**Python\Django додаткові питання**

1. **Python -** інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.
2. **Звідки Python -** Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів.
3. **Парадигми програмування** - об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.
4. Переваги :

* чистий [синтаксис](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81) (для виділення блоків слід використовувати відступи);
* переносність програм (що властиве більшості інтерпретованих мов);
* стандартний [дистрибутив](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%B8%D0%B2) має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
* можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
* стандартний дистрибутив має просте, але разом із тим досить потужне [середовище розробки](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8), яке зветься IDLE і яке написане мовою Python;
* зручний для розв'язання математичних проблем (має засоби роботи з [комплексними числами](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81%D0%BD%D1%96_%D1%87%D0%B8%D1%81%D0%BB%D0%B0), може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний [калькулятор](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D1%8C%D0%BA%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80));
* відкритий код (можливість редагувати його іншими користувачами).

1. **Python має ефективні структури даних** високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.
2. **Інтерпретатор** мови Python може бути розширений функціями та типами даних, розробленими на [C](https://uk.wikipedia.org/wiki/C_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) чи [C＋＋](https://uk.wikipedia.org/wiki/%D0%A1%D1%96-%D0%BF%D0%BB%D1%8E%D1%81-%D0%BF%D0%BB%D1%8E%D1%81) (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.
3. **Вплив інших мов**

* [ABC](https://uk.wikipedia.org/w/index.php?title=ABC_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)&action=edit&redlink=1)— відступи (поля) для групування операторів, високорівневі структури даних (map) (фактично, Python створювався як спроба виправити помилки, допущені при проєктуванні ABC);
* [Modula-3](https://uk.wikipedia.org/w/index.php?title=Modula-3&action=edit&redlink=1)— пакети, модулі, використання else спільно з try та except, іменовані аргументи функцій (на це також вплинув [Common Lisp](https://uk.wikipedia.org/wiki/Common_Lisp));
* [С](https://uk.wikipedia.org/wiki/%D0%A1%D1%96_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C++](https://uk.wikipedia.org/wiki/C%2B%2B) — деякі синтаксичні конструкції (як пише сам [Гвідо ван Россум](https://uk.wikipedia.org/wiki/%D0%93%D0%B2%D1%96%D0%B4%D0%BE_%D0%B2%D0%B0%D0%BD_%D0%A0%D0%BE%D1%81%D1%81%D1%83%D0%BC) — він використовував найбільш несуперечливі конструкції з С, щоб не викликати неприязнь у Сі-програмістів до Python[[15]](https://uk.wikipedia.org/wiki/Python" \l "cite_note-foreword-15));
* [Smalltalk](https://uk.wikipedia.org/wiki/Smalltalk) — об'єктно-орієнтоване програмування;
* [Lisp](https://uk.wikipedia.org/wiki/Lisp) — окремі риси [функціонального програмування](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (lambda, map, reduce, filter та інші);
* [Fortran](https://uk.wikipedia.org/wiki/Fortran) — зрізи масивів, комплексна арифметика;
* [Miranda](https://uk.wikipedia.org/w/index.php?title=%D0%9C%D1%96%D1%80%D0%B0%D0%BD%D0%B4%D0%B0_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)&action=edit&redlink=1) — спискові вирази;
* [Java](https://uk.wikipedia.org/wiki/Java) — модулі logging, unittest, threading (частина можливостей оригінального модуля не реалізована), xml.sax стандартної бібліотеки, спільне використання finally та except при обробці винятків, використання @ для декораторів;
* [Icon](https://uk.wikipedia.org/w/index.php?title=Icon_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Icon_(programming_language)) — [генератори](https://uk.wikipedia.org/wiki/Python#%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%B8).

1. **Типи й структури даних** Python підтримує [динамічну типізацію](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%97), тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і [комплексних чисел](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81%D0%BD%D1%96_%D1%87%D0%B8%D1%81%D0%BB%D0%B0). Python має багату бібліотеку для роботи з [рядками](https://uk.wikipedia.org/wiki/%D0%A0%D1%8F%D0%B4%D0%BE%D0%BA_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)), зокрема, кодованими в [юнікоді](https://uk.wikipedia.org/wiki/%D0%AE%D0%BD%D1%96%D0%BA%D0%BE%D0%B4).
2. **Класи -** Система класів підтримує [множинне успадкування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B8%D0%BD%D0%BD%D0%B5_%D1%83%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) і [метапрограмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Будь-який тип, включаючи базові, входить до системи [класів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), й за необхідності можливе успадкування навіть від базових типів.
3. **Інтерактивний режим** Подібно [Ліспу](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%81%D0%BF) та [Прологу](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BB%D0%BE%D0%B3_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) в режимі відлагодження, інтерпретатор Python має інтерактивний режим роботи, при якому введені з клавіатури вирази відразу ж виконуються, а результат виводиться на екран. Цей режим цікавий не тільки новачкам, але й досвідченим програмістам, які можуть протестувати в інтерактивному режимі будь-який фрагмент коду, перш ніж використовувати його в основній програмі, або просто використовувати як [калькулятор](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D1%8C%D0%BA%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80) з великим набором функцій.
4. **Об'єктно-орієнтоване програмування** Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація [ООП](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) в Python є елегантною, потужною та добре продуманою, але разом з тим, достатньо специфічною в порівнянні з іншими [об'єктно-орієнтованими мовами](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F).
5. **Можливості та особливості ООП :**

* Класи є одночасно об'єктами з усіма нижче наведеними можливостями.
* Успадкування, в тому числі множинне.
* Поліморфізм (всі функції віртуальні).
* Інкапсуляція (два рівні — загальнодоступні та приховані методи і поля). Особливість — приховані члени доступні для використання та помічені як приховані лише особливими іменами.
* Спеціальні методи, що керують життєвим циклом об'єкта: конструктори, деструктори, розподільники пам'яті.
* Перевантаження операторів (усіх, крім is, '.', '=' і символьних логічних).
* Властивості (імітація поля за допомогою функцій).
* Управління доступу до полів (емуляція полів і методів, частковий доступ тощо).
* Методи для управління найпоширенішими операціями (істиннісне значення, len(), глибоке копіювання, [серіалізація](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%96%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F), ітерація по об'єкту, …)
* Метапрограмування (управління створенням класів, тригери на створення класів, та ін)
* Повна [інтроспекція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D1%96%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)).
* Класові та статичні методи, класові поля.
* Класи, вкладені у функції та інші класи.

1. **парадигма**[**функціонального програмування**](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)**, зокрема**

* функція є об'єктом;
* функції вищих порядків;
* рекурсія;
* розвинена обробка списків (спискові вирази, операції над послідовностями, ітератори);
* аналог замикань (closures);
* часткове застосування функції;
* можливість реалізації інших засобів на самій мові (наприклад, [каррінг](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D1%80%D1%96%D0%BD%D0%B3)).

1. **Модулі та пакети** [Програмне забезпечення](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) ([застосунок](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) або [бібліотека](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))) на Python оформлюється у вигляді [модулів](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), які у свою чергу можуть бути зібрані в *пакунки*. Модулі можуть розташовуватися як у [каталогах](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3_(%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)), так і в [ZIP-архівах](https://uk.wikipedia.org/wiki/ZIP). Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування.
2. **Інтроспекція** Python підтримує повну [інтроспекцію](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D1%96%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) часу виконання. Це означає, що для будь-якого об'єкта можна отримати всю інформацію про його внутрішню структуру.
3. **Застосування інтроспекції** ([метапрограмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) є важливою частиною того, що називають «pythonic style», і широко застосовується в бібліотеках і [фреймворках](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) Python, таких як [PyRO](https://uk.wikipedia.org/w/index.php?title=Python_Remote_Objects&action=edit&redlink=1" \o "Python Remote Objects (ще не написана)), [Pyro](https://uk.wikipedia.org/w/index.php?title=Pyro&action=edit&redlink=1" \o "Pyro (ще не написана)), [PLY](https://uk.wikipedia.org/wiki/PLY), [CherryPy](https://uk.wikipedia.org/wiki/CherryPy), [Django](https://uk.wikipedia.org/wiki/Django) та інших, заощаджуючи час програміста, що ними користується.
4. **Обробка винятків** Для того, щоб в обробнику винятка виконати певні дії, а потім передати виняток далі, на один рівень обробників вище (тобто, викинути той самий виняток ще раз), використовується інструкція raise без параметрів:
5. **Ітератори** У програмах на Python широко використовуються [ітератори](https://uk.wikipedia.org/wiki/%D0%86%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80). Цикл for може працювати як з послідовністю, так і з ітераторами. Усі колекції, як правило, надають ітератор. Об'єкти визначеного користувачем класу теж можуть бути ітераторами. Модуль itertools стандартної бібліотеки містить багато корисних функцій для роботи з ітераторами. На відміну від звичайних послідовностей, всі елементи яких зберігаються в пам'яті, отримання наступного елемента забезпечує генератор — спеціальна функція, звернення до якої обчислює і повертає наступний елемент генератора.
6. **Генератори** Однією з цікавих можливостей мови є **генератори** — функції, що між викликами зберігають внутрішній стан: значення локальних змінних і поточну інструкцію (див. також: [співпрограма](https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D1%96%D0%B2%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0)). Генератори можуть використовуватися як [ітератори](https://uk.wikipedia.org/wiki/%D0%86%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80) для структур даних і для [лінивих обчислень](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D0%BD%D0%B8%D0%B2%D1%96_%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F).
7. **генераторні вирази** — вирази, що дають у результаті генератор. Генераторні вирази дозволяють заощадити пам'ять там, де інакше потрібно було б використовувати список із проміжними результатами:
8. Починаючи з версії 2.5, Python підтримує **повноцінні співпроцедури**: тепер в генератор можна передавати значення за допомогою методу send() та збуджувати в його контексті виняток за допомогою методу throw().
9. **Керування контекстом виконання** Оператор може застосовуватися в тих випадках, коли до та після деяких дій обов'язково мають виконуватися якісь інші дії, незалежно від створених у блоці винятків або операторів return: файли має бути закрито, ресурси звільнено, перенаправлення стандартного введення/виведення завершено, тощо. Оператор полегшує читання коду, отже, допомагає уникати помилок.
10. **Декоратори** Починаючи з версії 2.4, Python дозволяє використовувати, так звані, *Декоратори* (не слід плутати з однойменним [шаблоном проєктування](https://uk.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8_%D0%BF%D1%80%D0%BE%D1%94%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F)) для підтримки існуючої практики перетворення функцій та методів у місці визначення (декораторів може бути декілька). Після довгих дебатів для декораторів став використовуватися символ @ у рядках, що передують визначенню функції або методу.
11. У Python є **ще кілька можливостей**, що відрізняють його від багатьох інших мов високою гнучкістю та динамічністю.

Наприклад, клас є об'єктом, а в операторі визначення класу можна використовувати вирази в списку батьківських класів.

1. **Стандартна бібліотека** Багата стандартна [бібліотека](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) є однією з привабливостей мови Python. Тут є засоби для роботи з багатьма [мережевими протоколами](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%80%D0%B5%D0%B6%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) та форматами [Інтернету](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), наприклад, модулі для написання [HTTP-серверів](https://uk.wikipedia.org/wiki/HTTP-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) та клієнтів, для розбору та створення поштових повідомлень, для роботи з [XML](https://uk.wikipedia.org/wiki/XML), тощо. Набір модулів для роботи з [операційною системою](https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) дозволяє писати крос-платформні [застосунки](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BA%D0%B8). Існують модулі для роботи з [регулярними виразами](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D0%B8%D0%B9_%D0%B2%D0%B8%D1%80%D0%B0%D0%B7), текстовими [кодуваннями](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D1%81%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D1%96%D0%B2), [мультимедійними](https://uk.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D1%96%D0%B0) форматами, [криптографічними](https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F) протоколами, архівами, [серіалізацією](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%96%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F) даних, [юніт-тестуванням](https://uk.wikipedia.org/wiki/%D0%AE%D0%BD%D1%96%D1%82-%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) та ін.
2. **Модулі розширення та програмні інтерфейси** Крім стандартної бібліотеки існує багато інших, що надають інтерфейс до всіх системних викликів на різних платформах; зокрема, на платформі [Win32](https://uk.wikipedia.org/wiki/Win32) підтримуються всі виклики [Win32 API](https://uk.wikipedia.org/wiki/Windows_API), а також [COM](https://uk.wikipedia.org/wiki/Component_Object_Model) в обсязі не меншому, ніж у [Visual Basic](https://uk.wikipedia.org/wiki/Visual_Basic) або [Delphi](https://uk.wikipedia.org/wiki/Delphi_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Існує велика кількість прикладних бібліотек для Python у різноманітних галузях: [веб-розробка](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), [бази даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), обробка зображень, обробка тексту, [чисельні методи](https://uk.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%B5%D0%BB%D1%8C%D0%BD%D1%96_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B8), програми [операційної системи](https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) тощо.
3. **Бібліотека [NumPy](https://uk.wikipedia.org/wiki/NumPy)** для роботи з багатовимірними [масивами](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D0%B8%D0%B2_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)) дозволяє досягти продуктивності наукових розрахунків, порівнянної зі спеціалізованими пакетами. [SciPy](https://uk.wikipedia.org/wiki/SciPy) використовує NumPy і надає доступ до великого спектра математичних алгоритмів (матрична алгебра — [BLAS](https://uk.wikipedia.org/wiki/BLAS), level 1-3 і [LAPACK](https://uk.wikipedia.org/wiki/LAPACK); [ШПФ](https://uk.wikipedia.org/wiki/%D0%A8%D0%9F%D0%A4)).
4. **Бібліотека**[**WSGI**](https://uk.wikipedia.org/wiki/WSGI) — інтерфейс шлюзу з веб-сервером (Python Web Server Gateway Interface).
5. **Вбудовуванні інтерпретатори** Python у застосунки. Python легко вбудовується в програми на [Java](https://uk.wikipedia.org/wiki/Java), [C](https://uk.wikipedia.org/wiki/C_(%D0%BC%D0%BE%D0%B2%D0%B0))/[C++](https://uk.wikipedia.org/wiki/C%2B%2B), [Ocaml](https://uk.wikipedia.org/wiki/Ocaml). Взаємодія Python-застосунків з іншими системами можлива також за допомогою [CORBA](https://uk.wikipedia.org/wiki/CORBA), [XML-RPC](https://uk.wikipedia.org/wiki/XML-RPC), [SOAP](https://uk.wikipedia.org/wiki/SOAP), [COM](https://uk.wikipedia.org/wiki/Component_Object_Model).
6. **Графічні бібліотеки** З Python поставляється бібліотека [tkinter](https://uk.wikipedia.org/wiki/Tkinter) на основі [Tcl](https://uk.wikipedia.org/wiki/Tcl)/[Tk](https://uk.wikipedia.org/wiki/Tk) для створення крос-платформних програм з [графічним інтерфейсом](https://uk.wikipedia.org/wiki/GUI).

Для науково-технічної мети найбільшого поширення набуло використання [matplotlib](https://uk.wikipedia.org/wiki/Matplotlib) — бібліотеки з інтерфейсом, аналогічним MATLAB Plot Tool.

Існують розширення, що дозволяють використовувати всі основні GUI бібліотеки — [wxPython](https://uk.wikipedia.org/wiki/WxPython), засноване на бібліотеці [wxWidgets](https://uk.wikipedia.org/wiki/WxWidgets), [PyGTK](https://uk.wikipedia.org/wiki/PyGTK) для [GTK+](https://uk.wikipedia.org/wiki/GTK%2B), [PyQt](https://uk.wikipedia.org/wiki/PyQt) та [PySide](https://uk.wikipedia.org/wiki/PySide) для [Qt](https://uk.wikipedia.org/wiki/Qt) та інші. Деякі з них також надають широкі можливості для роботи з базами даних, графікою та мережами, використовуючи всі можливості бібліотеки, на якій базуються.

1. **Бібліотека [Pygame](https://uk.wikipedia.org/wiki/Pygame)**. Вона також надає великі засоби роботи з [мультимедіа](https://uk.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D1%96%D0%B0): з її допомогою можна керувати звуком і зображеннями, відтворювати відео. Надаване pygame апаратне прискорення графіки [OpenGL](https://uk.wikipedia.org/wiki/OpenGL) має більш високорівневий інтерфейс в порівнянні з PyOpenGL, що копіює семантику С-бібліотеки для OpenGL. Є також PyOgr, що забезпечує прив'язку до [OGRE](https://uk.wikipedia.org/wiki/OGRE) — високорівневої об'єктно-орієнтованої бібліотеки 3D-графіки. Крім того, існує бібліотека pythonOCC , що забезпечує прив'язку до середовища 3D-моделювання та симуляції OpenCascade.
2. **Порівняння з іншими мовами** Найчастіше Python порівнюють з [Perl](https://uk.wikipedia.org/wiki/Perl) та [Ruby](https://uk.wikipedia.org/wiki/Ruby). Ці мови також є інтерпретованими та мають приблизно однакову швидкість виконання програм. Як і Perl, Python може успішно застосовуватися для написання скриптів (сценаріїв). Як і Ruby, Python є добре продуманою системою для [ООП](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F).

Засоби [функціонального програмування](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) частково запозичені з [Scheme](https://uk.wikipedia.org/wiki/Scheme) та [Icon](https://uk.wikipedia.org/w/index.php?title=Icon_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)&action=edit&redlink=1" \o "Icon (мова програмування) (ще не написана)).

У середовищі комерційних застосунків швидкість виконання програм на Python можуть порівнювати з [Java](https://uk.wikipedia.org/wiki/Java)-[застосунками](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA).

Попри те, що Python має досить самобутній синтаксис, одним із принципів дизайну цієї мови є [принцип найменшого подиву](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%BD%D0%B0%D0%B9%D0%BC%D0%B5%D0%BD%D1%88%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%B4%D0%B8%D0%B2%D1%83).

1. **Низька швидкодія** Python, як і багато інших [інтерпретованих мов](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), які не застосовують, наприклад, [JIT](https://uk.wikipedia.org/wiki/JIT-%D0%BA%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%86%D1%96%D1%8F)-компілятори, мають загальний недолік — порівняно невисоку швидкість виконання програм.[[41]](https://uk.wikipedia.org/wiki/Python#cite_note-alioth-benchmark-41) Однак, у випадку з Python цей недолік компенсується зменшенням часу розробки програми.[[41]](https://uk.wikipedia.org/wiki/Python#cite_note-alioth-benchmark-41) У середньому, програма, написана на Python, в 2-4 рази компактніша, ніж її аналог на [C++](https://uk.wikipedia.org/wiki/C%2B%2B) або [Java](https://uk.wikipedia.org/wiki/Java).[[41]](https://uk.wikipedia.org/wiki/Python#cite_note-alioth-benchmark-41) Збереження [байт-коду](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) (файли .pyc і .pyo) дозволяє інтерпретатору не витрачати зайвий час на перекомпіляцію коду модулів при кожному запуску, на відміну, наприклад, від мови [Perl](https://uk.wikipedia.org/wiki/Perl). Крім того, існує спеціальна JIT-бібліотека psyco[[42]](https://uk.wikipedia.org/wiki/Python" \l "cite_note-42) (проте призводить до збільшення споживання оперативної пам'яті). Ефективність psyco в значній мірі залежить від архітектури програми.
2. [**Компілятор заднього плану**](https://uk.wikipedia.org/w/index.php?title=Back-end&action=edit&redlink=1). Прикладами таких реалізацій може служити PyPy, що базується на LLVM; більш ранньою ініціативою є проєкт [Parrot](https://uk.wikipedia.org/w/index.php?title=Parrot&action=edit&redlink=1" \o "Parrot (ще не написана))[[en]](https://en.wikipedia.org/wiki/Parrot_virtual_machine). Очікується, що використання ВМ типу LLVM призведе до тих самих результатів, що й використання аналогічних підходів для реалізацій мови Java, де низька обчислювальна продуктивність в основному подолана.
3. **Відсутність статичної типізації** Відсутність статичної типізації є не стільки вадою інтерпретатора, скільки вибором розробника мови. Справа в тому, що в Python прийнята так звана [«качина типізація»](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%87%D0%B8%D0%BD%D0%B0_%D1%82%D0%B8%D0%BF%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F). У силу цього типи переданих значень недоступні на етапі компіляції, та помилки на зразок AttributeError можуть виникати під час виконання. Відсутність статичної типізації також є однією з основних причин низької швидкодії.
4. **Відсутність статичної типізаці**ї і деякі інші причини не дозволяють реалізувати в Python механізм [перевантаження функцій](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B2%D0%B0%D0%BD%D1%82%D0%B0%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97) на етапі компіляції. Можливості Python дозволяють реалізувати динамічну перевантаження на етапі виконання, що, звичайно, уповільнює виклик, так як вирішення яку саме функцію викликати проводиться при кожному зверненні і є, в загальному випадку, досить складною процедурою. Відсутність перевантаження в Python намагаються компенсувати використанням віртуальних функцій.
5. **Неможливість модифікації вбудованих класів** У порівнянні з [Ruby](https://uk.wikipedia.org/wiki/Ruby) та деякими іншими мовами, в Python відсутня можливість модифікувати вбудовані класи, такі, як int, str, float, list та інші, що, однак, дозволяє Python споживати менше оперативної пам'яті і швидше працювати. Ще однією причиною введення такого обмеження є необхідність узгодження з модулями розширення. Багато модулів (з метою оптимізації швидкодії) перетворять Python-об'єкти елементарних типів до відповідних Сі-типів замість маніпуляцій з ними за допомогою Сі-API.
6. **Глобальне блокування інтерпретатора (GIL)** (Global Interpreter Lock) — проблема, притаманна [CPython](https://uk.wikipedia.org/wiki/CPython), [Stackless](https://uk.wikipedia.org/wiki/Stackless_Python) та [PyPy](https://uk.wikipedia.org/wiki/PyPy), але відсутня в [Jython](https://uk.wikipedia.org/wiki/Jython) та [IronPython](https://uk.wikipedia.org/wiki/IronPython). При своїй роботі основний інтерпретатор Python постійно використовує велику кількість нитко-небезпечних даних. В основному це словники, в яких зберігаються атрибути об'єктів. Для уникнення руйнування цих даних при спільній модифікації з різних [потоків](https://uk.wikipedia.org/wiki/%D0%9D%D0%B8%D1%82%D1%8C) перед початком виконання декількох інструкцій (за замовчуванням 100) потік інтерпретатора захоплює GIL, а після закінчення звільняє.
7. **Варіанти вирішення проблеми GIL**. Перший — відмова від спільного використання змінюваних даних. При цьому дані дублюються в нитях і необхідність забезпечення їхньої синхронізації (якщо така потрібна) лягає на програміста.[[56]](https://uk.wikipedia.org/wiki/Python#cite_note-56) Цей підхід веде до збільшення споживання оперативної пам'яті (однак не настільки сильно, як при використанні [процесів](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))).

Другий підхід — забезпечення більш гранульованої синхронізації — для окремих структур даних. У цьому випадку падає продуктивність внаслідок збільшення числа звільнень/захоплень блокувань.

1. **Реалізації** на кожній платформі Python підтримує характерні для даної платформи технології (наприклад, [Microsoft](https://uk.wikipedia.org/wiki/Microsoft) [COM](https://uk.wikipedia.org/wiki/Component_Object_Model)). Крім того, існує спеціальна версія Python для віртуальної машини [Java](https://uk.wikipedia.org/wiki/Java) — [Jython](https://uk.wikipedia.org/wiki/Jython), що дозволяє інтерпретатору виконуватися на будь-якій системі, що підтримує Java, класи Java можуть безпосередньо використовуватися з Python і навіть бути написаними на Python. Нещодавно почалася розробка системи, спрямованої на повнішу інтеграцію з платформою [.NET](https://uk.wikipedia.org/wiki/.NET) — [Iron Python](https://uk.wikipedia.org/wiki/IronPython).
2. **Подальша розробка** [Python Enhancement Proposal](https://uk.wikipedia.org/w/index.php?title=Python_Enhancement_Proposal&action=edit&redlink=1) («PEP») — це документ зі стандартизованим дизайном, що надає загальну інформацію про мову Python, включаючи нові пропозиції, описи та роз'яснення можливостей мови. PEP пропонуються як основне джерело для пропозиції нових можливостей і для роз'яснення вибору того або іншого дизайну для основних елементів мови. Видатні PEP рецензуються і коментуються [BDFL](https://uk.wikipedia.org/w/index.php?title=BDFL&action=edit&redlink=1).
3. **Графік і сумісність** Серії Python 2.x і Python 3.x протягом кількох випусків будуть існувати паралельно, при цьому серія 2.x буде використовуватися для забезпечення сумісності та швидше за все в неї будуть включені деякі можливості серії 3.x. PEP 3000 містить більше інформації про плановані випуски.
4. **Python 3.0** зворотньо не сумісний з попередньою серією 2.x. Код Python 2.x швидше за все буде видавати помилки при виконанні в Python 3.0. Динамічна типізація Python, разом з планами зміни декількох методів словників, робить механічний переклад з Python 2.x в Python 3.0 дуже складним. Однак, утиліта «2to3» вже здатна зробити більшість роботи з перекладу коду, вказуючи на підозрілі їй частини за допомогою коментарів і попереджень. PEP 3000 рекомендує тримати вихідний код для серії 2.x, і робити випуски для Python 3.x за допомогою «2to3». Отриманий код не слід редагувати, поки програма повинна бути працездатною в Python 2.x.
5. **Застосування** Python — стабільна та поширена мова. Вона використовується в багатьох проєктах та в різних якостях: як основна мова програмування або для створення розширень та інтеграції додатків. На Python реалізована велика кількість проєктів, також вона активно використовується для створення прототипів майбутніх програм.
6. **Питон чувствителен к регистру?** Язык считается чувствительным к регистру в случае, если он различает имена "myname" и "Myname". То есть, если он отслеживает разницу регистра (между верхним и нижним).
7. **Для чего нужен pass (pass statement) в питоне**? чтобы код не давал никакого результата и не показывал ошибку, например, если еще не готово, но нужно иметь синтаксический корректный код. Можно поставить pass:
8. **Функция help()в питоне.** Функция help() показывает строку документации и справку для ее аргумента.
9. **Функция dir() в питоне.** Функция dir() возвращает список, содержащий пространство имен в объекте.
10. **Как пишутся комментарии в питоне?** Для этого используется символ #. Все, что написано на строке после него, считается комментарием и игнорируется. Комментарии используются для объяснения цели написанного кода. Многострочных комментариев в прямом смысле слова в питоне нет.
11. **Как проверить, что все символы строки относятся к алфавитно-цифровым?** Для этого используется метод isalnum().
12. **Как в питоне узнать, в какой мы сейчас директории?** Для этого используется функция getcwd(). Она импортируется из модуля os:
13. **Что такое блок?** Когда мы пишем предложение (statement), нам нужно завершить первую строку двоеточием, а под ним написать блок кода, который исполняется в рамках этого предложения. Каждая строка блока пишется с одинаковым отступом.
14. **Зачем нужны break и continue?** Они используются для управления последовательностью операций: break останавливает исполнение цикла и переводит исполнение на следующий блок кода, continue как бы перепрыгивает на следующую итерацию цикла и не прекращает его исполнение.
15. **Если мы не поставим двоеточие в конце строки для цикла "do-while", он все равно сработает?** В питоне такой цикл не реализован. Это вопрос из тех, которые с подвохом, когда упоминают элементы других языков.
16. **Арифметический оператор //** Оператор // выполняет целочисленное деление и возвращает целую часть числа, полученного в результате операции:
17. **Арифметический оператор** \*\* - Оператор \*\* возводит в степень
18. **Арифметический Оператор %** возвращает результат деления по модулю, то есть остаток после деления.
19. **операторы сравнения в питоне?**Данные операторы сравнивают значения между собой. Оператор "меньше" (<): если значение с левой стороны от оператора меньше, он возвращает True:
20. **Оператор "больше" (>):** если значение с левой стороны от оператора больше, он возвращает True.
21. **Оператор "меньше или равно" (<=):** если значение с левой стороны от оператора меньше значения с правой стороны или равно ему, он возвращает True.
22. **Оператор равенства (==):** если значения равны, он возвращает True:
23. **логические операторы в питоне** Всего их три: and, or, not.
24. **Что такое оператор принадлежности?** Это операторы in и not in. Они показывают, является ли одно значение частью другого.
25. **Расскажите про операторы тождественности.** Операторы is и is not показывают, являются ли два значения идентичными.
26. **Какие типы данных поддерживаются в питоне?** В питоне используется пять типов данных:  
    - числа, которые содержат числовые значения;  
    - строки, который представляют собой последовательность символов; обозначаются одинарными или двойными кавычками.  
    - списки, который представляют собой коллекцию значений; обозначаются квадратными скобками.  
    - кортежи, которые похожи на списки, но отличаются тем, что не могут быть изменены.  
    - словари, которые содержат пары "ключ: значение"; обозначаются фигурными скобками.
27. **Что такое строка документации (docstring)?** Она вносится первой строкой в блок, определяющий содержание функции, класса или метода. Содержит описание их цели и способа исполнения. Обозначается тремя одинарными или двойными кавычками с каждой стороны.
28. **Как можно принять результат ввода на клавиатуре?** Если пользователь что-то вводит с помощью клавиатуры, можно использовать функцию input(). В качестве аргумента можно задать данной функции текст запроса на ввод. Результат ввода всегда является строкой.
29. **Что такое функция?** Когда мы хотим исполнить определенную последовательность (sequence of statements), мы можем дать ей имя. Например, определим функцию, которая принимает два числа и возвращает то, которое больше.
30. **Что такое рекурсия** Это когда функция вызывает саму себя. При этом она должна иметь базовое условие, чтобы не создать бесконечный цикл.
31. **Что такое оператор контроля последовательности (control flow statement)?** Обычно программа в питоне начинает исполнение с первой строки. После нее программа однократно исполняет каждое предложение. Когда будет исполнено последнее предложение, программа прекращается. Также контроль последовательности помогает усложнить обычный порядок исполнения программы.
32. **Как работать с числами, которые не входят в десятичную систему счисления?** В питоне можно вводить бинарные, восьмеричные и шестнадцатеричные числа. Бинарные. Это числа, составленные из 0 и 1. Для ввода в бинарном формате, используется префикс 0b или 0B:
33. **Чем Python отличается от Java?** Если сравнивать Python и Java:  
    - Java быстрее  
    - Python использует отступы, а Java нужны скобки  
    - в Python динамическая типизация, а в Java — статическая  
    - Python — простой и лаконичный, а Java — многословный язык  
    - Python — интерпретируемый язык  
    - Java не зависит от используемой платформы  
    - в Java есть интерфейс JDBC, который улучшает доступ к базам данных
34. **Как выйти из бесконечного цикла?** Можно нажать комбинацию клавиш Ctrl+C, которая прерывает исполнение.
35. **механизм передачи параметров.** В питоне используется передача параметров по ссылке. Если изменить параметр внутри функции, то это отразится на выводе функции. Однако, если использовать в качестве параметров литералы (строки, числа, кортежа), то они передаются по значению (потому что они не изменяемые).
36. **Что такое with в питоне?** Данная инструкция обеспечивает исполнение кода очистки после исполнения программы. Например, можно использовать ее для открытия файла, совершить с ним какие-то действия и автоматически закрыть файл после завершения работы. Аналогичным образом можно открывать соединение с базой данных и автоматически его закрывать. Код очистки исполняется даже в случае, когда появляется исключение (exception).
37. **Чем файл .pyc отличается от .py?** Оба файла содержат байткод, но .pyc является компилированной версией файла питона. Его байткод не зависит от платформы, поэтому он исполняется на всех платформах, которые поддерживают формат .pyc.
38. **Что делает питон объектно-ориентированным?** Он следует парадигме объектно-ориентированного программирования, которая построена вокруг классов (classes) и их экземпляров (instances). Это позволяет реализовать следующие функции:  
    - сокрытие внутренней структуры данных  
    - абстракция  
    - наследование  
    - полиморфизм (способность выбирать правильный метод в зависимости от типа данных)  
    - ограничение доступа к данным
39. **Какие есть типы объектов в питоне?** В питоне поддерживаются изменяемые (mutable) и не изменяемые (immutable) типы объектов. Не изменяемые не позволяют изменять свое содержание. Примеры: кортежи, булевы, строки, числа. Итерация по ним выполняется быстрее. Изменяемые позволяют изменять свое содержание. Примеры: списки, множества и словари. Итерация по ним выполняется медленнее.
40. Тернарный оператор – это оператор, который используется для демонстрации какого-либо условия, то есть вместо условной конструкции. Он состоит из истинного или ложного значений и      выражения, которое должно быть вычислено для этих значений.
41. **Как строка может быть преобразована в число** Для того чтобы преобразовать строку в число можно использовать встроенные функции языка Python, например, с помощью int() конструктора. Таким способом можно преобразовывать данные типа: int ('1') == 1. Функция float() также может быть использована для того, чтобы показать число в формате float(‘1’)=1.
42. **Делегирование –** это объектно-ориентированный метод, также называемый паттерном проектирования. Давайте представим, что у вас есть объект x, и вы хотите изменить поведение только одного из его методов. Вы можете создать новый класс, в котором этот метод будет переопределен, но вы заинтересованы в изменении и делегировании всех остальных методов, соответствующему методу x.
43. **Для чего нужен “self” в функциях** “Self” – это переменная, обозначающая принадлежность объекта самому себе. В большинстве объектно-ориентированных языков, данная переменная передается в качестве скрытого параметра методов, которые определяются объектами. Но в Python это определение и передача осуществляется явно. В первом аргументе, который создается в экземпляре класса A, происходит автоматическая передача параметров метода. Происходит обращение к отдельным экземплярам переменной для каждого отдельного объекта. Это первый аргумент, который используется в экземпляре класса, а метод «self» определен явно для всех используемых и существующих методов. Название переменных происходит вот по такому образцу: “self.xxx”.
44. **Как в методе можно явно определить “self” “Self”** – это ссылочная переменная и атрибут экземпляра, который используется вместо локальной переменной внутри класса. Функция или переменная self, такая как self.x или self.meth(), могут использоваться в том случае, если класс неизвестен. У такого класса нет переменных, объявленных как локальные. У него отсутствует какой-либо синтаксис, но в нем можно производить явную передачу ссылки или вызвать метод, который используется этим классом. Использование writebaseclass.methodname(self, <argument list>) показывает, что метод \_init\_() может быть расширен до базовых методов класса. Это также решает синтаксическую проблему, которая возникала между оператором присваивания и локальными переменными. Такой подход указывает интерпретатору, когда будут использоваться переменные экземпляра класса, а когда локальные переменные. Явное использование переменной self.var решает вышеупомянутую проблему.
45. Почему метод join() чаще используется при работе со строками, а не со списками и кортежами? Для обеспечения полного функционала всех методов и функций в string используется string модуль. Представленный string модуль использует метод join(): **", ".join(['1', '2', '4', '8', '16'])  #результат выглядит так: "1, 2, 4, 8, 16"**
46. **Метод join()** – это строковый метод, который предоставляет разделительную строку, чтобы можно было использовать функцию над последовательностью строки, то есть для того, чтобы можно было вставлять необходимую строку в нужное место. Метод может принимать любое количество аргументов, которые соответствуют некоторым требованиям, которые устанавливаются для объектов последовательности, а они уже определяются внутри своего класса. Join() используется в работе string модуля для объединения символов строки в одну строку.
47. **Компиляция и использование ссылок (связывание)** позволяет новым расширениям корректно компилироваться без каких-либо ошибок, а процесс связывания может быть совершен только тогда, когда процедура компиляции проходит успешно. При использовании динамической загрузки, она определяется стилем операционной системы. Python интерпретатор может быть использован для обеспечения динамической загрузки установочных файлов конфигурации. В дальнейшем это действие приведет в пересборке интерпретатора.
48. **Процесс извлечения значений из объекта в Python.** Для того извлечения значения необходимо чтобы был известен тип объекта, и только после этого можно будет получить его значения.
49. **Опишите свою последовательность действий по созданию скрипта, выполняемого на Unix.**

Для создания скрипта, который будет выполняться на Unix, необходимо выполнить следующие шаги:

* Вначале создайте файл скрипта и напишите код, который хотите, чтобы выполнялся в Unix;
* Делаем из этого файла выполняемый файл, путем установки специального режима с помощью добавления к первой строке #! данную строку считывает Python-интерпретатор;
* Устанавливаем необходимые полномочия файлу, используя команду chmod +x file. Файл сперва будет считывать самую главную строчку:
* #!/usr/local/bin/python
* Эта строка указывает путь, который передается интерпретатору Python, и он не зависит от программной среды;
* Python-интерпретатор сможет выполнить файл только по полному пути к нему.

1. **Как мутация глобального значения используется в потокобезопасности?**

Глобальная блокировка интерпретатора используется для того, чтобы по очереди запускать потоки, то есть не все сразу. Работает это все исключительно внутри программы и используется для того чтобы функция выполнялась на всех виртуальных машинах. Python позволяет переключаться между потоками для более высокой производительности, за счет использования инструкций байт-кода, который, в свою очередь, используется для обеспечения платформенной независимости. Метод sys.setcheckinterval() предназначен для переключения во время реализации программы и инструкции. Это дает понимание того, как работает самый нижний уровень архитектуры и то, что можно использовать байт-код в вашей реализации, чтобы сделать ее более платформенно независимой. За счет свойства атомарности можно использовать общие переменные в качестве встроенных типов данных.

1. **Модуль, предназначенный для чтения и записи двоичных д**анных, известен более как struct. Этот модуль предоставляет функционал, в котором половина всего состоит из классов string. Данные классы содержат двоичные данные, которые представлены в виде чисел, которые преобразованы в python-объекты для дальнейшего использования и наоборот.
2. **Модуль popen2()** используется для запуска под-процессов, но из-за сложности таких процессов, как: блокировка потоков, которая блокирует процесс, что приводит к ожиданию выходных данных от «потомка», а «потомок», в свою очередь, ждет данные на вход.
3. **Какие существуют, различные методы генерации рандомных чисел**? Модуль random – это стандартный модуль, который используется для генерации рандомных чисел.
4. **Singleton** – это паттерн, предназначенный для того, чтобы ограничивать количество экземпляров, которые могут быть использованы одним классом. Также этот паттерн позволяет расшаривать один и тот же объект среди многих других частей кода. Это позволяет использовать глобальные переменные, поскольку фактически используемые данные скрыты интерфейсом класса singleton. Интерфейс класса singleton имеет только один публичный член класса и один метод этого класса Handle. Приватные конструкторы не используются для создания объектов, которые используются вне класса. Процесс ждет статического члена функции для того, чтобы создать новый экземпляр и вернуть singleton-объект:
5. **Модуль Pickle** принимает любой объект Python и преобразует его в строку и выгружает его в файл с помощью dump функции, этот процесс называется pickling. Обратный процесс извлечения Python объекта из строки, соотвественно называется unpickling.
6. **Что такое пространство имён в Python?** В Питоне каждое введенное имя имеет место, где оно живет и может быть подключено к сети. Это известно как пространство имен. Оно похоже на окошко, где имя переменной отображается на размещаемом объекте. Всякий раз, когда переменная ищется, это окошко будет искаться, чтобы получить соответствующий объект.
7. **Что такое лямбда функции в Python**? Это анонимная функция с одним выражением, часто используемая в качестве inline функции.
8. **Что такое pass в Python?** Пропуск означает, что в составном заявлении не должно быть пустого места, и там не должно быть ничего написано, то есть он является владельцем места в составном заявлении, где не должно быть пустого места.
9. **Что такое Unittest в Python?** Униттестирование на Python известно как юниттест. Она поддерживает обмен настройками, автоматизированное тестирование, код завершения работы тестов, агрегирование тестов в коллекции и т.д.
10. **В Питоне что режут (slicing)?** Механизм выбора ряда элементов из типов последовательности, таких как список, кортеж, строки и т.д., известен как нарезка (slicing).
11. **Упомяните об использовании сплит-функции на Python?** Использование функции split на Python заключается в том, что она разбивает строку на более короткие строки с помощью заданного разделителя. Она дает список всех слов, присутствующих в строке.