**2. Основи мови Python**

## 2.1. Синтаксис мови

Синтаксис мови Python, як і сама мова, достатньо простий. Можна виділити наступні твердження.

* Кінець рядка є кінцем інструкції (прикінцеві символи непотрібні).
* Для надання значень змінним оператором присвоювання є знак дорівнює «=». Формат оператора: ім’я\_змінної=вираз.
* Основна інструкція та вкладені інструкції (вкладений блок інструкцій) записуються відповідно до одного шаблону: основна інструкція завершується двокрапкою, наступними рядками розташовуються вкладені інструкції, з однаковим відступом на початку рядків по відношенню до основної інструкції. Наприклад:

Основна інструкція:

Вкладений блок інструкцій

Тобто вкладені інструкції об'єднуються в блоки за величиною відступів. Відступ може бути будь-яким, головне, щоб в межах одного вкладеного блоку відступ був однаковий. Не варто забувати про читабельність коду, так, відступ в 1 пропуск є малочитабельним. Рекомендується використовувати 4 пропуски.

Можна бачити, що синтаксис оформлення основної інструкції та вкладеного блоку інструкцій істотно відрізняється від синтаксису більшості мов, в яких використовуються операторні дужки для виділення вкладеного блоку інструкції (наприклад, begin … end в Паскалі або { … } в Сі).

* Розмір літер має значення, тобто великі і маленькі літери вважаються різними. Більшість службових слів (окрім: False, None, True) та вбудованих функцій пишуться маленькими літерами.

Проте існує декілька спеціальних випадків:

* Можна записати кілька інструкцій в одному рядку, розділяючи їх крапкою з комою:

a = 1; b = 2; print(a, b)

* Можна записувати одну інструкцію в декілька рядків. Для цього необхідно розмістити її в парі круглих, квадратних або фігурних дужок:

if (a < 1 and b < 2 and c < 3 and d < 4):

print('spam' \* 3)

* Якщо тіло вкладеної інструкції містить єдиний оператор, то він може розташовуватися в тому ж рядку, що і основна інструкція:

if x > y: print(x)

Крім конструкцій мови, програма може містити коментарі.

***Коментар*** – це довільний текст у будь-якому місці програми, що пишеться після символу #, і представляє інтерес лише як замітка для того, хто буде переглядати код програми.

Наприклад:

print('Привіт, Світ!') # print -- це функція або:

# print -- це функція print('Привіт, Світ!')

Коментарі містять пояснювальні тексти і полегшують читання і розуміння програм, тому намагайтеся в своїх програмах писати якомога більше корисних коментарів, які містять: припущення, важливі рішення, важливі деталі, проблеми, які ви намагаєтеся вирішити, проблеми, яких ви намагаєтеся уникнути і т.д.

## 2.2. Запуск програм

Виконання програм на комп'ютері здійснюється операційною системою. В завдання операційної системи входить виділення ресурсів (оперативної пам'яті та ін.) для програми, заборона або дозвіл на доступ до пристроїв введення / виведення і. т.д.

Для запуску програм, написаних мовою Python, окрім операційної системи необхідна програма-інтерпретатор (віртуальна машина) Python (Рис. 2.1). Як зазначалося раніше, ця програма приховує від Pythonпрограміста всі особливості операційної системи, тому, написавши програму мовою Python в системі Windows, її можна запустити, наприклад, в GNU/Linux і отримати такий же результат її виконання.

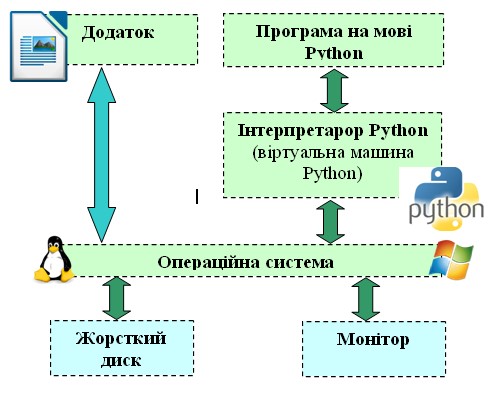


Рис. 2.1. Схема орієнтовної роботи програмі на мові Python

Існує два способи запуску програм, написаних мовою Python:

використання інтерактивного режиму інтерпретатора (інтерактивний режим) та використання файлу з текстом програми (пакетний режим)

## Інтерактивний режим роботи

Подібно мовам Lisp та Prolog в режимі налагодження, інтерпретатор Python має інтерактивний режим роботи, при якому введений з клавіатури вираз відразу ж виконується, а результат виконання виводиться на екран. Цей режим може бути цікавий не тільки початківцям, але й досвідченим програмістам, які мають можливість протестувати в інтерактивному режимі будь-який фрагмент коду, перш ніж використовувати його в основній програмі.

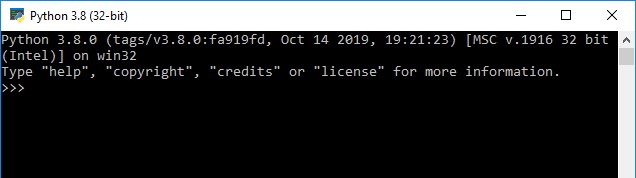
Для запуску інтерпретатора Python в інтерактивному режимі необхідно в терміналі чи командному рядку ввести команду python або python3 та натиснути Enter. Користувачі Windows можуть запустити інтерпретатор в командному рядку, якщо належним чином встановлено змінну PATH або з директорії (папки), в якій встановлено Python.

В комплекті разом з інтерпретатором Python йде IDLE (Integrated Development and Learning Environment - інтегроване середовище розробки).

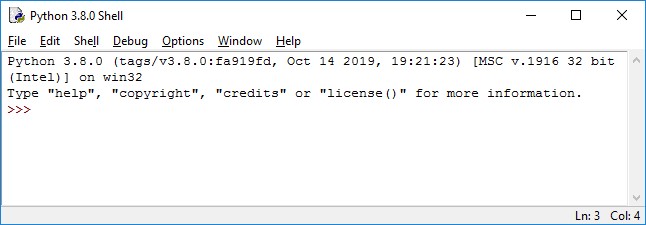
За своєю суттю воно подібне до інтерпретатора, запущеного в командному рядку, але має розширений набір можливостей (підсвічування синтаксису, перегляд об'єктів, налагодження і т.п.).

Для запуску IDLE в Windows необхідно вибрати ярлик "IDLE (Python 3.х GUI - XX bit)" з списку встановлених програмних засобів в меню "Пуск". В GNU Linux оболонка IDLE за замовчуванням відсутня, тому її попередньо потрібно встановити. Також можна скористатися online консоллю для Python (https://www.python.org/shell/).

В результаті запуску Python в інтерактивному режимі буде виведено запит командного рядка в вигляді символів «>>>». Після вказаних символів і можна набирати команди для виконання (Рис. 2.2).



а) текстове вікно інтерактивного режиму



б) графічне вікно інтерактивного режиму

Рис. 2.2. Вікно інтерактивного режиму

В інтерактивному режимі Python можна використовувати як калькулятор для різноманітних обчислень, а якщо додатково підключити необхідні математичні бібліотеки, то за своїми можливостями він стає практично рівним таким пакетам, як Matlab, Octave і т. п.

Наприклад, якщо ввести вираз 2+3 і натиснути кнопку Enter, то ви отримаєте число 5, яке є результатом обчислення. Зверніть увагу, що інтерпретатор Python видає результат роботи введеної команди негайно.

>>>2+3

5

Якщо ви використовуєте IDLE або оболонку GNU/Linux чи BSD, для виходу з командного рядка інтерпретатора необхідно натиснути комбінацію кнопок Ctrl+D або виконати команду exit() (не забудьте написати дужки, "()"). Якщо ж використовуєте командний рядок Windows, для виходу з командного рядка інтерпретатора необхідно натиснути комбінацію кнопок Ctrl+Z, а потім кнопку Enter.

## Пакетний режим роботи

Набирати програму в командному рядку інтерпретатора кожний раз за необхідності її виконання не досить зручно. Краще зберегти програму в файл, щоб потім, за необхідності, мати можливість запустити її.

Перш ніж приступити до написання та збереження програми мовою Python, необхідний редактор для роботи з такими файлами. Вибір редактора є вкрай важливим, використання зручного редактора допоможе зробити розробку програм більш комфортною. Досить важливим для такого редактора можна вважати інтерактивні підказки, підсвічування синтаксису та автоматизацію вставляння відступів при написанні вкладених інструкцій.

Редактор для програм мовою Python можна обрати серед великої кількості текстових редакторів (наприклад: Notepad++, Sublime Text, Komodo Edit, Vim, Emacs), спеціалізованих оболонок (наприклад: Thonny, PyCharm, Wing IDE) або використовувати вбудований редактор IDLE Python. Не рекомендується використовувати Блокнот Windows, оскільки він не має функції підсвічування синтаксису, а також не дозволяє автоматично вставляти відступи, що дуже важливо для програм, які пишуться мовою Python.

Напишемо нашу першу програму, як редактор для її написання будемо використовувати редактор, вбудований в оболонку IDLE Python.

В запущеній оболонці IDLE Python оберіть послугу головного меню File/New File і в відкритому вікні введіть команду: print('Привіт, Світ!')

Ця команда буде виводити повідомлення 'Привіт, Світ!' на екран (детально функція print() буде розглянута пізніше).

Щоб зберегти програму, скористайтеся послугою головного меню File/Save та вкажіть ім’я файлу, в якому буде збережено програму, наприклад hello.py.

Для запуску програми на виконання необхідно скористатися послугою головного меню Run\Run Module, або натиснути функціональну кнопку F5. В результаті буде активовано оболонку IDLE Python, в якій і виведеться результат роботи програми (Рис. 2.3).

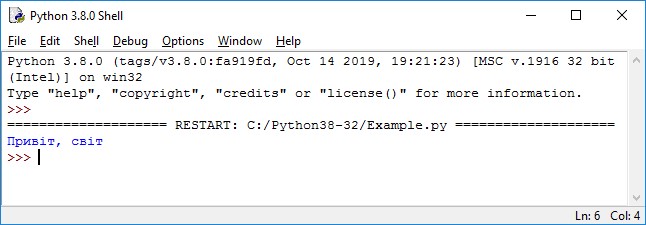


Рис. 2.3.

Якщо необхідно запустити програму, збережену в файлі, з терміналу або командного рядка потрібно в командному рядку викликати інтерпретатор Python і в якості аргументу передати ім’я файлу з програмою. Наприклад, для файлу з ім'ям hello.py команда запуску буде виглядати так: python hello.py або python3 hello.py. Зрозуміло, що поточним каталогом має бути каталог, в якому знаходиться файл, програми інакше необхідно вказати шлях то файлу з програмою.

В ОС Windows файл з Python-програмою (файл з розширенням .py) можна запустити як звичайний виконуваний файл. Тобто знайшовши файл на диску, запустити його подвійним клацанням лівої кнопки миші.

## 2.3. Змінні та літерали

***Ідентифікатор*** – це ім’я деякої сутності (змінної, функції, класу) в програмі для її позначення. При виборі ідентифікаторів необхідно дотримуватися таких правил:

* Першим символом ідентифікатора може бути буква з алфавіту (символ ASCII в верхньому або нижньому регістрі, або символ Unicode) або символ підкреслення «\_».
* Інша частина ідентифікатора (всі символи крім першого) може складатися з букв (символи ASCII в верхньому або нижньому регістрі, а також символи Unicode), символу підкреслення «\_» або цифр (0-9).
* Імена ідентифікаторів чутливі до регістру. Наприклад, myname і myName - це два різні ідентифікатори (зверніть увагу на регістр літер "n" та "N").
* Ідентифікатор не може співпадати з ключовими (зарезервованими) словом інтерпретатора Python. Перелік ключових слів можна отримати, виконавши послідовно дві команди: import keyword та print(keyword.kwlist). Для версії 3.7.1 це: 'False', 'None', 'True',

'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'.

Приклади допустимих імен ідентифікаторів: a, \_\_my\_name, name\_23, a1b2\_c3 і будь\_які\_символи\_utf8\_δξёѐўЩΞέά.

Приклади неприпустимих імен ідентифікаторів: 2things, тут є пропуск, my-name і "це\_в\_кавичках".

**Змінні.**

При написанні програм досить часто необхідно зберігати різні дані та мати можливість маніпулювання цими даними. Ось тут як раз і знадобляться змінні. Слово «змінні» говорить сама за себе - їх значення може змінюватися, а значить, можна зберігати в змінній все, що завгодно. ***Змінна в мові Python*** – це просто посилання на область пам'яті комп'ютера, в якій зберігається деякі дані (посилання на деякий об’єкт). Для задання імен змінних використовуються ідентифікатори. В змінних можуть зберігати значення різних типів.

**Літерали.**

Літерал (literal – константа) – постійне значення певного типу даних, записане у вихідному коді комп'ютерної програми. Прикладом літералу може бути число (наприклад: 5, 1.23, 9.25e-3) або рядок (наприклад, 'Це рядок', "It's a string!"). Значення літералу використовується буквально, число 2 завжди представляє саме себе і нічого іншого - це «константа», тому що її значення не можна змінити. В мові Python немає окремої синтаксичної конструкції для оголошення літералів. Досить часто для вказання того, що змінну не варто змінювати, її пишуть великими літерами (це вказівка розробнику, а не

# інтерпретатору).

## 2.4. Типи даних

Мова Python відноситься до мов з неявною строгою динамічною типізацією.

***Неявна типізація*** означає, що при оголошенні змінної її тип не вказується (при явній типізації тип змінної вказується обов’язково). Як приклад мов з явною типізацією можна привести Pascal, C++. Ось як буде виглядати оголошення цілочисельної змінної у мовах Pascal, C++ та Python.

Pascal:

Var a: Integer;

C++:

int a=1;

Python:

a=1

Для мов з ***динамічною типізацією*** тип змінної визначається безпосередньо при виконанні програми. Окрім того, можна зазначити:

* будь-яка змінна є посиланням;
* типом змінної є те, на що вона посилається;
* тип змінної може довільно змінюватися по ходу виконання коду, коли змінна починає посилатися на інший об’єкт.

Тому замість «присвоєння значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». З іншого боку, якщо мова має статичну типізацію, то тип змінної визначається на етапі компіляції і не може бути змінений на всьому протязі свого життєвого циклу.

***Строга типізація*** (сильна типізація або strong typing) не дає можливості проводити операції у виразах з даними різних несумісних типів. Не строга типізація (слабка типізація або weakly typed) дає можливість виконати операції над несумісними типами даних і отримати деякий непередбачуваний результат. Тобто, у мовах з строгою типізацією ви не можете складати наприклад рядки і числа, необхідне приведення до одного типу. Строго типізованими мовами є: Pascal та Python, до мов з не строгою типізацією відносяться - С і С++.

### 2.4.1 Вбудовані типи даних

Тип даних – це множина значень і операцій на цими значеннями. В Python типи даних можна розділити на вбудовані в інтерпретатор (built-in) і невбудовані, які можна використовувати при імпортуванні відповідних модулів.

До основних вбудованих типів відносяться:

1. None (невизначене значення змінної)
2. Логічний тип ( Boolean Type ) 3. Числа ( Numeric Type )
   1. int – ціле число
   2. float – число з плаваючою точкою (дійсне число)
   3. complex – комплексне число
3. Послідовності ( Sequence Type )
   1. list – список
   2. tuple – кортеж
   3. range – діапазон
4. Рядки (Text Sequence Type )
   1. str
5. Бінарні послідовності ( Binary Sequence Types )
   1. bytes – байти
   2. bytearray – масиви байт
   3. memoryview – спеціальні об'єкти для доступу до внутрішніх даних

об'єкта через protocol buffer

1. Множини ( Set Types )
   1. set – множина
   2. frozenset – незмінювана множина
2. Словники ( Mapping Types )
   1. dict – словник

### 2.4.2. Ініціалізація змінних

Враховуючи неявну типізацію мови Python при оголошенні змінної, їй повинно бути надане значення (вона має бути ініціалізована). Щоб оголосити та ініціалізувати змінну? необхідно написати її ім’я, потім поставити оператор присвоєння (знак рівності «=») і вказати значення, з яким дана змінна буде створена. Наприклад: z=5.

При ініціалізації змінної на рівні інтерпретатора відбувається наступне:

* створюється цілочисельний об’єкт 5 (можна представити, що в цей момент виділяється комірка пам’яті певного об’єму і в неї записується число 5);
* за оператором «=» відбувається зв’язування змінної z і цілочисельного об’єкту 5 (визначається адреса отриманого об’єкту 5, яка присвоюється змінній z, тобто в результаті змінна z посилається на об’єкт 5).

Якщо розглянути більш складний вираз, наприклад z=5+26.0. Інтерпретатор обчислює значення виразу справа (5+26.0) і створює об’єкт з отриманим значенням (31.0). Далі аналогічно до попереднього.

Як було сказано раніше, Python є об’єктно-орієнтовною мовою програмування, тому об’єктів в мові Python багато. Спробуємо розібратися, як пов’язане поняття об’єкту з змінними та їх значеннями (при ініціалізації змінної спочатку створюється цілочисельний об’єкт). Об’єкт в даному випадку – це абстракція для подання даних певного типу, що розміщуються в пам’яті за адресою. Кожен такий об’єкт має три атрибути: адресу (ідентифікатор об’єкту), значення та тип. Ідентифікатор об’єкту – це унікальна ознака об’єкту (адреса об’єкту в пам’яті), за якою можна відрізнити один об’єкти від іншого. Ідентифікатор представляється в вигляді деякого цілочисельного значення.

Нехай ініціалізовано змінну Z.

>>> Z=134

Для отримання значення змінної достатньо ввести її ім’я.

>>> Z

134

Щоб переглянути ідентифікатор об’єкту, на який посилається певна змінна, призначена функція id().

>>> id(Z)

505998272

Для визначення типу призначена функція type().

>>> type(Z)

<class 'int'>

Цікавий ефект буде при використанні цілих чисел в діапазоні від -5 до 256. Ідентифікатори об’єктів різних змінних з однаковими значеннями в межах зазначеного діапазону завжди будуть однаковими, оскільки для покращення швидкодії інтерпретатор при запуску створює об’єкти з цими значеннями.

>>> z=123

>>> y=123

>>> id(z)

505998096

>>> id(y)

505998096

Також при ініціалізації змінних можливі наступні вирази:

>>> x,y,z=2,3,4

>>> a=b=c=2

В першому випадку змінним x, y, z будуть надані значення відповідно 2, 3, 4. В другому випадку змінним a, b, c буде надане значення 2.

Для прикладу розглянемо задачу. «Дано дві змінні x та y з певним значенням. Написати програму, за якою змінна x отримає значення змінної y, а змінна y отримає значення змінної x». Зважаючи на зазначений принцип роботи інтерпретатора при виконанні присвоєння і можливі вирази для задання значень змінних, дана задача розв’язується застосуванням досить простого виразу: x,y=y,x.

>>> x=500

>>> y=600

>>> x,y=y,x

>>> x 600

>>> y

500

### 2.4.3. Змінювані і незмінювані типи

Всі типи даних в Python належать до однієї з 2-х категорій: змінювані (mutable) та незмінювані (immutable).

До незмінюваних (immutable) типів належать:

* логічні (bool);
* цілі числа (int);
* числа з плаваючою точкою (float);
* комплексні числа (complex);
* кортежи (tuple);
* рядки (str);
* незмінювані множини (frozen set).

До змінюваних ( mutable ) типів відносяться:

* списки (list);
* множини (set);  словники (dict).

Незмінюваність типа даних означає, що створений об'єкт цього типу більше не змінюється (не мутується). При зміні значення змінної (наданні нового значення змінній) створюється новий об’єкт і змінна посилається на нього. В цьому і полягає незмінюваність об’єкту – це не константність, оскільки значення можна змінити, проте це буде вже посилання на інший об’єкт з новим значенням.

Наприклад, якщо ми виконуємо вираз a = 5, то буде створено об'єкт з цілочисельним значенням 5 та унікальним ідентифікатором. В загальному можна зазначити наступне:

* Об’єкт за адресою id1 має тип int і значення 5;
* Змінна a містить адресу id1 (змінна a посилається на значення 5).

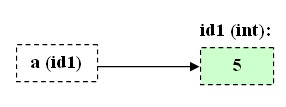


Рис. 2.4. Виконання виразу a=5.

>>> a = 5

>>> id(a)

1672501744

Тобто об’єкт з id=1672501744 буде мати значення 5. Оскільки це цілочисельний тип, який є незмінюваним, то змінити значення цього об’єкту не можна. Проте це не забороняє виконати вираз a = 6 і надати нового значення змінній a.

>>> a = 6

>>> id(a)

1873521735

Тобто створено новий об’єкт з новим ідентифікатором.

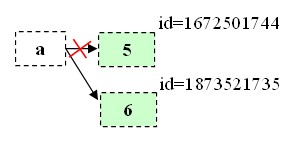


Рис. 2.5. Виконання виразу a=6.

## 2.5. Введення виведення даних

**Виведення результатів роботи програми.**

Для виведення результату роботи програми може бути використана функція print(), яка має наступний формат:

print(value1[, value2, ...][, sep=' '][, end='\n']

[, file=sys.stdout][, flush=False])

Використовуючи функцію print() можна вивести як одне, так і декілька значень. Для виведення декількох значень їх необхідно відділити один від одного комами (value1[, value2, ...]). Значення, що призначені для виведення, можуть бути різного типу, проте при виведенні все рівно вони будуть перетворені в рядковий тип.

Якщо виводиться декілька значень, то при виведенні одне від одного вони будуть відокремлені літералом (символом чи декількома символами), вказаним в параметрі sep (за замовчуванням це пропуск).

Після виведення всіх значень кінець рядку буде доповнений літералом (символом чи декількома символами), вказаними в параметрі end (за замовчуванням символ закінчення рядка та переходу на новий).

За замовчуванням, функція print() виводить значення в стандартний пристрій виведення sys.stdout (екран). За допомогою параметру file можна перенаправити виведення в інше місце – наприклад, до файлу. При цьому, якщо параметр flush має значення False, значення, що виводяться, примусово буде записане до файлу (відбувається автоматичне виштовхування з буферу). Перенаправлення виведення буде розглянуто при розгляді роботи з файлами.

Наприклад:

print(5 + 10)

15 print('Рядок1') print('Рядок2')

Рядок1 Рядок2

print('Рядок1', 'Рядок2')

Рядок1 Рядок2

print('Рядок1', 'Рядок2', sep='-') Рядок1-Рядок2 print('Рядок1', 'Рядок2', end='-') print('Рядок3')

Рядок1 Рядок2-Рядок3

**Введення даних до програми.**

При написанні програм досить часто виникає необхідність в отриманні даних для опрацювання від користувача. Для реалізації такої можливості код програми повинен містити оператори, за якими можна отримати дані, введені користувачем. В мові Python для цього передбачено функцію input([prompt]), за якою можна отримати дані зі стандартного пристрою введення sys.stdin (клавіатура).

Необов’язковий параметр prompt, призначений для вказання запрошення до введення, та буде виведений на стандартний пристрій виведення. Використання функції має формат:

[<змінна>=]input([<prompt >])

Для прикладу розглянемо програму, за якою буде запитуватися ім’я користувача і виводитися привітання. print('Як Вас звати?') name = input() print('Вітаю', name)

Розглянемо ще одну задачу: «Написати програму обчислення суми двох чисел, заданих користувачем». Для розв’язання задачі необхідно: вивести запит користувачу на введення чисел, зчитати два числа, введених користувачем, виконати операцію додавання та результат вивести на екран.

Базуючись на раніше викладеному матеріалі та елементарних знаннях з математики, створимо програму такого вигляду: print('Введіть два числа:') a = input() b = input() s = a + b print(s)

Проте після запуску програми і введення значень, наприклад 2 та 3, в результаті виконання програми отримаємо 23, хоча напевне нами очікувалося 5. Це пов’язане з тим, що за функцією input() повертається значення рядкового типу. А вказана операція «+» для змінних рядкового типу виконує їх об’єднання (детальна робота з рядковими змінними буде розглянута пізніше).

Отже необхідно отримані від функції input() рядкові значення перетворити в цілі або дійсні числа. Для цього можна скористатися функціями int() або float() відповідно.

Тоді програма матиме вигляд:

print('введіть два числа') a = int(input()) b = int(input()) s = a + b print(s)