

СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

**122 «Комп'ютерні науки»
КН-19**

2020 / 2021 навчальний рік

PYTHON #8

1. ООП в Python.
2. Створення класу.
3. Простір імен.
4. Дерево класів, успадкування.
5. Вбудовані аргументи.

КЛАСИ

Класи Python реалізують усі основні концепції ООП:

- ❑ **Інкапсуляція:** об'єднання даних та методів
- ❑ **Спадкування:** створення нового класу на базі існуючого
- ❑ **Поліморфізм:** використання об'єктів з однаковим інтерфейсом без інформації про тип і внутрішню структуру об'єкта

КЛАСИ

КЛАС Python

→ **об'єкт** Python

ЕКЗЕМПЛЯР (інстанс, instance) КЛАСУ

→ **об'єкт** Python

!!! Різні об'єкти !!!

C++	Python	
Поле	АТРИБУТ	Змінна (вказівник)
Метод (процедура, функція)		Функція, визначена в класі

СТВОРЕННЯ КЛАСУ

```
class <name> (superclass, ... ):  
    <тіло класу>
```

Кожна інструкція **class** створює новий об'єкт типу **клас**.

Кожний раз, коли викликається клас, він створює новий об'єкт – **екземпляр** класу.

Екземпляри автоматично зв'язуються з класом із яких вони були створені.

Класи зв'язані із своїми суперкласами, які вказані в круглих дужках в інструкції **class**. Послідовність суперкласів у списку визначає порядок розташування в дереві пошуку імен.

СТВОРЕННЯ КЛАСУ

```
class <name> (superclass, ... ):  
    <тіло класу>
```

class – **інструкція** створення **об'єкту**, посилення на який зберігається в просторі імен як **<ім'я класу>**.

<тіло_класу> сукупність інструкцій. Може містити присвоювання змінним **data=value** (**data** стає **атрибутом** -> **полем** класу) та визначення функцій **def ...(self, ...):** (стає **атрибутом** -> **методом** класу).

Перший аргументом методу завжди є екземпляр класу, для якого викликається цей метод. За угодою, цей аргумент називається **"self"**.

Спеціальний метод **__init__()** – конструктор – викликається автоматично при створенні екземпляра класу.

СТВОРЕННЯ КЛАСУ

```
class MyClass :
```

```
    def __init__(self, val)  
        self.value = val
```

```
    def prntvl (self)  
        print('value =', self.value)
```

СТВОРЮЄ об'єкт
типу **КЛАС**

Конструктор

Функція класу

```
Myobj = MyClass(3.14)
```

```
Myobj.prntvl()
```

Звернення до аргументу (методу)

```
Myobj.value = "НОНОНО"
```

```
Myobj.prntvl()
```

Звернення до аргументу (поле)

СТВОРЮЄ об'єкт типу
ЕКЗЕМПЛЯР КЛАСУ

→ 3.14

→ НОНОНО

ПРОСТІР ІМЕН

Виклик об'єкта класу як функції створює новий об'єкт екземпляру.

Кожен об'єкт екземпляру успадковує атрибути класу і набуває **свій власний простір імен**. Вони спочатку **порожні**, але успадковують атрибути класів, з яких були створені.

Методи класу отримують в першому аргументі (з ім'ям *self*) посилання на оброблюваний об'єкт – екземпляр. Присвоювання атрибутів через посилання *self* створює чи змінює дані **екземпляру**, а не класу.

ПРОСТІР ІМЕН

```
class Cls_Empty :  
    pass
```

Список імен поточної області видимості

```
dir() → ['In', 'Out', ....., Cls_Empty', ...]
```

```
dir(Cls_Empty)
```

Список атрибутів класу

```
→ ['__class__', '__doc__', '__dict__', .....] ]
```

```
Cls_empty.__dict__
```

Словник класу

```
→ ({'__module__': '__main__',  
     '__doc__': None,  
     '__dict__': <...>, .....
```

```
Cls_empty.X = 25; Cls_empty.Y = 42  
Cls_empty.__dict__
```

Словник класу

```
→ ({..., 'X': 25, 'Y': 42})
```

!!! X Y

ПРОСТІР ІМЕН

ex_1 = Cls_empty()

ex_2 = Cls_empty()

ex_1.X → 25

ex_2.Y → 42

ex_1.X = 'НОНОНО'

ex_2.Y = '146823'

ex_1.X → 'НОНОНО'

ex_1.Y → 42

ex_2.X → 25

ex_2.Y → 146823

2 екземпляри класу

Змінюються атрибути
екземплярів, но не класу

cl_empty.X → 25

cl_empty.Y → 42

ПРОСТІ ІМЕНА

```
class MixedData :
```

data - атрибут класу

```
    data = 'STRING'
```

data - атрибут екземпляру

```
    def __init__(self, value):
```

```
        self.data = value
```

Функція друку атрибуту
класу та атрибуту
екземпляру

```
    def dispout(self):
```

```
        print (self.data, MixedData.data)
```

```
x = MixedData ('FIRST')
```

```
y = MixedData (2)
```

```
x.dispout()
```

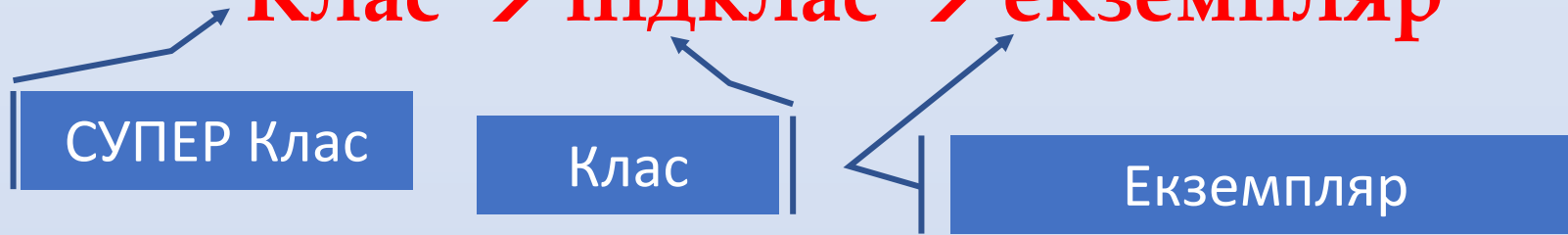
→ *FIRST STRING*

```
y.dispout()
```

→ *2 STRING*

СПАДКУВАННЯ: СУПЕРКЛАС, ПІДКЛАС

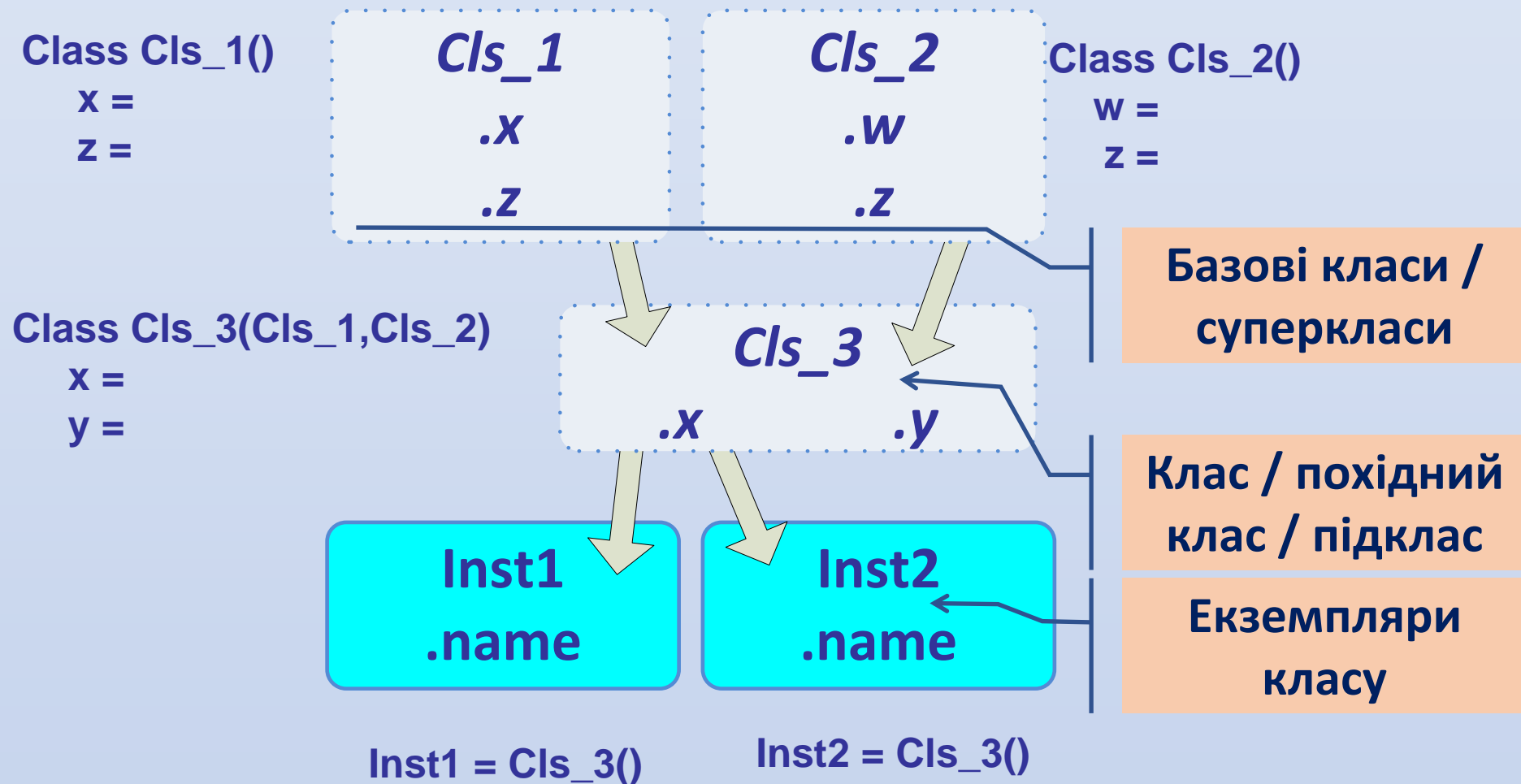
Клас → підклас → екземпляр



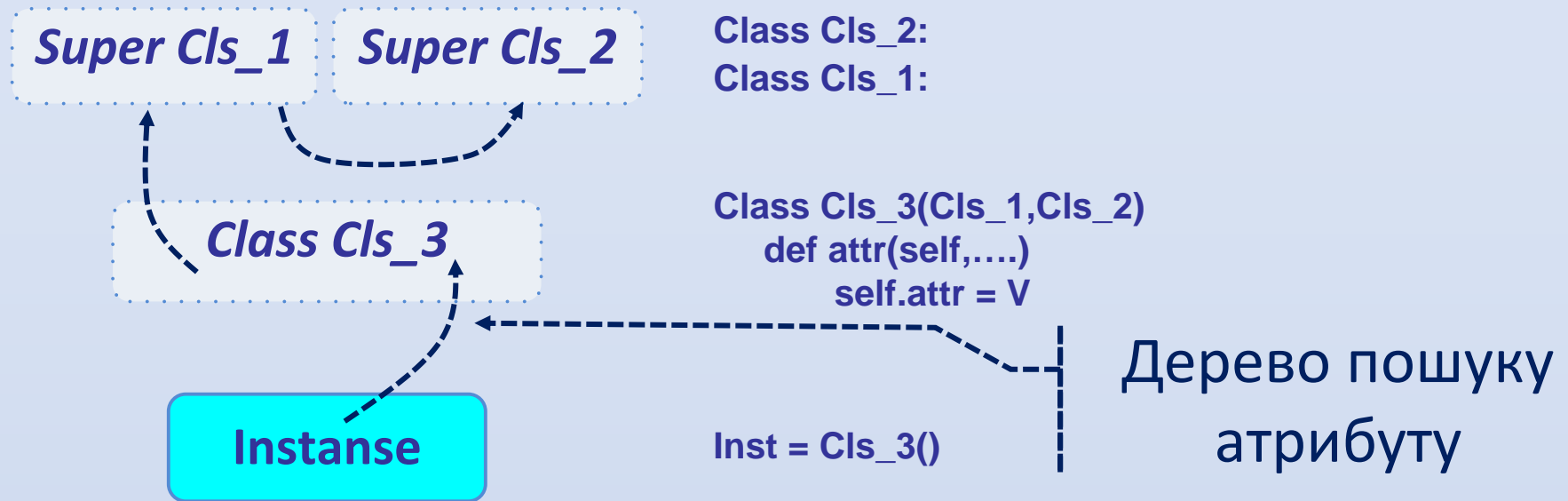
Суперкласи перераховуються в круглих дужках в заголовку інструкції *class*. Клас, що наслідує, називається **підкласом**, а клас, який унаслідується, називається його **супер-класом**.

Клас наслідує атрибути суперкласу, **екземпляр** наслідує атрибути класу.

ДЕРЕВО КЛАСІВ: СУПЕРКЛАС, КЛАС, ЕКЗЕМПЛЯР



ДЕРЕВО УСПАДКУВАННЯ



Атрибути екземплярів створюються за допомогою присвоювання атрибутів аргументу *self* в методах класу. Атрибути класів створюються інструкціями (привласнення), розташованої всередині інструкції **class**.

Посилання на суперклас - перерахування класів в круглих дужках в заголовку інструкції **class**.

СИНТАКСИС КВАЛІФІКАЦІЇ ІМЕН

Для доступу до атрибутів будь-якого об'єкта використовується синтаксис кваліфікації імені *object.attribute* -> це вираз, що повертає значення, яке присвоєно імені атрибуту.

Звернення до імен, кваліфікуючи їх, явно вказує інтерпретатору об'єкт, атрибут якого потрібно використати.

- Прості змінні (правило **LEGB**), наприклад **X** - пошук імені **X** в поточних областях видимості.
- Кваліфіковані імена **X.Y** - пошук імені **X** спочатку в поточних областях видимості, а потім буде виконаний пошук атрибута **Y** в об'єкті **X** (не в областях видимості).
- Кваліфіковані шляхи **X.Y.Z** - буде проведений пошук імені **Y** в об'єкті **X**, а потім пошук імені **Z** в об'єкті **X.Y**.

СИНТАКСИС КВАЛІФІКАЦІЇ ІМЕН

додаток для класів

Пошук в дереві успадкування кваліфікованих імен типу *object.X*

Посилання *object.X* → пошук атрибута *X* виконується спочатку в об'єкті *object*, потім у всіх класах, розташованих вище в дереві спадкоємства.

Присвоєння *object.X = value* → створюється чи змінюється атрибут з ім'ям *X* в просторі імен об'єкта *object*, **і нічого більше.**

ВБУДОВАНІ АРГУМЕНТИ

КЛАСУ

Атрибут	
<code>__name__</code>	Ім'я класу
<code>__module__</code>	Ім'я модуля
<code>__dict__</code>	Словник атрибутів класу
<code>__bases__</code>	Кортеж суперкласів
<code>__doc__</code>	Рядок документації класу

ВБУДОВАНІ АРГУМЕНТИ

ЕКЗЕМПЛЯРУ

Атрибут	
<code>__dict__</code>	Словник атрибутів класу
<code>__class__</code>	Клас, що є «батьком» екземпляру
<code>__init__</code>	Конструктор
<code>__del__</code>	Деструктор (фіналізатор)
<code>__cmp__</code>	Викликається для порівняння (немає в версіях ≥ 3.0)
<code>__hash__</code>	Хеш-значення об'єкту
<code>__getattr__</code>	Вертає значення атрибуту
<code>__setattr__</code>	Встановлює значення атрибуту
<code>__delattr__</code>	Видаляє атрибут
<code>__call__</code>	Виконується при виклику екземпляру

ВБУДОВАНІ АРГУМЕНТИ (приклад: __class__, __bases__)

```
def clstree(cls, indent):  
    print('.' * indent + cls.__name__)  
    for supercls in cls.__bases__:    #!рекурсія суперкласів  
        clstree(supercls, indent+4)  
def instancetree(inst):  
    print('tree of ', inst)  
    clastre(inst.__class__, 4)
```

```
class A          : pass  
class B(A)       : pass  
class C(A)       : pass  
class D(B,C)     : pass  
class E          : pass  
class F(D,E)     : pass
```

```
instancetree(B())
```

що буде надруковано?

Рекомендована ЛІТЕРАТУРА

- **Програмування числових методів мовою Python:** підруч. / А. В. Анісімов, А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий ; за ред. А. В. Анісімова. – К. : Видавничо-поліграфічний центр "Київський університет", 2014. – 640 с.
- **Програмування числових методів мовою Python:** навч. посіб. / А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий, Є. В. Глушко ; за ред. А. В. Анісімова. – К. : Видавничо-поліграфічний центр "Київський університет", 2013. – 463 с.
- **Основи програмування Python:** Підручник для студ. спеціальності 122 «Компютерні науки» / А.В.Яковенко; КПІ.- Київ: КПІ, 2018 . – 195 с.
- **Лутц М.** Изучаем Python, 4-е издание. - СПб.: Символ-Плюс. 2011.- 1280 с.: ил.

Рекомендовані посилання

- **Python OOP Tutorias – Working with Classes: 6 lessons.** – <https://www.youtube.com/watch?v=ZDa-Z5JzLYM&list=PL-osIE80TeTsghluOqKhwlXsIBldSeYtc>

Контрольні запитання

- Обґрунтуйте необхідність та вкажіть переваги використання класів.
- Поясніть відмінності класу і екземпляру класу. Надайте приклад створення класу и екземпляру.
- Поясніть призначення об'єкту **self** та надайте приклади його застосування.
- Поясніть використання аргументу **__init__** при створенні класу, надайте приклади.
- Поясніть призначення вбудованого аргументу класу **__bases__**.
- Поясніть призначення вбудованого аргументу класу **__doc__**.
- Поясніть призначення вбудованого аргументу екземпляру **__dict__**.

The END
Mod 1. Lec 08.