Тема 2
Змінні та команди

Ви вже ознайомлені з Python – мовою програмування, розробленою для того, щоб зробити кодування простим і цікавим. Відкрийте Thonny, середовище, яке будете використовувати для написання та запуску коду на Python.

Структуру вашої програми можна змінювати залежно від призначення та її складності.

Змінні

Щоб робити обчислення і показувати результати, потрібно використовувати змінні. Змінна – контейнер для збереження та обробки даних у програмуванні. Для зберігання різних типів даних можна використовувати дві основні категорії змінних: числа й текст. Текстові змінні також називають рядками.

```
level=5
Score=1200
TotalAmount=120.50
```

Числа (числові змінні)

```
name='Богдан'
Message="Ви будете грати знов? Так/Ні"
email='john@binary-academy.com'
```

Текст (рядки)

Типи даних

Типи даних виникають внаслідок поділу даних на певні категорії. Тип даних задає як множину можливих значень, так і операції, які можна виконувати над цими значеннями. Python містить деякі вбудовані типи даних.

Структура програми на Python

Імпорт модулів

```
import pygame
pygame.init()
```

Глобальні змінні

```
width, height = 800, 600
```

Визначення функцій

```
def sum(a,b):
    c=a+b
    return c
```

Визначення класів

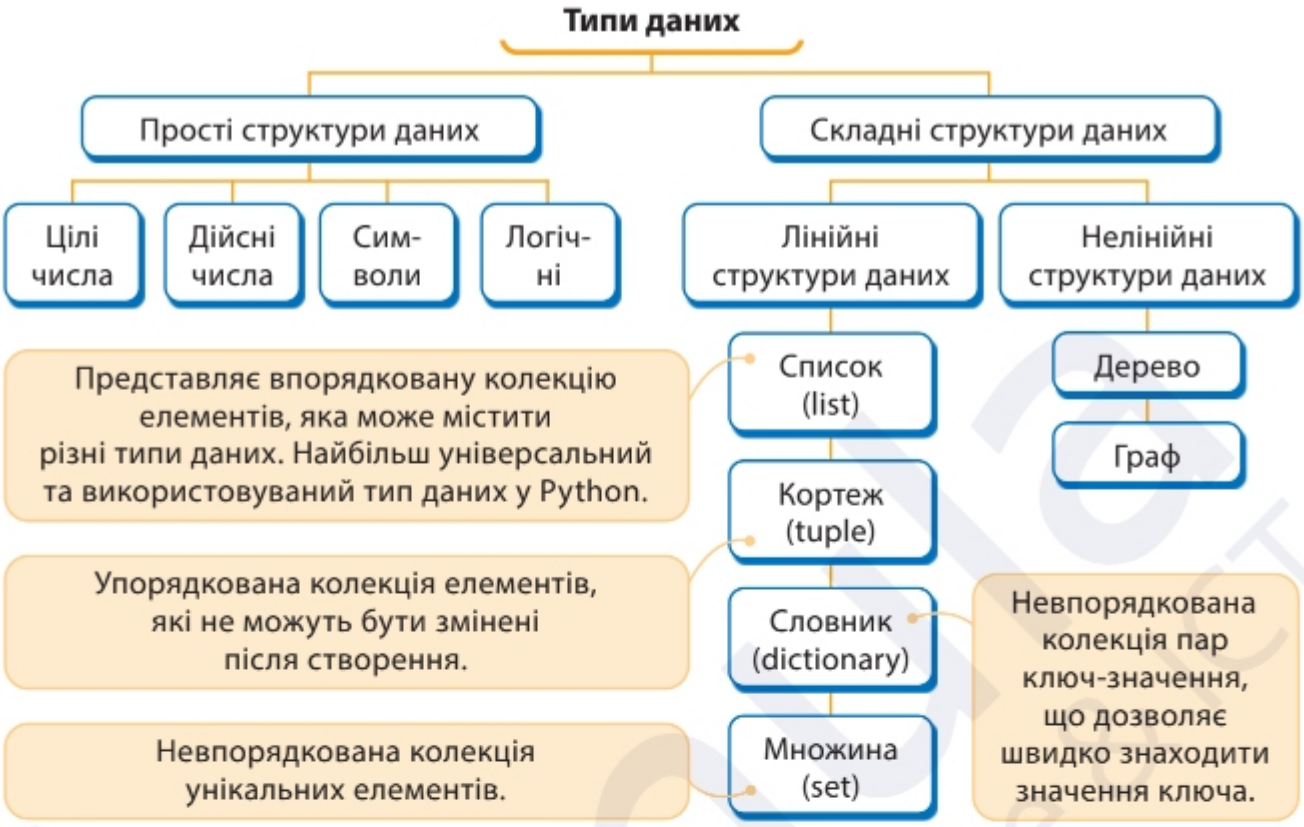
```
class MyClass:
    def __init__(self):
```

Основний код програми

```
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
```

ПІДКАЗКА

Пам'ятайте, що текст для рядкових змінних потрібно брати в лапки – одинарні ' чи подвійні ". Вони повинні бути однакові на початку та в кінці.



ТИП ДАНИХ	НАЗВА В PYTHON	ПРИКЛАД
Цілі числа	int	12, -999, 0, 900000
Дійсні числа	float	4.5, 0.0003, -90.5, 3.0
Символи текстові, числові та спеціальні	str	"Привіт!", "Іванка", "$$$"
Логічні (булеві) дані	bool	False, True

Список використовують для зберігання кількох елементів зі схожими характеристиками. Зазвичай ці елементи мають однаковий тип.

Списки

У Python можна зберігати дані в змінних, але їх можна розміщувати й у списках. Створити список так само просто, як ввести різні значення, розділені комами, між квадратними дужками. Кожному елементу списку присвоюється значення за допомогою **індексу**.

```
MyList=[15, 20, 35.78, 'Четвер']
```

Команда `MyList=[2]` відобразить третій елемент списку, який має значення **35.78**. Аналогічно можна змінити третій елемент, просто призначте його безпосередньо до списку, набравши `MyList[2]="Субота"`.

ПІДКАЗКА

Деякі імена не можна використовувати, оскільки вони є спеціальними словами, які вже використовуються в мові програмування. Такі слова зарезервовані системою: **print**, **return**, **while**, **True**, **False**, **else**, **and**, **not**, **import**, **None**, **global**, **break**.

Видалення та додавання елементів списку

Якщо точно відомо, який елемент списку треба видалити, скористайтеся командою **del**. Протестуйте наведений код.

```
JumpList=[6.27, 5.20, 7, 5.5, 6.2]
print(JumpList)
del JumpList[2]
print("Після видалення елемента з індексом 2")
print(JumpList)
```

```
[6.27, 5.2, 7, 5.5, 6.2]
Після видалення елемента з індексом 2
[6.27, 5.2, 5.5, 6.2]
```

Якщо треба додати елемент до списку, використайте команду **append** із назвою списку. Наприклад, щоб уставити значення 6.6 як елемент у кінець наявного списку, введіть наведений нижче код.

```
JumpList=[6.27, 5.20, 7, 5.5, 6.2]
print(JumpList)
JumpList.append(6.6)
print("Після додавання елемента")
print(JumpList)
```

```
[6.27, 5.2, 7, 5.5, 6.2]
Після додавання елемента
[6.27, 5.2, 7, 5.5, 6.2, 6.6]
```

Для отримання частини списку використайте двокрапку (:) між квадратними дужками списку. Наприклад, команда **Mylist[2:5]** повертає підсписок із третього елемента до п'ятого, не включаючи сам елемент з індексом 5.

```
JumpList=[6.27, 5.20, 7, 5.5, 6.2, 7]
print(JumpList[2:3])
print(JumpList[2:5])
```

```
[7]
[7, 5.5, 6.2]
```

Операції зі списками

len() показує кількість елементів у списку.

```
grades=[89,88,35,95]
print(grades)
gradesItems=len(grades)
print(gradesItems)
```

```
[89, 88, 35, 95]
4
```

sum() додає всі елементи списку та повертає їхню суму.

```
grades=[89,88,35,95]
myGrades=sum(grades)
print("Сума оцінок:",myGrades)
```

```
Сума оцінок: 307
```

max() показує найбільший елемент зі списку.

```
grades=[89,88,35,95]
maxGrade=max(grades)
print("Максимальна оцінка:",maxGrade)
```

```
Максимальна оцінка: 95
```

min() показує найменший елемент зі списку.

```
grades=[89,88,35,95]
minGrade=min(grades)
print("Мінімальна оцінка:",minGrade)
```

```
Мінімальна оцінка: 35
```

Коли вводите код, будьте уважні. Завжди використовуйте правильний синтаксис мови.

*Функція **sum()** працює тільки із числовими значеннями.*

ПІДКАЗКА

Перший елемент списку в Python має індекс 0, другий – 1 і далі. Тому, щоб отримати третій елемент списку **MyList**, треба надрукувати **MyList[2]**.

append() можна використати для створення списку на основі вхідних даних від користувача. Значення додаються в кінець списку.

Створіть пустий список.

```
subjects=[ ]
print("введіть назву предмета",0)
subjects.append((input()))
print("введіть назву предмета",1)
subjects.append((input()))
print("введіть назву предмета",2)
subjects.append((input()))
print(subjects)
```

```
введіть назву предмета 0
фізика
введіть назву предмета 1
математика
введіть назву предмета 2
історія
['фізика', 'математика', 'історія']
```

На екран виводиться створений список.

remove() видаляє вказаний елемент зі списку.

Видаляє елементу зі списку за його значенням.

```
grades=[89,35,98,18,95]
grades.remove(18)
print(grades)
grades.remove(grades[2])
print(grades)
```

```
[89, 35, 98, 95]
[89, 35, 98]
```

Видаляє елемент зі списку по індексу.

*Функції **min()** та **max()** не працюють зі змішаними списками, які містять і числа, і символи.*

count(x) підраховує, скільки разів указаний елемент відображається в списку.

```
grades=[89,35,98,18,95]
grades.append(100)
grades.remove(18)
print(grades)
y=grades.count(100)
print(y)
```

```
[89, 35, 98, 95, 100]
1
```

sort() сортує елементи списку за зростанням.

```
grades=[89,35,98,18,95]
grades.append(100)
grades.remove(18)
print(grades)
grades.sort()
print(grades)
```

```
[89, 35, 98, 95, 100]
[35, 89, 95, 98, 100]
```

reverse() змінює послідовність елементів списку на зворотний.

```
grades=[89,35,98,18,95]
grades.append(100)
grades.remove(18)
print(grades)
grades.reverse()
print(grades)
```

```
[89, 35, 98, 95, 100]
[100, 95, 98, 35, 89]
```

clear() видаляє всі елементи списку.

```
grades=[89,35,98,18,95]
grades.append(100)
grades.remove(18)
print(grades)
grades.clear()
print(grades)
```

```
[89, 35, 98, 95, 100]
[]
```

Вкладені списки

Список може містити елементи різного типу, навіть інший список. Вам уже відомо, що цикл може бути всередині іншого циклу. Так само й список може бути всередині іншого списку.

```
list=[3,["a","b","c"],7.5,-2,"апельсин"]
print(list)
print(list[0])
print(list[1])
print(list[2])
print(list[3])
print(list[4])
```

```
[3, ['a', 'b', 'c'], 7.5, -2, 'апельсин']
3
['a', 'b', 'c']
7.5
-2
апельсин
```

Другий елемент списку є інший список.

Внутрішній список подібний до інших елементів списку. Він опрацьовується так, як решта елементів.

Довжина списку не змінилася.

```
list=[3,["a","b","c"],7.5,-2,"апельсин"]
l=len(list)
print("довжина списку:",l)
```

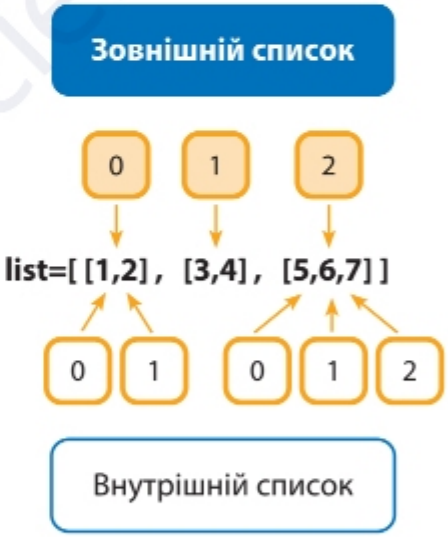
```
довжина списку: 5
```

Щоб вивести на екран елемент вкладеного списку, потрібні два числа. Перше – це індексний номер елемента в зовнішньому списку, а другий – індекс внутрішнього списку.

Виведемо на екран вкладені списки та їхні елементи.

```
list=[[1,2],['c','d'],[15,62,79]]
#print the 1st item
print(list[0])
print(list[0][0])
print(list[0][1])
#print the 2nd item
print(list[1])
print(list[1][0])
print(list[1][1])
#print 3rd item
print(list[2])
print(list[2][0])
print(list[2][1])
print(list[2][2])
```

```
[1, 2]
1
2
['c', 'd']
c
d
[15, 62, 79]
15
62
79
```



Практичне завдання

Завантажте файл-заготовку зі сторінки інтернет-підтримки та виконайте запропоновані завдання.