

ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

Тема 10. Модулі та пакети

Поняття модуля

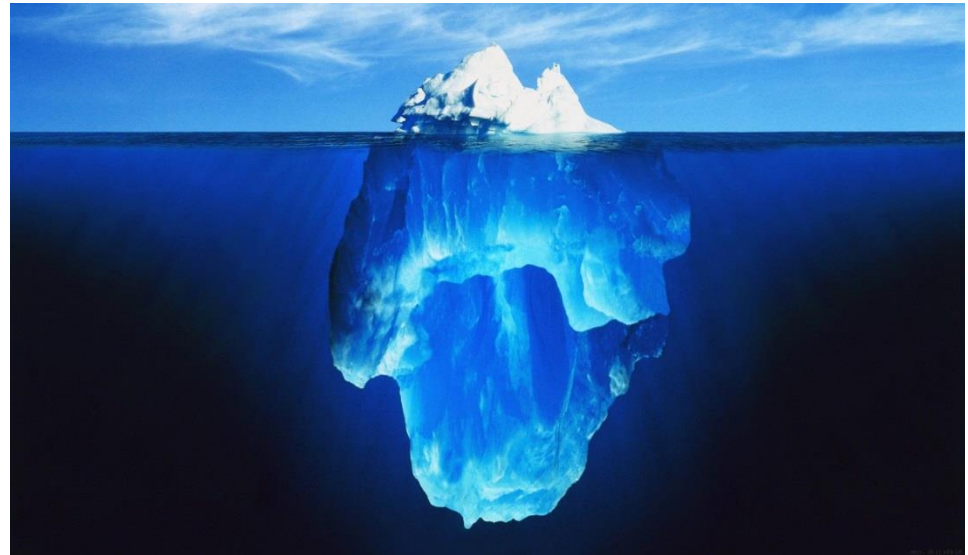
- Використання підпрограм дозволяє підняти рівень абстракції алгоритмів.
- Але підпрограми не дають можливості зберігати разом описи даних – констант та змінних - які необхідні для роботи підпрограм.
- До того ж, для побудови великих програм підпрограми є все ж дуже маленькими „цеглинами”.
- Ці проблеми вирішуються введенням та використанням модулів.

Поняття модуля.2

- **Модуль** об'єднує підпрограми разом з описами констант, змінних, типів, які призначені для розв'язання певного класу задач.
- Характерною рисою модуля є інкапсуляція, тобто приховування певної частини модуля від використання ззовні.
- Взагалі, модуль поділяється на дві частини: видиму (інтерфейсну) та частину реалізації.

Поняття модуля.3

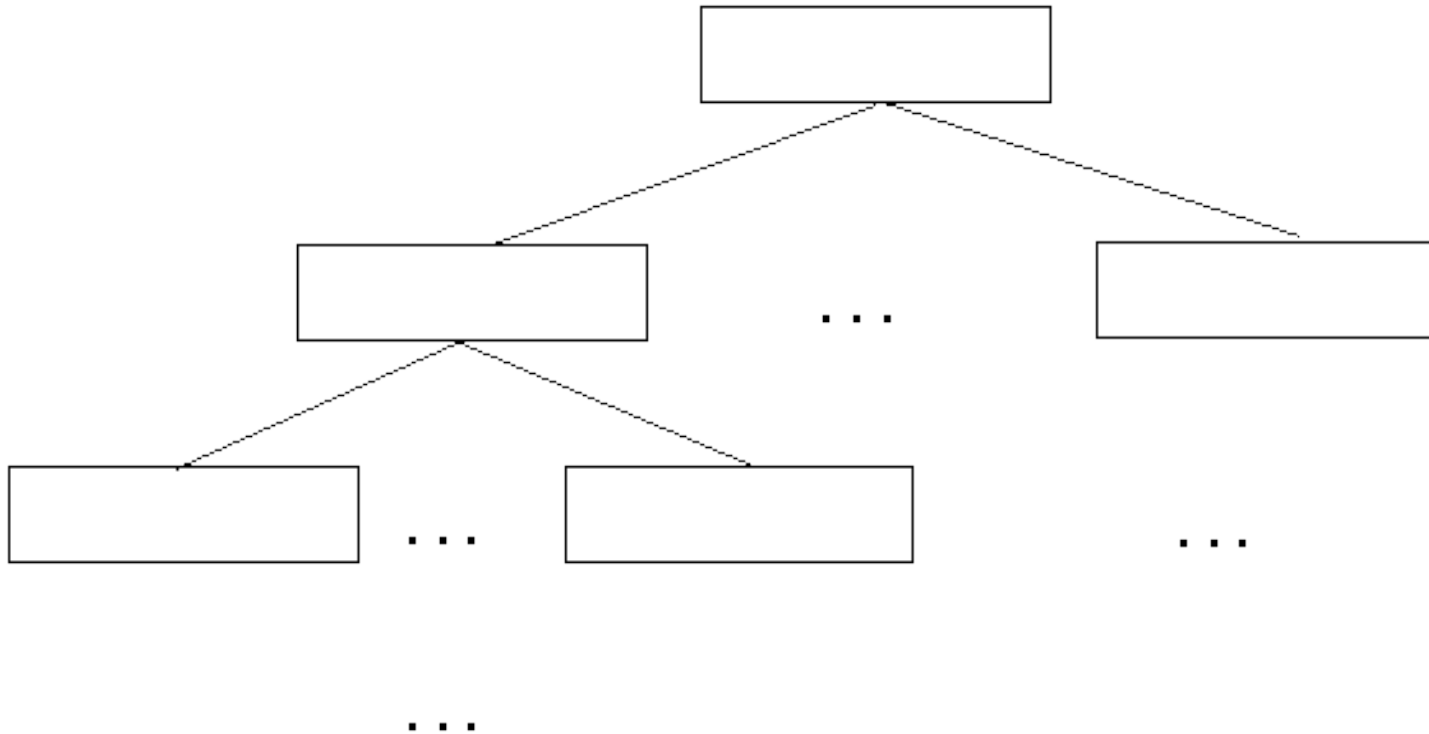
- Всі описи, які вказані у інтерфейсній частині модуля, доступні ззовні.
- Частина ж реалізації модуля є внутрішньою та недоступною для безпосереднього використання.
- У цьому розумінні модуль нагадує айсберг, у якого більша частина знаходиться під водою та не є видимою



Модульна структура програм

- Програма, що має модульну структуру, зазвичай складається з сукупності модулів, серед яких виділяють один головний модуль.
- Частіше за все така програма має ієрархічну структуру, зображену на рисунку (головний модуль - нагорі), хоча у деяких мовах програмування дозволяють циклічні посилання на модулі.

Модульна структура програм.2



Модулі у Python

- Модуль у Python – це окремий файл, який містить програмний код Python.
- Насправді, всі програми, які ми розглядали до цього часу, є модулями з точки зору Python.
- Тож модуль складається з описів функцій а також з ланцюгів команд.
- Ім'я модуля – це ім'я файлу, у якому зберігається модуль, без розширення імені “.py”.
- Наприклад, якщо модуль зберігається у файлі “myunit.py”, то ім'я модуля – “myunit”.
- У модулях Python не виділяється окрема інтерфейсна частина та частина реалізації.
- Всі об'єкти модуля потенційно можуть бути використані у інших модулях.

Імпорт модулів

- Модуль може використовувати інші модулі. Для цього треба їх імпортувати.
- Імпорт модуля позначається так:

import M

- де M – ім'я модуля (у цьому випадку імпортуються всі об'єкти модуля)
- або

from M import x_1, \dots, x_n

- де x_1, \dots, x_n – імена функцій або змінних з модуля M (у цьому випадку імпортуються тільки вказані об'єкти модуля)
- або

from M import *

- (у цьому випадку імпортуються всі об'єкти модуля).
- Від вигляду команди імпорту залежить форма подальшого використання об'єктів модуля.

Використання об'єктів модулів

- Використання об'єктів модуля – це використання змінних у виразах та присвоєннях, а також виклик функцій
- Якщо модуль імпортовано командою

import M

- то для використання об'єкту x_i модуля M треба писати $M.x_i$

- Якщо окремі або всі об'єкти модуля імпортовано командою

from M import x_1, \dots, x_n

- або

from M import *

- то для використання об'єкту x_i модуля M треба писати просто

x_i

Виконання імпортованих модулів

- Виконання імпортованих модулів відбувається тоді, коли Python зустрічає команду **import**.
- При цьому, виконуються всі ланцюги команд імпортованого модуля, які вказані на рівні модуля.
- Звичайно, описані у імпортованому модулі функції будуть виконуватись тоді, коли їх викличуть.

ГОЛОВНИЙ МОДУЛЬ

- Головним модулем у програмі у Python, що складається з декількох модулів, є той, з якого починається виконання програми інтерпретатором.
- Головний модуль отримує від Python зарезервоване ім'я “__main__”. Це ім'я можна використати, щоб визначити, чи є модуль головним, чи імпортованим (ім'я поточного модуля зберігається Python у спеціальній змінній __name__).
- Як правило, таке визначення дозволяє виконувати або не виконувати деякий ланцюг команд в залежності від того, чи є модуль головним.
- Стандартне використання:

```
if __name__ == '__main__':
```

P

Видимість об'єктів модуля

- У Python усі об'єкти імпортованого модуля, які описані на рівні модуля, є видимими. Прихованих змінних або функцій немає.
- Тобто, у Python покладаються на програміста у тому, щоб не використовувати ззовні модуля його внутрішні об'єкти.
- Для позначення змінних або функцій, які не повинні використовуватись ззовні даного модуля, прийнято починати імена таких змінних або функцій підкресленням ('_').
- Наприклад: `_x` або `_f` тощо.
- Треба також зазначити, що коли з модуля імпортують тільки окремі об'єкти командою

from M import x_1, \dots, x_n

- всі інші об'єкти модуля не будуть видимі у модулі, що імпортує даний.

Області дії імен

- Області дії імен задають правила, за якими Python визначає, яку саме змінну використати у даному контексті: на рівні модуля, функції тощо.
- У Python виділяють три області видимості: локальну, глобальну та нелокальну.
- Ми вже давали означення локальних та глобальних змінних у темі «Підпрограми». А саме:
- **Локальними** називають змінні, що вказують на рівні підпрограм.
- **Глобальними** називають змінні, вказані на рівні модуля.
- Їх можна використовувати та змінювати у підпрограмах, якщо у підпрограмі написати оператор `global`. Наприклад,

global x,y

Області дії імен.2

- Нелокальні змінні – це змінні, які вперше визначені у зовнішніх по відношенню до даної функціях (але не на рівні модуля).
- Нелокальні змінні позначаються так:

nonlocal x,y

- Наприклад, наступний програмний код після виконання покаже значення 2 та 2.

```
def g():
```

```
    x = 1
```

```
    def f():
```

```
        nonlocal x
```

```
        y = x + 1
```

```
        x = y
```

```
        return y
```

```
    y = f()
```

```
    print (x, y)
```

```
x = 5
```

```
g()
```

Пакети. Вказання пакетів під час імпорту модулів

- Пакети слугують для об'єднання декількох логічно пов'язаних між собою модулів.
- Якщо модулі у Python еквівалентні файлам, у яких зберігається програмний код, то пакети – це каталоги, що містять файли з програмним кодом, тобто, модулі.
- Якщо у програмі є пакет P1, підпакет P11 та модуль M111, то щоб вказати імпорт модуля M111 ззовні, треба написати

import P1.P11.M111

- Щоб все це працювало, шлях до батьківського каталогу для каталогу P1 повинен бути включений у змінну PYTHONPATH.
 - Попередні версії Python до версії 3.3 вимагали наявності у каталогу, що є пакетом, файлу '__init__.py'.
 - Наявність цього файлу вказувала Python, що даний каталог містить модулі.
 - Починаючи з версії 3.3 файл '__init__.py' став необов'язковим.

Стандартні модулі

- Стандартна бібліотека Python складається з десятків модулів, які виконують різноманітні функції.
- Імпорт та використання стандартних модулів нічим не відрізняється від вже розглянутої процедури.
- Ми вже зустрічались у попередніх темах зі стандартними модулями `math`, `cmath`, `collections`.
- Надалі ми будемо розглядати й інші стандартні модулі Python.

Рядки документації модуля

- Рядок документації модуля, як і рядок документації функції, – це рядок, який обмежений трьома апострофами ''' або трьома подвійними лапками """, і, таким чином, включає декілька фізичних рядків.
- Рядок документації повинен йти на початку модуля та мати нульовий відступ.
- Щоб побачити рядок документації модуля (а також рядки документації функцій модуля), треба у інтерпретаторі набрати
- `help('M')`
 - де M – ім'я модуля.
- Рекомендовано перший фізичний рядок рядка документації починати з великої літери та закінчувати крапкою.
- Другий фізичний рядок залишати порожнім, а з третього, - давати докладний опис модуля.

Функція `dir()`

- Функція `dir()` повертає перелік усіх функцій та глобальних змінних модуля.
- Виклик у інтерпретаторі `dir()` без параметрів повертає дані поточного модуля.
- Якщо у параметрах вказати ім'я модуля, то повертаються дані вказаного модуля (модуль повинен бути імпортований).

Приклад

- Описати модуль для обробки поліномів. Для поліному $P(x)$ задають коефіцієнти та степені.
- Реалізувати такі дії над поліномами:
 - 1) Обчислення значення поліному $P(x)$ у точці x
 - 2) Сума поліномів $P_1(x)$, $P_2(x)$
 - 3) Різниця поліномів $P_1(x)$, $P_2(x)$
 - 4) Добуток поліномів $P_1(x)$, $P_2(x)$
 - 5) Похідна поліному $P(x)$
 - 6) Введення поліному $P(x)$
 - 7) Виведення поліному $P(x)$
- Будемо реалізовувати поліном у вигляді словника, ключами якого є степені, а значеннями, - коефіцієнти.
- Нульові коефіцієнти зберігати не будемо.
- Для введення визначимо дві підпрограми: послідовне введення коефіцієнтів та степенів а також введення рядка, що задає поліном.
- Згодом напишемо головний модуль, що імпортує даний.

Імпорт модуля під іншим ім'ям

- Інколи доцільно змінити ім'я модуля при імпорті. Наприклад коли це ім'я є занадто довгим або раніше застосовувалось інше ім'я модуля тощо.
- Для такого імпорту треба написати:

import M as N

- Після цього ім'ям імпортованого модуля з точки зору програми стає N.

Приклад

- Програма, що використовує модуль обробки поліномів (версія 2).

Резюме

- Ми розглянули:
 1. Означення модуля.
 2. Модулі у Python
 3. Імпорт модулів та використання об'єктів модулів
 4. Видмість об'єктів модуля
 5. Області дії імен
 6. Пакети
 7. Стандартні модулі
 8. Рядки документації модуля

Де прочитати

1. A Byte of Python (Russian) Версия 2.01 Swaroop С Н (Translated by Vladimir Smolyar),
<http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf>
2. Бублик В.В., Личман В.В., Обвінцев О.В..
Інформатика та програмування. Електронний конспект лекцій, 2003 р.,
3. Марк Лутц, Изучаем Python, 4-е издание, 2010,
Символ-Плюс
4. Python 3.4.3 documentation
5. Марк Саммерфилд, Программирование на Python 3.
Подробное руководство. - Символ-Плюс, 2009.