ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

Тема 1. Лінійні програми

Коротка історія

- 1452-1519 – проект механічної обчислювальної машини Леонардо да Вінчи
- 1623 - перша рахункова машина побудована німцем В.Шикардом
- 1642 - французький вчений Б.Паскаль сконструював механічний обчислювач
- 1673 - німецький вчений Г.В.Лейбніц розробив рахунковий пристрій – арифмометр
- 1815-1864 - Дж.Буль створив свою алгебру - алгебру Буля - яка оперує тільки двома поняттями: Істинно і Хибно
- 1834 - проект аналітичної машини англійського математика Ч.Беббіджа

Коротка історія. 20 століття

- 1940 –перша релейна обчислювальна машина Z3 побудована Конрадом Цузе в Німеччині
- 1946 - перша електронно-обчислювальна машина на радіолампах ENIAC винайдена в США під керівництвом Д.Маучлі і Д.Еккерта
- 1946 - принципи архітектури сучасних комп'ютерів були обгрунтовані Дж. фон Нейманом
- 1949 - перша ЕОМ, заснована на принципах Дж. фон Неймана - EDSAC була створена в Англії М.Уїлксом
- 1950 - перша ЕОМ на території колишнього СРСР (МЕСМ) була створена в Києві за проектом С.О.Лебедєва

Інформатика та інформація

- **Інформатика** - це наука про методи представлення, накопичення, передачі і обробки інформації за допомогою комп’ютерів (computer science).

- **Інформація** - це поняття, що передбачає наявність
  - матеріального носія інформації,
  - джерела і передавача інформації,
  - приймача інформації,
  - каналу зв'язку між джерелом і приймачем інформації.

Алгоритми та виконавці

- **Алгоритм** - точний припис, який визначає зміст і послідовність кроків, що переводять задану сукупність початкових даних у шуканий результат за скінченний час.

- **Виконавцем** ми будемо називати пристрій, здатний виконувати дії із заданого набору дій.
  - Команду на виконання окремої дії звичайно називають оператором або інструкцією.

Приклади виконавців

- пральна машина,
- смартфон,
- мультиварка,
- комп'ютер

- Приклади інструкцій:
  - виконати прання бавовняної білизни,
  - встановити з'єднання із заданим номером,
  - приготувати плов

Програми та програмування

- **Програма** – запис алгоритму у формі, придатній для виконання комп’ютером.

- **Програмування** – процес побудови комп’ютерних програм.
  - У більш широкому сенсі під програмуванням розуміють весь спектр діяльності, пов'язаний зі створенням і підтримкою в робочому стані програмного забезпечення (software engineering)

Компілятори та інтерпретатори

- **Компілятор** – програма, що здійснює трансляцію програми, складеної у високорівневій мові програмування, в еквівалентну програму низькорівневою мовою, близькою до машинного коду.

- **Інтерпретатор** – програма, що здійснює покомандний аналіз програми, її обробку та виконання.

Основні етапи побудови програми

- Введення (набір) тексту
- Перевірка синтаксичної правильності
- [Компіляція]
- Виконання

Мова Python

- Створений на початку 1990-х років Гвідо ван Россумом.
  - Назва походить від популярної гумористичної передачі Monty Python’s Flying Circus

- Основні версії Python
  - Python 1.0 - січень 1994
  - Python 2.0 - 16 жовтня 2000
    - Python 2.7 - 3 липня 2010
  - Python 3.0 - 3 грудня 2008
    - Python 3.4 - 16 березня 2014
    - Python 3.5.2 - 27 червня 2016

Чому Python?

- проста мова
- потужна мова
- широко розповсюджений
- має інструменти для наукових розрахунків
- універсальна мова

Де використовують

- Компанія Google використовує Python у своїй пошуковій системі
- Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm і IBM, використовують Python для тестування апаратного забезпечення
- Служба колективного використання відеоматеріалів YouTube в значній мірі реалізована на Python
- NSA (АНБ США) використовує Python для шифрування і аналізу розвідданих
- Компанії JPMorgan Chase, UBS, Getco і Citadel застосовують Python для прогнозування фінансового ринку
- Популярна програма BitTorrent для обміну файлами в пірінгових мережах написана мовою Python
- Популярний веб-фреймворк App Engine від компанії Google використовує Python як прикладну мову програмування
- NASA, Los Alamos, JPL і Fermilab використовують Python для наукових обчислень.

Звідки завантажити

- https://www.python.org/downloads/ для Windows, Linux/Unix, Mac OS
  - у багатьох інсталяціях Linux та Mac OS інтерпретатор Python вже включений і його не треба завантажувати. Щоб перевірити, чи встановлено Python, треба набрати команду python -V
- https://github.com/qpython-android/qpython3/releases для Android
- https://itunes.apple.com/us/app/python-3.4-for-ios/id583476348?mt=8 або https://itunes.apple.com/ua/app/pythoni3.3/id585633449?mt=8 для IOS

Як запустити

- Для Windows
  - Запуск з командного рядка
    - Перед тим, як запускати інтерпретатор вперше, треба змінити так звану «змінну середовища» PATH, дописавши до неї «;C:\Python34»
    - Запуск виконується так: спочатку у пункті меню «Виконати» набираємо **cmd**. Відкривається вікно з системною підказкою. Після цього набираємо **python**. Отримуємо інформацію про встановлену версію інтрепретатора та підказку інтерпретатора **>>>**.
  - Запуск середовища розробки IDLE з інтерфейсу Windows
    - «Пуск» → «Программы» → «Python 3.4» → «IDLE(Python GUI)».

Як запустити.2

- Для UNIX/Linux/Mac OS у вікні емулятора терміналу набираємо **python**. Отримуємо інформацію про встановлену версію інтрепретатора та підказку інтерпретатора **>>>**.

- Для Android/IOS – звичайний запуск програми натисненням на її піктограму, після чого вибрати «Console» для запуску інтерпретатора або «Editor» для запуску редактора.

Як запустити.3

- Завершення роботи з інтерпретатором
  - Для Windows
    - Якщо інтерпретатор запущено з командного рядка, то натиснути **<Ctrl+Z>** та **<Enter>**
    - Для середовища IDLE просто використовуємо інтерфейс для виходу.
  - Для UNIX/Linux/Mac OS натиснути **<Ctrl+D>** або набрати **exit()**.
  - Для Android/IOS – звичайне завершення програми або набрати **exit()**.

Константи

- Цілі числа будемо записувати в десятковій позиційній системі числення.
- Дійсні - у вигляді десяткового неперіодичного дробу. Будемо відділяти дробову частину від цілої крапкою.

- Приклади запису чисел:
  - 0,
  - 1,
  - +31,
  - -176,
  - 3.14159.

Рядки-константи

- Рядки-константи беруть у апострофи
- `'це рядок'`
  - або у подвійні лапки
- `"це теж рядок"`
  - або у потрійні апострофи
- `'''це також`
- `рядок'''`
  - або у потрійні подвійні лапки
- `"""навіть це також`
- `рядок"""`

Змінні

- Будь-яка **змінна** має ім’я та значення.
  - Ім’я – це ідентифікатор, а значення – константа.
  - Значення змінної може бути визначено або не визначено.

- **Ідентифікатор** - це слово, складене з літер і цифр, на першому місці якого обов'язково знаходиться літера.
- Приклади ідентифікаторів:
  - x,
  - pi,
  - a1,
  - max3,
  - _E2,
  - sigma.

Основні команди Python

- Присвоєння
- Введення
- Виведення
- Тотожня команда

Арифметичний вираз. Множина операцій

- Позначимо множину операцій

Ω = {+, -, *, /, //, %, **}

+ - додавання

- - віднімання

* - множення

/ - ділення

// - ділення націло

% - остача від ділення

** - піднесення до степеня

Арифметичний вираз. Означення

- **Арифметичним виразом** назвемо вираз e, який визначається індуктивно:
  - 1. Якщо *e* - числова константа, то *e* - арифметичний вираз;
  - 2. Якщо *e* - змінна, то *e* – арифметичний вираз;
  - 3. Якщо e_1, e_2 - арифметичні вирази, ω ϵ Ω - арифметична операція, то $e = e_1$ ω e_2 - арифметичний вираз;
  - 4. Якщо e_1 - арифметичний вираз, то $e = (e_1)$ - арифметичний вираз.

- Приклади арифметичних виразів:
- 1, 1+2, x, (x+1), (x+1)+z, 1+2*x, 2*(x+y)

Арифметичний вираз. Обчислення

- Пріоритет операцій:

Операції
**
*, /, //, %
+, -

  - Спочатку обчислюється результат операцій вищого пріоритету.
  - Операції однакового пріоритету обчислюються зліва направо.
  - Порядок обчислення може бути змінений дужками.

- Наприклад x/y*z буде обчислюватись як (x/y)*z.

Команда присвоєння

- Синтаксис:

```
x = e
```

  - де *x* – змінна, *e* – вираз.

- Правило присвоєння:
  - Python обчислює значення виразу *e* у правій частині присвоєння та робить його значенням змінної *x*. При цьому попереднє значення змінної *x* стає недосяжним.
- Приклади присвоєнь:

```
r = s
s = 1
v = i * t
x = x + 1
a2 = (b - c) / (c + d)
```

Команда введення

- Синтаксис:

```
x = input(S)
```

  - де *x* – змінна, *S* – рядок підказки.

  - Для введення цілого числа, треба писати

```
x = int(input(S))
```

  - Для введення дійсного числа, треба писати

```
x = float(input(S))
```

- Правило введення:
  - Python очікує введення значення змінної з клавіатури. Після введення робить його значенням змінної *x*. При цьому попереднє значення змінної *x* стає недосяжним.
- Приклади введення:

```
m = int(input('Введіть m: '))
y = float(input('y=? '))
```

Команда виведення

- Синтаксис:

```
print(e1,...,en)
```

  - де e_i – вирази.

- Правило виведення:
  - Python виводить на екран значення виразів e_i.

- Приклади виведення:

```
print('m=',m,'y=',y)
```

Тотожня команда

- Синтаксис:

```
pass
```

- Правило виконання тотожньої команди:
  - Python нічого не робить.

Спеціальні команди присвоєння

- Доволі часто зустрічаються ситуації коли треба виконувати присвоєння такого вигляду:

$x = x \ \omega \ e$

  - де ω – деяка операція
- У подібних випадках можна застосовувати скорочену форму команди присвоєння

$x \ \omega= e$

Спеціальні команди присвоєння. 2

- А саме:

```
x += e
x -= e
x *= e
x /= e
x **= e
x //= e
x %= e
```

- Наприклад

```
a += b
```

замість

```
a = a + b
```

Ланцюг команд. Визначення

- Визначимо **ланцюг команд** наступним чином:
  1. Якщо P – команда присвоєння, введення, виведення або тотожня команда, то P – ланцюг.
  2. Якщо P, Q – ланцюги, то

  P

  Q

  - ланцюг.

  Також

  $P;Q$

  - це ланцюг.

Ланцюг команд. Правило ланцюга

- Правило ланцюга.
  - Python виконує команди з ланцюга послідовно.

- Приклади ланцюгів:

```
v = i * t

r = s
s = 1

x = x + 1
x = x + 1
```

Рівносильність інструкцій

- Дві інструкції *P*, *Q* будемо називати рівносильними

$P \equiv Q$

якщо вони дають однаковий результат.

  - Результатом інструкції є зміна значення деякої змінної, або/та стану клавіатури (введення), або стану екрану (виведення). Тобто, інструкції можна розглядати як функції на станах виконавця Python

- Властивість ланцюга:

$P \equiv$ `pass` $\equiv P$

`pass` *P*

Лінійні програми

- **Лінійною** називається програма, яка є ланцюгом команд введення, виведення, присвоєння або тотожньої команди.

Правила запису програм

- У програмах у Python можна виділити фізичні та логічні рядки.
- **Фізичний рядок** – це один рядок у інтерпретаторі або у файлі з текстом програми.
- **Логічний рядок** – це один оператор програми з точки зору інтерпретатора.
- Частіше за все у одному фізичному рядку записують один логічний рядок.

- Якщо треба продовжити логічний рядок програми на наступний фізичний рядок, у кінці першого рядка треба поставити \.
- Наприклад:

```
x = (a - b - c) * (c - b + a) * \
(c-2*b) / (a * a + b * b + c * c)
```

- Декілька фізичних рядків об’єднуються у 1 логічний також коли відкриваюча дужка стоїть у першому фізичному рядку, а відповідна їй закриваюча, - у одному з наступних.

Відступи та коментарі

- У Python відступи є важливими.
- Відступ у логічному рядку – це кількість пропусків перед першим символом, який не є пропуском.
- Оператори, які утворюють ланцюг, повинні мати однаковий відступ.

- Коментарі починаються з # або беруться у ''' з обох боків.

```
v = i * t #обчислення швидкості
```

- або

```
''' Це коментар,
який включає багато рядків
'''
```

Приклад лінійної програми

- Обчислення значення поліному

```
y=x**6 – 4*x**4 + 3*x - 7
```

Резюме

- Ми розглянули:
  1. Коротку історію розвитку обчислювальної техніки
  2. Визначення інформатики та інформації
  3. Поняття алгоритму та виконавця
  4. Програми та програмування, компілятори та інтерпретатори
  5. Де завантажити та як запустити Python
  6. Основні команди Python (присвоєння, введення, виведення, тотожню команду)
  7. Ланцюги та лінійні програми
  8. Рівносильність інструкцій
  9. Правила запису програм

Де прочитати

1. A Byte of Python (Russian) Версия 2.01 Swaroop C H (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., http://www.matfiz.univ.kiev.ua/books (також на http://obvintsev.info/compuscience/lectures/index.htm)
3. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
4. Самоучитель Python. http://pythonworld.ru/samouchitel-python
5. С. Шапошникова. Основы программирования на Python. Версия 2 (2011). http://younglinux.info/pdf
6. Python 3.4.3 documentation