Прикладне програмування

ТЕМА 2. ВИКОРИСТАННЯ ОПЕРАЦІЙНОЇ СИСТЕМИ

Операційна система

Коли ми на будь-якому комп’ютері, телефоні або планшеті виконуємо дії над файлом, запускаємо на виконання програму, вводимо команду, - ми працюємо з операційною системою.

Операційна система (operating system) – це комплекс взаємопов'язаних програм, призначених для управління ресурсами комп'ютера та організації взаємодії з користувачем.

Під ресурсами у цьому означенні маються на увазі процесорний час, дискові пристрої, оперативна пам’ять, інші ресурси.

Зазвичай задачі взаємодії з операційною системою вирішуються командами самої системи або стандартними програмами.

Але бувають ситуації, коли стандартних та готових засобів, з тієї чи іншої причини, недостатньо.

Тоді треба писати програми, що використовують операційну систему для розв’язання саме таких задач.

Модулі та пакети для використання операційної системи

У Python, як і у інших мовах програмування, є модулі, що дозволяють використовувати можливості операційної системи.

При цьому слід відмітити, що у Python ці модулі забезпечують кросплатформність, тобто, однакове використання більшості функцій у різних популярних операційних системах: MS Windows, Linux, Mac OS.

Серед модулів та пакетів, що забезпечують використання функцій операційної системи виділимо такі:

- sys
- os
- glob
- shutil

Модулі та пакети для використання операційної системи.2

Насправді, таких модулів набагато більше, але ми навели ті, що найчастіше використовуються та є найбільш загальними.

Модуль sys містить функції, що дозволяють отримати дані або змінити системне оточення інтерпретатора Python.

- Частина цього оточення пов'язана з операційною системою.
- У попередніх темах ми вже зустрічались з sys.path, sys,getsizeof, sys.argv.

Пакет та модуль os містить функції інформування про операційну систему, роботи з файлами та каталогами, управління процесами.

- Пакет os включає декілька модулів, серед яких частіше використовують модуль для роботи з файлами та каталогами os.path.

Модуль glob містить функції роботи з наборами файлів та каталогів.

Модуль shutil (shell utilities – системні програми командного рядка) дозволяє виконати одним викликом масові дії над файлами та каталогами.

Стандартні файли stdin, stdout, stderr

Python під час виконання програм інтерпретатором працює з 3 стандартними файлами: stdin, stdout, stderr.

Ці файли описані у модулі sys.

- sys.stdin – стандартний файл для введення.
- sys.stdout – стандартний файл для виведення.
- sys.stderr – стандартний файл для виведення інформації про помилки.

Стандартні файли Python сам відкриває у момент запуску програми та закриває після її закінчення.

За угодою файл stdin зв’язаний з клавіатурою, а stdout та stderr – з екраном у режимі виведення тексту.

Є можливість перенаправити стандартне введення/виведення у інші файли, тимчасово змінивши значення sys.stdin, sys.stdout або sys.stderr.

Файлова система

Файлова система визначає спосіб організації файлів на зовнішніх носіях даних.

У сучасних операційних системах файлова система на логічному рівні представляє собою ієрархію каталогів у вигляді дерева або ациклічного графу.

У кожному каталозі можуть міститись файли та/або підкаталоги.

Один з каталогів вважається поточним.

Поточний каталог можна змінити.

Файл задається ім’ям файлу (рядком). У кінці імені може бути вказано розширення імені через крапку ‘.’, яке визначає тип файлу.

Наприклад, .txt, .exe, .pdf, .py тощо.

Файлова система.2

Щоб визначити розташування файлу, задають шлях у вигляді послідовності каталогів, починаючи з поточного (відносний шлях) або деякого початкового каталогу (абсолютний шлях).

Каталоги та власне ім’я файлу розділяються символом-розділювачем.

У MS Windows таким символом є обернена коса риска ‘\’, у Unix, – коса риска ‘/’, у MacOS, - двокрапка ‘:’ або коса риска.

Приклади можливого шляху до файлу myfile.txt у різних операційних системах:

- C:\Documents\myfile.txt – шлях у MS Windows
- etc/pub/myfile.txt – шлях у Unix
- Mac HD:Documents:myfile – шлях у Mac OS

Робота з файлами та каталогами. Модулі os та os.path

Типова робота з файлами у файловій системі – це перевірка наявності файлу у деякому каталозі, копіювання (переміщення) файлу з одного каталогу до іншого, видалення файлу, отримання інформації про файл.

Типова робота з каталогами – це отримання списку файлів та підкаталогів каталогу, створення нового каталогу, видалення каталогу.

У таблиці нижче зібрані головні функції модуля os з роботи з каталогами та файлами.

Робота з файлами та каталогами. Модулі os та os.path.2

Функція	Опис
os.listdir(path='.')	Повертає список файлів та підкаталогів у каталозі path
os.walk(top)	Генератор-функція, що повертає послідовність кортежів для всіх підкаталогів дерева, що починається з каталогу top. Для кожного підкаталогу (та для самого top) повертається кортеж (dirpath, dirnames, filenames), де dirpath – каталог, dirnames – список підкаталогів каталогу dirpath, filenames – список файлів каталогу dirpath.
os.getcwd()	Повертає поточний каталог (current working directory)
os.chdir(path)	Робить поточним каталог path
os.mkdir(path)	Створює каталог path
os.rmdir(path)	Видаляє каталог path. Каталог повинен бути порожнім
os.remove(path)	Видаляє файл path

Робота з файлами та каталогами. Модулі os та os.path.3

Модуль os.path містить декілька корисних функцій з отримання інформації щодо файлу та обробки шляхів до каталогів та файлів.

Функція	Опис
os.path.exists(path)	Перевіряє, чи існує каталог (файл) path
os.path.getmtime(path)	Повертає дату та час останньої зміни файлу (каталогу) path
os.path.getctime(path)	Повертає дату та час створення файлу (каталогу) path
os.path.getsize(path)	Повертає розмір файлу path
os.path.isdir(path)	Чи є path каталогом
os.path.isfile(path)	Чи є path файлом
os.path.normpath(path)	Повертає path у «стандартному» вигляді.
os.path.join(path, *paths)	Об’єднує декілька частин шляху та імені файлу paths у єдиний шлях path. При цьому символи-розділювачі у шляху вставляються/змінюються у відповідності до вимог операційної системи.
os.path.split(path)	Розбити шлях на дві частини: каталоги та ім’я файлу
os.path.splitext(path)	Розбити шлях на дві частини: каталоги плюс початок імені файлу та розширення імені файлу, починаючи з ‘.’

Робота з наборами файлів та каталогів

Часто у команді треба одночасно звернутись не до одного, а до декількох файлів, що мають спільне ім’я або його частину.

Таку можливість надає модуль glob.

glob.glob(mask) повертає список файлів та каталогів, що відповідають масці mask. Маска може містити як звичайні, так і універсальні символи.

До універсальних символів відносяться зірочка ‘*’ (означає будь-який підрядок) та знак питання ‘?’ (означає будь-який символ).

Наприклад ‘c?t.*’ відповідають файлам усіх розширень з іменами з 3 літер, перша з яких ‘c’, а остання - ‘t’.

Приклад: Побудова списку каталогів разом з їх розмірами

Є жартівливе твердження про те що жорсткий диск будь-якого об’єму з часом заповнюється на 90%.

Коли це стається, виникає задача чистки диску, тобто видалення з нього тих файлів, які не зовсім потрібні, або зовсім непотрібні.

Доцільно починати цей процес з тих каталогів, які мають найбільший об’єм.

Але операційні системи, як правило, дають інформацію тільки про розмір одного вибраного каталогу.

Тому, якщо каталогів багато, задача стає не зовсім простою.

Приклад: Побудова списку каталогів разом з їх розмірами.2

У даному прикладі побудовано програму, яка повертає список підкаталогів деякого каталогу разом з їх розмірами.

Функція getdirsize повертає розмір одного каталогу, а функція getdirslist – список каталогів та їх розміри.

Список упорядковується за незростанням розміру каталогу.

Тобто, найбільші каталоги будуть на початку списку.

Основна частина програми отримує параметри з командного рядка або вводить їх, якщо параметри не надано.

Версія 2 програми відрізняється від версії 1 тим, що спрямовує стандартне виведення у текстовий файл ‘dirslist.txt’.

Приклад: Збереження файлів з заданих каталогів (backup). Версія 1

Задача збереження файлів з деяких каталогів станом на певну дату та час також є відомою та часто використовується для збереження даних від випадкового знищення а також, можливо, для повернення до попередньої версії даних.

Такі дії називають backup.

Програма у цьому прикладі здійснює backup файлів з заданих каталогів у каталог для backup.

Приклад: Збереження файлів з заданих каталогів (backup). Версія 1.2

Функція copyfile копіює 1 файл з одного каталогу до іншого. Функція copydir рекурсивно копіює каталог та всі його підкаталоги та файли у інший каталог.

Функція getbackupname повертає ім’я нового каталогу, в який будуть збережені файли.

Це ім’я будується як рядок з поточної дати та часу.

Для повернення дати та часу та побудови цього рядка використовується стандартний модуль datetime.

Нарешті, функція backupdirectories зберігає файли з вказаних каталогів у каталозі backup.

Усі функції пропускають файл (каталог), якщо виникає помилка, та виводять повідомлення про помилку.

Основна частина програми отримує параметри з командного рядка або вводить їх, якщо параметри не надано.

Архівування файлів

Архівування файлів – це їх стиснення для подальшого зберігання.

На сьогодні є декілька розповсюджених стандартів стиснення файлів.

Python містить модулі, які забезпечують стиснення файлів у архів та розкриття архівних файлів.

Зокрема, це модулі

- zipfile
- tarfile

які працюють відповідно з архівами у форматах zip та tar.

Ці модулі мають схожі, хоча й не ідентичні функції обробки архівних файлів.

Сама обробка архівних файлів майже не відрізняється від обробки звичайних файлів: архівний файл треба відкрити для читання або запису, читати або писати інформацію з файлу, закрити файл.

Архівування файлів.2

Модуль tarfile дозволяє стискати та розкривати файли у різних форматах: gzip, bzip2, lzma.

Модуль містить такі основні функції:

Функція	Опис
tarfile.open(name= None, mode='r')	Відкрити файл name для подальшого стиснення або розкриття згідно з режимом mode. Повертає об’єкт класу TarFile
tf.add(filename)	Додати файл name до архіву tf
tf.extractall(path=".")	Розкрити всі файли архіву tf у каталог path
tf.close()	Закрити файл tf

Архівування файлів.3

Параметр mode у функції open може набувати таких значень:

mode	значення
'r'	Відкрити архівний файл для читання (розкриття)
'w:gz'	Відкрити файл для запису (стиснення) у форматі gzip
'w:bz2'	Відкрити файл для запису (стиснення) у форматі bzip2
'w:xz'	Відкрити файл для запису (стиснення) у форматі lzma

Приклад: Збереження файлів з заданих каталогів (backup). Версія 2

Необхідно забезпечити збереження даних декількох каталогів (backup).

Регулярне збереження потенційно великих обсягів даних може призвести до вичерпання ресурсу жорсткого диску.

Щоб запобігти цьому, треба архівувати усі попередні версії backup окрім останньої.

Це і робить програма збереження файлів з заданих каталогів 2 версії.

Для архівування каталогів у каталозі backup додались 2 функції: archivesubdirs та removedir.

archivesubdirs виконує архівування підкаталогів заданого каталогу у 1 архівний файл, а removedir, - рекурсивно видаляє заархівований раніше підкаталог та всі його файли.

Масові дії над файлами та каталогами

Масові дії над файлами та каталогами можна виконати, використавши модуль shutil.

Цей модуль містить функції копіювання файлів та каталогів, видалення файлів та каталогів, архівування файлів.

Деякі функції shutil зібрано у таблиці

Масові дії над файлами та каталогами.2

Функція	Опис
shutil.copy(src, dst)	Копіює файл src у файл (або каталог) dst. src – повне ім’я файлу, включаючи шлях до нього. Якщо dst – це каталог, то файл буде скопійовано з тим же ім’ям, що й у src.
shutil.copytree(src, dst)	Копіює каталог src у каталог dst рекурсивно разом з усіма підкаталогами.
shutil.rmtree(path)	Видаляє каталог path рекурсивно разом з усіма підкаталогами. При цьому, каталог може бути непорожнім
shutil.make_archive(base_name, format[, root_dir])	Створює архів з ім’ям base_name у форматі format та додає у нього усі файли з каталогу root_dir. Значенням format може бути один з рядків “zip”, “tar” або “bztar”
shutil.unpack_archive(filename[, extract_dir])	Розкрити всі файли архіву filename у каталог extract_dir

Приклад: Збереження файлів з заданих каталогів (backup). Версія 3

Необхідно забезпечити збереження даних декількох каталогів (backup).

У версії 3 використовуються функції модуля shutil для копіювання каталогу, створення архіву та видалення каталогу.

Це дозволяє суттєво скоротити текст програми у порівнянні з версією 2.

Залишаються тільки функції getbackupname, archivesubdirs, backupdirectories.

Запуск процесів

Процесом у програмуванні називають екземпляр програми, що виконується.

Запуском, синхронізацією та завершенням процесів керує операційна система.

У стандартній бібліотеці Python є декілька модулів для організації запуску процесів та взаємодії з ними.

Ми розглянемо тільки окремі можливості, які надає модуль os.

Запуск процесів.2

Функція	Опис
os.system(command)	Виконує рядок command як команду операційної системи
os.popen(command, mode='r')	Запускає процес command та повертає відкритий об’єкт файлового типу, що відповідає стандартному виведенню процеса command. Якщо параметр mode='w', то повертається файл стандартного введення
os.startfile(path)	Запустити програму, яка пов’язана з файлом path. У операційній системі це рівносильно подвійному натисненню «миші» на піктограмі файлу.

Приклад: Запуск програми збереження файлів з заданих каталогів (backup) за розкладом

Звичайно, написавши програму для збереження файлів з заданих каталогів, ми зацікавлені, щоб вона виконувалась регулярно через задані проміжки часу.

Програма у цьому прикладі здійснює запуск раніше написаної програми для backup через задану кількість годин.

Ця програма використовує модуль роботи з конфігураційними файлами, розглянутий у темі «Регулярні вирази».

У конфігураційному файлі зберігаються параметри, що визначають каталог для backup, каталоги, які треба зберігати, проміжок часу у годинах між збереженням файлів.

Стандартне виведення програми для backup записується у системний журнал.

Програма містить функції backupneeded, що перевіряє, чи вичерпався проміжок часу після останнього збереження файлів, та backup, що запускає збереження у окремому процесі та веде системний журнал.

Стандартна функція sleep(sleeptime) з модуля time призупиняє виконання програми на sleeptime секунд.

Резюме

Ми розглянули:

1. Означення операційної системи.
2. Модулі та пакети для використання операційної системи
3. Стандартні файли stdin, stdout, stderr
4. Файлова система
5. Робота з файлами та каталогами
6. Робота з наборами файлів та каталогів
7. Архівування файлів
8. Масові дії над файлами та каталогами
9. Запуск процесів

Де прочитати

1. Обвінцев О.В. Об’єктно-орієнтоване програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. David Beazley - Python Cookbook, 3rd edition – 2013
3. Mark Pilgrim. - Dive into Python, Version 5.4 - 2004
4. Mark Lutz - Programming Python. 4th Edition - 2011
5. Magnus Lie Hetland - Beginning Python from Novice to Professional, 2nd ed – 2008
6. Jason Brittain, Ian F. Darwin. - Python for Unix and Linux System Administration. - 2008
7. Прохоренок Н.А. - Python 3 и PyQt. Разработка приложений – 2012
8. Hellmann D. - The Python Standard Library by Example – 2011