

СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

**122 «Комп'ютерні науки»
КН-19**

2020 / 2021 навчальний рік

PYTHON #6

1. Рекурсивні функції
2. Непрямі виклики
3. Інтроспекція
4. Анотації функцій
5. Анонімні функції (*lambda*)
6. Відображення на послідовність: (*map*)

https://github.com/eabshkvprof/2020_Mod_Prog_Tech

РЕКУРСИВНА ФУНКЦІЯ

Визначено деяке x_0 .

Обчислення x_n -будується як $x_i = F(x_{i-1})$,
 $i=1,2,\dots, n$.

Кожне обчислення $F(x)$ – ітерація, i –
номер ітерації, процес – *ітеративний*.

З іншої сторони : $x_n = F(F(F(\dots F(x_0)))$ –
рекурентний процес обчислення x_n .

Рекурсія → – метод визначення об'єкту
через раніш визначені об'єкти, серед яких
сам об'єкт.

Ітерація – багаторазове повторення.

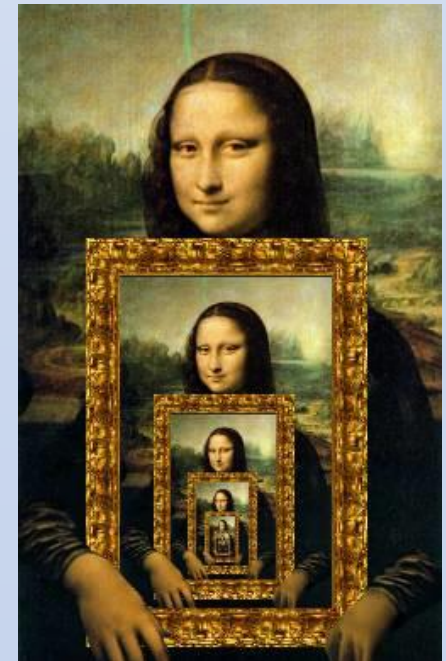
Рекурсія – багаторазове звернення.

РЕКУРСИВНА ФУНКЦІЯ

Теорема → рекурсивне обчислення завжди можна перетворити в ітераційне обчислення (що використовує цикли). І навпаки, будь-яке ітераційне обчислення, що припускає використання циклів, можна реалізувати як рекурсивне.

Рекурсія VS Ітерація → предмет багаторічних суперечок програмістів

	Рекурсія	Ітерація
Запис	Компактний	НЕ компактний
Час	Повільніше	Швидше
Пам'ять	Більше	Менше



РЕКУРСИВНА ФУНКЦІЯ

Рекурсія → – метод визначення об'єкту через раніш визначені об'єкти, серед яких сам об'єкт.

Рекурсивна функція → – це така функція, серед виконуваних інструкція, якою є оператор виклику самої цієї функції.

```
def <name>(arg1, arg2,...,argN) :  
    <statements>  
    name (_arg1_, _arg2_, ... _argN_)  
    <statements>
```

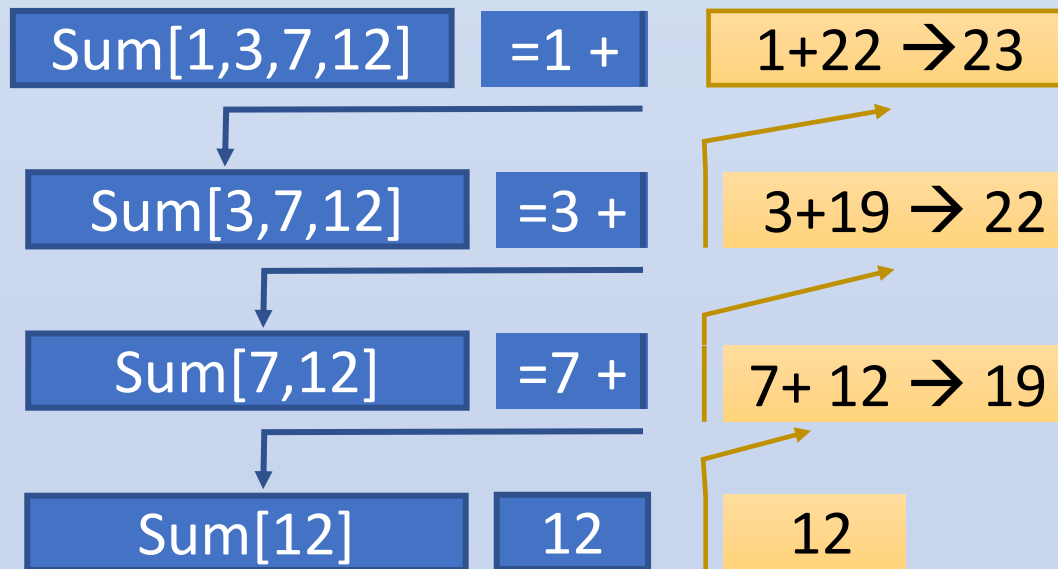
РЕКУРСИВНА ФУНКЦІЯ

```
def listsum (L) :  
    if len(L)==1:  
        return L[0]  
    else:  
        return L[0]+listsum(L[1 :])
```

Нерекурсивна гілка
(вихід з рекурсії, база)

Рекурсивна гілка (тіло)

Print(listsum([1,3,7,12]))



РЕКУРСИВНА ФУНКЦІЯ

Глибина рекурсії - кількість рекурсивних входжень в функцію.

Обмеження - максимальна допустима глибина рекурсії.

Виключна ситуація
exception *RecursionError*

Керування максимальною глибиною рекурсії:

- *sys.getrecursionlimit()*
- *sys.setrecursionlimit(limit)*

НЕПРЯМИ ВИКЛИКИ ФУНКЦІЙ

Python функція - об'єкт, можна працювати з ними, як зі звичайним об'єктом.

Об'єкти функції можуть: присвоюватися, передаватися іншим функціям, зберігатися в структурах даних і так далі (подібно простим числам або рядками).

Об'єкти функції підтримують спеціальні операції: викликатися перерахуванням аргументів в (), наступних відразу ж за виразом функції.

Після виконання *def()*, ім'я функції є посилення на об'єкт - її можна привласнити іншим іменам і викликати функцію за допомогою одного з них.

ІНТРОСПЕКЦІЯ


ІНТРОСПЕКЦІЯ – можливість для будь-якого об'єкта (функції також) отримати всю інформацію про його внутрішню структуру і середовищі виконання.

Дві групи: а) стандартні можливості (описані в документації по мові),
б) нестандартні (характерні для конкретної реалізації мови (наприклад, *CPython*)).

АНОТАЦІЇ

Анотації (короткий опис) не мають ніякого семантичного значення, використовуються в Python тільки для підтримки інформативності коду та його автоматизованого аналізу. Анотування це опція , не вимога мови.

```
def <name>(arg1 : expr, \
            arg2 : expr = value, \
            ..., *args : expr \
            *kwargs : expr ) : ->expr
    <statements>
```



АНОТАЦІЇ

Доступ до анотації через атрибут функції
`__annotations__`

Вертає словник.

```
def foo(a: 'x', b: 5 + 6, c: list) -> max(2, 9): ...
```

```
foo.__annotation__
```

```
{'a': 'x', 'b': 11, 'c': list, 'return': 9}
```

АНОНІМНІ ФУНКЦІЇ (*lambda*)

Створення об'єкту «функція» в формі виразу.

Lambda – це **вираз**!

Lambda - складається з **одного** виразу!

Lambda - вираз вертає функцію, але **НЕ зв'язує** створений об'єкт (функцію) з іменем (змінною).

Головне: можна використовувати там, де def неможливо. Наприклад, в інших виразах.

LAMBDA ВИРАЗ

lambda_expr ::=

lambda **arg1, arg2,...,argN : some_expression**

Lambda вираз

def lamda (arg1, arg2,...,argN) :
***return* some_expression**

Еквівалентна функція

Lambda - вираз не може містити анотацій та інші вирази.

Аргументи та області видимості аналогічні def.../

ОТОБРАЖЕННЯ НА ПОСЛІДОВНІСТЬ (map)

map_function ::=

map (**func**, iterable1, iterable2, ... iterableN)

def func(arg1, arg2, ... argN)



Застосовує **func** до кожного елементу ітеруємої послідовності/ послідовностей.

python ver < 3.0 -> list

python ver 3.0+ -> iterator

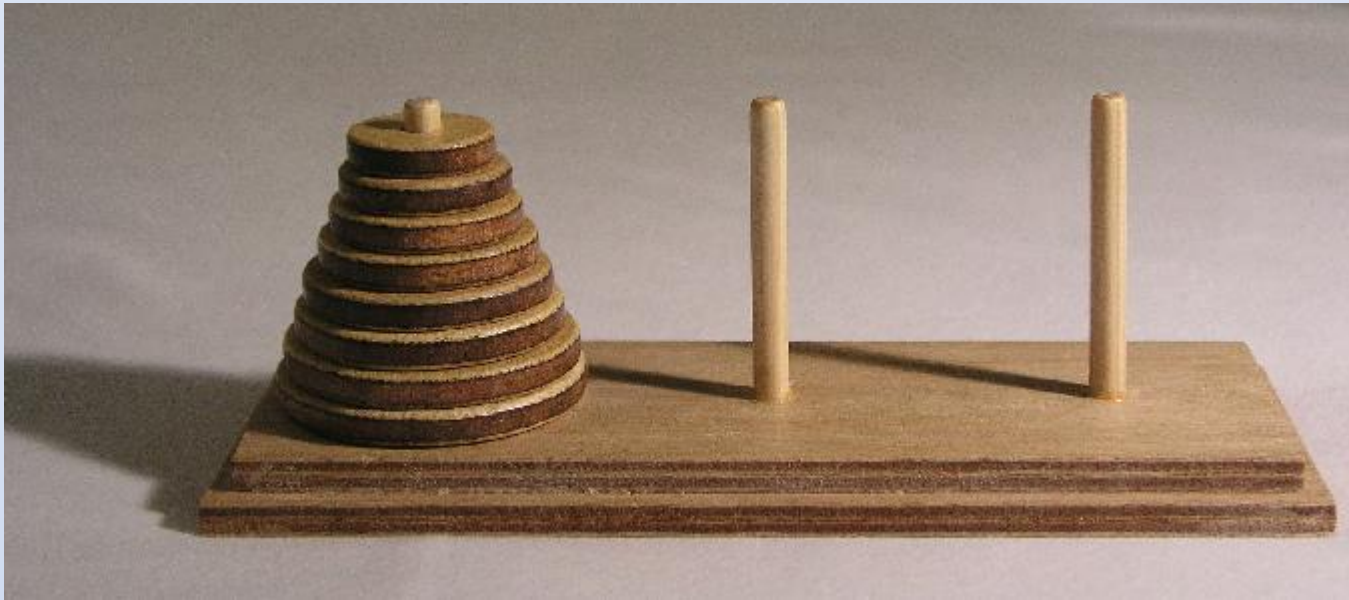
ОТОБРАЖЕННЯ НА ПОСЛІДОВНІСТЬ (map)

Не рекомендовано

! filter () ver < 3.0

! reduce () ver < 3.0

ХАНОЙСЬКІ БАШТИ



Дано три стрижня, на один з яких нанизані вісім кілець (n), причому кільця відрізняються розміром і лежать менше на більшій. Завдання полягає в тому, щоб перенести піраміду з восьми кілець за найменше число ходів на інший стрижень. За один раз дозволяється переносити тільки одне кільце, причому не можна класти більше кільце на меншу.

ХАНОЙСКІ БАШТИ



Написати програму з використанням рекурсивного виклику функції.

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.

Контрольні запитання

- Наведіть визначення рекурсивної функції в мові Python. Надайте приклади створення та використання рекурсивних функцій.
- Визначте поняття анотації функції, надайте приклади створення анотацій та їх використання.
- Наведіть визначення `lambda` виразу, надайте приклади створення та використання **`lambda`** виразів.

The END
Mod 1. Lec 6.