема оцінювачі: C-SVM класифікація `SVC([C, kernel, degree, gamma, coef0, ...])`, Nu-SVM класифікація `NuSVC([nu, kernel, degree, gamma, ...])`, лінійна SVM класифікація `LinearSVC([penalty, loss, dual, tol, C, ...])`, епсілон-SVM регресія `SVR([kernel, degree, gamma, coef0, tol, ...])`, Nu-SVM регресія `NuSVR([nu, C, kernel, degree, gamma, ...])` тощо та низькорівневі методи для тренування моделі, використовуючи `libsvm` `libsvm.fit`, передбачення ймовірності

`libsvm.predict_proba`, передбачення цільових значень X за заданою моделлю `libsvm.predict`, передбачення межі `libsvm.decision_function`.

Модуль `sklearn.gaussian_process` імплементує скалярне прогнозування на основі гаусового процесу.

Модуль `sklearn.covariance` включає методи та алгоритми для надійного оцінювання коваріації властивостей на основі множини точок: клас оцінювача коваріації методом найбільшої правдоподібності `EmpiricalCovariance(...)` і відповідна функція `empirical_covariance(X[, ...])`, клас оцінювача Ledoit-Wolf `LedoitWolf([store_precision, ...])` і відповідна функція `ledoit_wolf(X[, assume_centered, ...])`, об'єкт для виявлення викидів у множині даних, розподілених за гаусівським законом, `EllipticEnvelope(...)`, клас оцінювача коваріації на основі алгоритму Oracle Approximating Shrinkage `OAS([store_precision, ...])` і відповідна функція `oas(X[, assume_centered])`, оцінювач коваріації зі стисканням `ShrunkCovariance(...)`, оцінювач коваріації з L1-штрафом `graph_lasso(emp_cov, alpha[, ...])` тощо.

Модуль `sklearn.model_selection` включає засоби крос-валідації та оцінювання продуктивності: ітератор k -кратної крос-валідації `KFold(n[, n_folds, ...])`, ітератор крос-валідації з виключенням по одному об'єкту `LeaveOneOut()`, ітератор крос-валідації з вилученням p -вибірок `LeavePOut(p)`, ітератор багатошарової k -кратної крос-валідації `StratifiedKFold([n_splits, ...])`, ітератор крос-валідації випадковою перестановкою `ShuffleSplit(n[, n_iter, ...])`, ітератор багатошарової крос-валідації випадковою перестановкою `StratifiedShuffleSplit(...)`, розбиття масивів або матриць на випадкові навчальні та тестові підмножини `train_test_split(*arrays, ...)`, `GridSearchCV(estimator, param_grid)`, який виконує пошук методом повного перебору для заданих значень параметрів оцінювача, `ParameterGrid(param_grid)`, сітка параметрів з дискретним числом значень для кожного, `ParameterSampler(...[, random_state])`, генератор параметрів, вибраних за даним розподілом, `RandomizedSearchCV(estimator, ...)`, випадковий пошук на гіперпараметрах.

Модуль `sklearn.datasets` включає засоби для завантаження даних та генерації вибірок. Функції завантаження даних дозволяють завантажувати імена файлів і дані з 20 новинних груп

(`fetch_20newsgroups([data_home, ...])`), цифрові множини даних для класифікації (`load_digits([n_class])`), текстові файли з категоріями в якості імен підкаталогів (`load_files(container_path[, ...])`), дані про класифікацію ірисів (`datasets.load_iris()`), масиви `numpy` окремих вибіркових зображень (`load_sample_image(image_name)`), вибірки зображень для маніпуляцій з зображеннями (`datasets.load_sample_images()`). Функції генерації вибірок дозволяють згенерувати випадкову проблему класифікації на n класів (`make_classification([n_samples, ...])`), дані для бінарної класифікації (`make_hastie_10_2([n_samples, ...])`), випадкову регресійну проблему (`make_regression([n_samples, ...])`), випадкову симетричну позитивно-визначену матрицю (`make_spd_matrix(n_dim[, random_state])`) тощо.

Модуль `sklearn.decomposition` включає алгоритми розкладання матриць, включаючи зокрема метод головних компонент (`PCA([n_components, copy, whiten])`), ядерний метод головних компонент (`KernelPCA([n_components, ...])`), факторний аналіз (`FactorAnalysis([n_components, ...])`), факторизацію невід'ємних матриць за допомогою проекції градієнта (`ProjectedGradientNMF([...])`), швидкий алгоритм для аналізу незалежних компонент (клас `FastICA([n_components, ...])`) і функція `fastica(X[, n_components, ...])`, пониження розмірності з використанням присіченого SVD (`TruncatedSVD([n_components, ...])`), розріджений метод головних компонент (`SparsePCA([n_components, ...])`), розрідженої кодування (клас `SparseCoder(dictionary[, ...])`) і функція `sparse_encode(X, dictionary[, ...])`, розріджений метод головних компонент над міні-групами даних (`MiniBatchSparsePCA([...])`), алгоритм навчання словника (`DictionaryLearning([...])`) та його варіант на міні-групах даних (`MiniBatchDictionaryLearning([...])`), розв'язання проблеми факторизації матриці навчання словника (`dict_learning(X, n_components, ...)` і `dict_learning_online(X[, ...])`).

Розглянемо приклад практичного застосування алгоритмів машинного навчання, представлених у бібліотеці `scikit-learn`, для розпізнавання зображень цифр, написаних вручну, розміром 8x8:

Python

```

import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics
digits = datasets.load_digits()
images_and_labels = list(zip(digits.images, digits.target))
for index, (image, label) in enumerate(images_and_labels[:8]):
    plt.subplot(2, 8, index + 1)
    plt.axis('off')
    plt.imshow(image, cmap=plt.cm.gray_r,
               interpolation='nearest')
    plt.title('Train: %i' % label)

n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
classifier = svm.SVC(gamma=0.001)
classifier.fit(data[:n_samples / 2], digits.target[:n_samples / 2])

expected = digits.target[n_samples / 2:]
predicted = classifier.predict(data[n_samples / 2:])
print("Classification report for classifier %s:\n%s\n"
      % (classifier,
         metrics.classification_report(expected, predicted)))
print("Confusion matrix:\n%s" % metrics.confusion_matrix(expected, predicted))

images_and_predictions = list(zip(digits.images[n_samples / 2:], predicted))
for index, (image, prediction) in enumerate(images_and_predictions[:8]):
    plt.subplot(2, 8, index + 9)
    plt.axis('off')
    plt.imshow(image, cmap=plt.cm.gray_r,
               interpolation='nearest')
    plt.title('Predict: %i' % prediction)
    plt.show()

```

На рис. 5.1 представлено результат виконання даної програми. Матриця неточностей за результатами машинного навчання:

```
[[87 0 0 0 1 0 0 0 0]
 [ 0 88 1 0 0 0 0 0 1 1]
 [ 0 0 85 1 0 0 0 0 0 0]
 [ 0 0 0 79 0 3 0 4 5 0]
 [ 0 0 0 0 88 0 0 0 0 4]
 [ 0 0 0 0 0 88 1 0 0 2]
 [ 0 1 0 0 0 0 90 0 0 0]
 [ 0 0 0 0 0 1 0 88 0 0]
 [ 0 0 0 0 0 0 0 0 88 0]
 [ 0 0 0 1 0 1 0 0 0 90]]
```

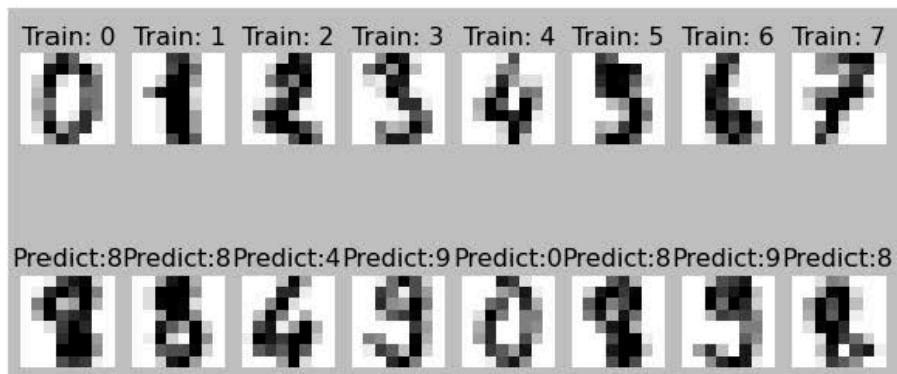


Рисунок 5.1 – Розпізнавання цифр, написаних вручну

При підготовці матеріалів даного пункту використано документацію [16], [25].

5.3. Завдання на лабораторну роботу

5.3.1. Ознайомитись з теоретичними відомостями, необхідними для виконання роботи.

5.3.2. Розробити програмне забезпечення у відповідності з індивідуальним завданням, узгодженим з викладачем.

5.3.2.1. Для покупця інтернет-магазину створити систему, яка дозволить порадити йому набір товарів у залежності від тих товарів, які він придбав на даний момент, та в залежності від обраного в даний момент товару. Використати один з критеріїв оцінювання подібності товарів. Дані про товари завантажити з одного з існуючих інтернет-магазинів, заповнивши ними власний проект.

5.3.2.2. Розробити пошукову систему, яка дозволяє виконувати пошук в деяких галузях знань. Визначити, чи належить запит до однієї з даних галузей (можливі параметри: тема запиту, мова, локалізація пошуку тощо). Виконувати пошук тільки в тому випадку, якщо запит відповідає умовам попередньої перевірки. Обрати алгоритм оцінювання релевантності та у відповідності з даним показником виводити результати на екран. Індекс сформувати за допомогою власного скрипта з деякої області сайтів.

5.3.2.3. Розробити систему, яка виконує аналіз інформаційного вмісту заданого набору вебсайтів (новинних агентств). У результаті аналізу повинно виконуватись визначення контенту, який є рекламию деякої продукції, та інформування користувачів про виявлену ситуацію у деякому матеріалі. Матеріалом може бути як ціла стаття (новина), так і її частина.

5.3.2.4. Розробити програмне забезпечення, яке дозволяє автоматизовано визначити жанр кінострічки на основі її синопсису, виступаючи в ролі порадника з одного боку, а з іншого – в ролі контролера жанрів, що було обрано самостійно користувачами сервісу. Використати в якості навчання дані про кінострічки з бази даних про кінематограф imdb.com, видобуті автоматизовано.

5.3.2.5. Розробити програмне забезпечення прогнозування цін на ринку житла в одному з міст. Використати відкрите сховище даних, в якому представлені дані про ринок житла в даному місті. Програмне забезпечення повинно автоматизовано видобувати дані з такого сховища через інтернет. Використати декілька моделей для прогнозування та порівняти їх ефективність.

5.3.2.6. Розробити програмне забезпечення, яке формує стрічку новин за інтересами користувача, використовуючи RSS-стрічки новинних сайтів та уникуючи дублювання інформації. Для кожної новини надається назва, за якою можна переглянути її короткий опис, а за необхідності перейти на відповідний вебсайт (або агрегуючи

декілька у випадку дублювання новин) та прочитати новину повністю (такий випадок розглядається як зацікавленість користувача).

5.3.2.7. Розробити програмне забезпечення, яке дозволяє фільтрувати повідомлення, які приходять на e-mail користувача. Визначити окремо повідомлення, які належать до спаму та фішингових атак через e-mail. Виділити ознаки, що характеризують фішингові повідомлення. Навчання щодо виділення повідомлень у спам проводити як на основі готових прикладів таких повідомлень, так і на основі вибору користувача.

5.3.2.8. Розробити програмне забезпечення, яке пропонує геш-теги для коротких повідомлень з соціальної мережі твіттер. Пропозиції формувати двома способами: на основі визначення подібності з існуючими геш-тегами з врахуванням динамічності природи використання геш-тегів та на основі співставлення повідомлення з найпопулярнішими на даний момент геш-тегами.

5.3.2.9. Розробити програмне забезпечення, яке дозволяє розшифрувати текст, представлений за допомогою САРТСНА, шляхом сегментації зображення та класифікації отриманих символів. Проаналізувати використання декількох методів машинного навчання для розв'язання даної задачі. Пристосувати програмне забезпечення для ефективної роботи з декількома алгоритмами формування САРТСНА (можлива наявність шуму або поворот символів).

5.3.2.10. Розробити програмне забезпечення, яке дозволяє визначити товар за його зображенням. Сформувати базу зображень товарів. Дозволити користувачу завантажити фотографію товару в обраному самостійно форматі. Визначити, якому товару дане зображення належить, або вивести повідомлення, якщо достатньо точно визначити даний товар не вдалось.

5.3.2.11. Розробити програмне забезпечення, яке дозволяє визначити сторінки в мережі твіттер, які можуть потенційно зацікавити користувача і можуть бути віднесені до рекомендацій для підписки. Для створення рекомендацій використовувати дані про підписки користувачів (граф мережі) та визначати, на скільки цікавим може бути інформація, що міститься на стрічці відповідного користувача, даному користувачу.

5.3.2.12. Розробити програмне забезпечення, яке дозволяє визначати оптимальний склад команди для гри у фентезі-футбол на

основі Ліги чемпіонів УЄФА. Фентезі-футбол – це гра, у якій учасники формують віртуальну команду футболістів, чиї прототипи беруть участь у реальних змаганнях і залежно від актуальної статистики своїх виступів набирають залікові бали. Використовувати статистику футболіста для прогнозування балів, які він набере в наступних іграх.

5.3.2.13. Розробити програмне забезпечення, яке дозволяє прогнозувати касові збори кінострічки на основі описових даних про неї (режисер, актори, жанр тощо). Обрати як мінімум дві моделі для прогнозування самостійно, застосувати їх та порівняти отримані результати. Описові дані для навчання використаної моделі сформувати на основі бази даних про кінематограф kinopoisk.ru.

5.3.2.14. Розробити програмне забезпечення, яке дозволяє формувати список відтворення музичних композицій. Список наявних композицій сформувати на основі платформи soundcloud. На початку розподілити наявні композиції на декілька списків та запропонувати користувачу обрати з них. Далі використовувати дані про прослуховування користувачем композицій для формування списку подальшого відтворення на основі його інтересів.

5.3.2.15. Розробити програмне забезпечення, яке виконує прогнозування результатів футбольних матчів одного з європейських чемпіонатів. Прогнозування виконувати на основі статистики виступів команд, гравців, їх поточного положення (або останніх ігр) та інших умов проведення гри. Властивості, необхідні для прогнозування, обирати автоматизовано. Статистику про виступи команд на поточний момент використати з одного з відповідних вебсайтів (наприклад, whoscored.com).

5.3.2.16. Розробити програмне забезпечення, яке виконує стискання найпопулярніших фотографій з фотохостингу (розглянути в даній якості flickr.com). Стискання виконати шляхом зменшення кількості кольорів у зображенні на основі виділення основних кольорів та приведення інших кольорів до основних. Кількість кольорів визначити автоматизовано в заданому проміжку. Спосіб визначення запропонувати самостійно.

5.3.2.17. Розробити програмне забезпечення, яке формує перелік туристичних подорожей, які пропонуються туристичними операторами онлайн. Базу туристичних операторів наповнювати за

рахунок пошуку в мережі інтернет та їх подальшої класифікації. Розподіляти туристичні подорожі на декілька типів (круїз, дайвінг, екотуризм тощо, залежні від регіону).

5.3.2.18. Розробити програмне забезпечення, яке дозволяє визначити акорди, наявні в представленому аудіо-файлі. Розглянути файли формату WAV. Використати для навчання аудіо-файли та відповідні представлення нотних записів у вигляді файлів формату MusicXML. Розглянути найпопулярніші акорди. Проаналізувати використання декількох методів машинного навчання для розв'язання даної задачі.

5.3.2.19. Розробити програмне забезпечення, яке дозволяє виділити в даному зображенні набір основних кольорів. Кількість кольорів може бути обрана попередньо з заданого діапазону. Алгоритм кластеризації обрати самостійно. Для обраного зображення визначити також набір зображень з подібними кольорами. Банк зображень повинен формуватися автоматизовано.

5.3.2.20. Розробити систему розпізнавання відсканованого тексту. Обрати фільтр для поліпшення якості зображення. Приклади символів визначити в тому числі самостійно. Використовувати словники. У випадку неоднозначностей дозволити користувачу поліпшити варіант перекладу, але у виконаному перекладі обрати варіант, який система оцінює як найбільш відповідний. Передбачити роботу з двома природніми мовами. Система повинна автоматично обирати мову тексту.

5.3.3. Виконати тестування розробленого програмного забезпечення.

5.3.4. Оформити звіт.

5.3.5. Відповісти на контрольні запитання.

5.4. Зміст звіту

5.4.1. Мета роботи.

5.4.2. Завдання до роботи.

5.4.3. Опис алгоритма розв'язання задачі.

5.4.4. Текст програми з основними коментарями.

5.4.5. Інтерфейс роботи з програмою в декількох режимах.

5.4.6. Результати тестування: вхідні дані та результати роботи програми.

5.4.7. Перелік проблем, які були виявлені і розв'язані або не розв'язані під час роботи над проектом, з описом відповідної ситуації. Якщо проблему було розв'язано, то необхідно додатково описати прийняті рішення. Якщо проблему не було розв'язано, ситуацію має бути ідентифіковано максимально детально з поясненням причин.

5.4.8. Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

5.5. Контрольні запитання

5.5.1. Для чого призначена бібліотека pandas?

5.5.2. Які структури даних та для чого використовуються в пакеті pandas?

5.5.3. Які функції бібліотеки pandas можуть бути використані для введення/виведення даних?

5.5.4. Яким чином можна модифікувати дані, що зберігаються в структурах даних пакету pandas?

5.5.5. Яким чином виконується індексація даних у пакеті pandas?

5.5.6. Які функції для роботи з декількома структурами даних підтримуються бібліотекою pandas?

5.5.7. Яким чином виконується оброблення часових рядів у бібліотеці pandas?

5.5.8. Яку галузь знань охоплює машинне навчання та які завдання воно дозволяє розв'язати?

5.5.9. Для виконання яких завдань призначена бібліотека scikit-learn?

5.5.10. Які засоби вибору властивостей надає бібліотека scikit-learn?

5.5.11. Які засоби попереднього оброблення даних надає бібліотека scikit-learn?

5.5.12. Які засоби кластеризації надає бібліотека scikit-learn?

5.5.13. Які засоби класифікації надає бібліотека scikit-learn?

5.5.14. Які методи навчання без учителя представлені в бібліотеці scikit-learn?

5.5.15. Яким чином інтегруються засоби бібліотек pandas та scikit-learn?

5.5.16. Які засоби перетворення даних використовуються в бібліотеці scikit-learn?

5.5.17. Які засоби пониження розмірності представлені в бібліотеці scikit-learn?

5.5.18. Які регресійні моделі можуть бути застосовані за допомогою засобів scikit-learn?

5.5.19. Які метрики надаються бібліотекою scikit-learn?

5.5.20. За допомогою яких засобів бібліотеки scikit-learn можна оцінити коваріацію?

5.5.21. Яким чином можна візуалізувати результати машинного навчання?

5.5.22. Для чого необхідні методи крос-валідації та які інструменти для цього використовуються в бібліотеці scikit-learn?

5.5.23. Яким чином можна згенерувати дані для машинного навчання?

5.5.24. Дані яких типів та з яких джерел можна обробляти за допомогою засобів бібліотеки scikit-learn?

ЛІТЕРАТУРА

1. Vincent, W. S. Django for Beginners [Текст] : Build Websites with Python and Django / W. S. Vincent. – WelcomeToCode, 2020. – 317 p.
2. Олещенко, Л. М. Машинне навчання [Електронний ресурс] : комп'ютерний практикум з дисципліни "Машинне навчання": навчальний посібник / Л. М. Олещенко. – Київ : КПІ ім. Ігоря Сікорського, 2022. – 92 с.
3. Копей, В. Б. Мова програмування Python для інженерів і науковців [Текст] : навчальний посібник / Б. В. Копей. – Івано-Франківськ : ІФНТУНГ, 2019. – 275 с.
4. Lee, W.-M. Python Machine Learning [Текст] / W.-M. Lee. – Wiley, 2019. – 320 р.
5. Raschka, S. Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2 [Текст] / Sebastian Raschka, Vahid Mirjalili. – Packt Publishing, 2019. – 772 р.
6. Програмування числових методів мовою Python [Текст] : підручник / А. В. Анісімов, А. Ю. Дорошенко, С. Д. Погорілій, Я. Ю. Дорогий ; за ред. А. В. Анісімова. – К. : Видавничо-поліграфічний центр "Київський університет", 2014. – 640 с.
7. Sarkar, D. Text Analytics with Python [Текст] : A Practitioner's Guide to Natural Language Processing / D. Sarkar. – Apress, 2019. – 698 р.
8. Moore, A. D. Python GUI Programming with Tkinter [Текст] : Design and build functional and user-friendly GUI applications / A. D. Moore. – Packt Publishing, 2021. – 664 р.
9. Lutz, M. Programming Python [Текст] : Powerful Object-Oriented Programming / M. Lutz. – Fourth edition. – O'Reilly Media, 2011. – 1626 р.
10. tkinter – Python interface to Tcl/Tk [Електронний ресурс]. – Режим доступу : <https://docs.python.org/3/library/tkinter.html>.
11. NumPy documentation [Електронний ресурс]. – Режим доступу : <https://numpy.org/doc/stable/>.
12. SciPy documentation [Електронний ресурс]. – Режим доступу : <https://docs.scipy.org/doc/scipy/>.
13. Tutorials – Matplotlib [Електронний ресурс]. – Режим доступу : <https://matplotlib.org/stable/tutorials/index>.

14. Django documentation [Електронний ресурс]. – Режим доступу : <https://docs.djangoproject.com/en/4.2/>.
15. pandas documentation [Електронний ресурс]. – Режим доступу : <https://pandas.pydata.org/docs/>.
16. User guide [Електронний ресурс] : contents – scikit-learn 1.2.2 documentation. – Режим доступу : https://scikit-learn.org/stable/user_guide.html.
17. NLTK [Електронний ресурс] : Natural Language Toolkit. – Режим доступу: <https://www.nltk.org/>.
18. MySQLdb User's Guide – MySQLdb 1.2.4b4 documentation [Електронний ресурс]. – Режим доступу : https://mysqlclient.readthedocs.io/user_guide.html.
19. Python 3.11.5 documentation [Електронний ресурс]. – Режим доступу : <https://docs.python.org/3/>.
20. Matplotlib Release 3.5.3 [Електронний ресурс] / John Hunter, Darren Dale, Eric Firing, Michael Droettboom. – Режим доступу : <http://matplotlib.org/Matplotlib.pdf>.
21. Graphical User Interfaces with Tk – Python 3.10.6 documentation [Електронний ресурс]. – Режим доступу : <https://docs.python.org/3/library/tk.html>.
22. TkDocs Tutorial [Електронний ресурс]. – Режим доступу : <https://tkdocs.com/tutorial/index.html>.
23. Python Documentation contents – Python 3.6.3 Documentation [Електронний ресурс]. – Режим доступу : <https://documentation.help/Python-3.6.3/contents.html>.
24. Tkinter 8.5 reference [Електронний ресурс] : a GUI for Python. – Режим доступу : <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html>.
25. API reference – scikit-learn 1.1.2 documentation [Електронний ресурс]. – Режим доступу : <http://scikit-learn.org/stable/modules/classes/documentation.html>.
26. API reference – pandas 1.6.0 [Електронний ресурс]. – Режим доступу : <https://pandas.pydata.org/docs/dev/reference/index.html>.
27. MySQL 5.6 Reference Manual. Including MySQL Cluster NDB 7.3-7.4 Reference Guide [Електронний ресурс]. – Режим доступу : <http://downloads.mysql.com/docs/refman-5.6-en.a4.pdf>.

28. MySQLdb Package – MySQLdb 1.2.4b4 documentation [Електронний ресурс]. – Режим доступу : <https://mysqlclient.readthedocs.io/MySQLdb.html>.
29. Effective Django – Effective Django [Електронний ресурс]. – Режим доступу : <http://effectivedjango.com/>.
30. NLTK Book [Електронний ресурс]. – Режим доступу : <https://www.nltk.org/book/>.
31. Shipman, J. W. Tkinter 8.5 reference [Електронний ресурс] : aGUI for Python / J. W. Shipman. – Режим доступу : <https://tkdocs.com/shipman/tkinter.pdf>.
32. Tcl8.5.19/Tk8.5.19 Documentation [Електронний ресурс]. – Режим доступу : <https://www.tcl.tk/man/tcl8.5/contents.html>.
33. NumPy quickstart – NumPy v1.24.dev0 Manual [Електронний ресурс]. – Режим доступу : <https://numpy.org/devdocs/user/quickstart.html>.
34. Users guide – Matplotlib 3.5.3 documentation [Електронний ресурс]. – Режим доступу : <https://matplotlib.org/stable/users/index.html>.
35. User guide – pandas 1.4.4 documentation [Електронний ресурс]. – Режим доступу : https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html.