Компетентності рівня junior для Python	Загальна оцінка компетентності (max. 9)	Компанії		
		IT-Jim	SoftServe	Avenga
1. Основи Python. Базові знання з програмування				
1.1 Змінні та типи даних. Структури даних. Знає різні типи даних та структури у Python - числа, рядки, булеві значення, списки (lists), кортежі (tuples), словники(dictionaries) та множини(sets), здатний використовувати їх для зберігання даних, усвідомлює правила оголошення та присвоєння значень змінним. Використовує спеціальні операції та методи для створення та ініціалізації, роботи зі структурами даних. Практикує імпорт бібліотек для використання готових функцій та класів.	**9**	3	3	3
1.2 Оператори та цикли, інструкції мови, умовні конструкції Ідентифікує математичні оператори, логічні оператори та оператори порівняння. Знає умовні конструкції (if-else, if-elif-else) та вміє застосовувати їх для прийняття рішень у програмі. Використовує цикли (for, while) для ітерації та повторення коду. Застосовує рекурсію для розв'язання завдань. Розуміє поняття ітераторів та використовує їх для послідовного перебору елементів. Знає про декоратори та застосовує їх для модифікації функцій, класів або методів. Вміє читати та розуміти код, який використовує ці оператори та інструкції.	**9**	3	3	3
1.3 Функції. Використання map, zip, lambda Вміє використовувати вбудовані функції та методи для обробки даних та керування програмою. Володіє навичками створення власних функцій, розуміє принцип використання функцій як значень для створення більш гнучких програм. Розуміє поняття області видимості та простору імен у функціях. Ідентифікує параметри та аргументи функції, параметри по замовчуванню, повернення значень з функції. Розуміє роботу функції map та її використання для ефективної обробки даних шляхом застосування іншої функції до кожного елементу послідовності. Використовує функцію zip для ітерації одночасно по кількох послідовностях та об'єднання елементів з кількох послідовностей в кортежі. Застосовує lambda-функції для створення анонімних коротких функцій без необхідності оголошення окремої функції.	**8**	3	3	2
1.4 Date. Обробка дати та часу. Має навички роботи зі спеціальним модулем datetime, обробкою даних у форматі дати/часу. Усвідомлює особливості роботи з датами, форматування дат та виконання операцій з ними. Розуміє поняття часової зони та різниці між локальним та універсальним часом (UTC). Використовує модуль pytz для роботи з різними часовими зонами. Здійснює розрахунок проміжків часу, прирощення часу та роботу з датами у минулому або майбутньому за допомогою класу timedelta для представлення різниці в часі. Використовує модуль dateutil для розбору дати з різних рядкових форматів.	**5**	1	2	2
1.5 Помилки та виключення Вміє працювати з помилками в Python та їх обробкою. Розрізняє синтаксичні та логічні помилки. Ідентифікує різні типи виключень (TypeError, NameError, ValueError тощо). Розуміє причини та умови, за яких вони виникають. Використовує конструкції try-except, try-except-else, try-except-finally для обробки виключень. Усвідомлює необхідність коректного закриття ресурсів у блоках finally для запобігання витоку пам'яті та інших проблем. Практикує використання ключового слова raise для генерації власного виключення. Розуміє особливості використання об'єкта виключення для отримання додаткової інформації про помилку (повідомлення та стек викликів). Здатний виводити власні повідомлення про помилки для полегшення розуміння причини виключення.	**5**	1	2	2
2. Теоретичний блок. Розуміння ООП				

2.1 Основні концепції ООП. Об'єкти Знає принципи використання об'єктів для організації та структурування даних та функціональності у програмі. Ідентифікує розрізнення між класом (шаблоном) та його екземплярами (об'єктами). Розуміє основні концепції ООП (інкапсуляція, спадкування та поліморфізм)	**8**	2	3	3
2.2 Класи, конструктори, властивості, модифікатори доступу Ідентифікує поняття класу та його роль в ООП. Знає синтаксис класів у Python та їх використання для створення об'єктів. Знає метод визначення конструктора класу, оператор створення нових об'єктів класу, процедуру визначення методів класу для виконання певних операцій над об'єктами класу, виклик методів класу на об'єктах класу. Усвідомлює поняття приватних властивостей та їх використання для приховування даних від зовнішнього доступу. Застосовує модифікатори доступу (public, private та protected). Працює з декоратором @property для визначення доступу до атрибутів через методи-геттери та контролю доступу та валідації значень атрибутів методами-сеттерами. Використовує спеціальні методи (__str__, __repr__, __len__) для визначення поведінки класу при виведенні на екран, репрезентації, обчислення довжини тощо Керується принципами SOLID при розробці програм.	**7**	2	3	2
2.3 Наслідування, множинне наслідування, MRO				
2.3.1 Розуміє поняття наслідування та його ролі в ООП. Вміє створювати ієрархії класів за допомогою наслідування. Розуміє механізм успадкування властивостей та методів між класами. Здатний застосовувати множинне наслідування. Усвідомлює особливості створення класів, які успадковують властивості та методи від кількох базових класів. Розуміє пріоритет методів при множинному наслідуванні та процедуру вирішення конфліктів.	**7**	1	3	3
2.3.2 Розуміє поняття MRO (Method Resolution Order) та його роль в визначенні послідовності пошуку методів при множинному наслідуванні. Застосовує метод mro() для отримання послідовності розрішення методів. Використовує лінійний порядок розрішення методів при множинному наслідуванні. Використовує абстрактні класи для створення інтерфейсів та забезпечення спільної поведінки класів.	**5**	1	2	2
2.4 Поліморфізм Ідентифікує поняття та концепцію поліморфізму та його роль в ООП. Вміє створювати та використовувати поліморфні методи, які можуть виконувати різні дії залежно від типу об'єкта. Використовує поліморфізм для заміни одного об'єкта іншим, який реалізує однаковий інтерфейс. Застосовує перевантаження операторів для надання різної поведінки операціям залежно від типу об'єкта. Знає визначення спеціальних методів класу, таких як __add__, __sub__, __str__, для перевантаження арифметичних операцій та репрезентації об'єкта. Використовує абстрактні класи для створення спільного інтерфейсу, який може бути реалізований різними класами Ідентифікує різницю між статичним та динамічним поліморфізмом. Розуміє, як поліморфізм використовується у контексті успадкування.	**7**	2	3	2
2.5 Global Interpreter Lock (GIL)				
2.5.1. Знає механізм забезпечення потокобезпеки під час виконання коду та усвідомлює вплив GIL на багатопотоковий код. Ідентифікує обмеження щодо використання багатопотоковості в CPython та заборону багатопотоковому коду виконувати паралельні операції на рівні CPU-інтенсивності. Усвідомлює можливість використання багатопотоковості для підтримки I/O-операцій в CPython. Розуміє що GIL не заважає виконанню операцій вводу-виводу (I/O), таких як читання/запис до файлів чи мережевих операцій.	**3**	1	1	1
2.5.2 Використовує модулі threading та concurrent.futures для створення та керування багатопотоковим кодом, використовує потоки для виконання фонових задач та покращення відгуку програми. Знайомий з іншими імплементаціями Python, такими як Jython, IronPython та PyPy, які мають відмінну модель GIL або його відсутність. Усвідомлює альтернативу застосування багатопроцесовості (multiprocessing) замість багатопотоковості для виконання коду паралельно на різних процесорах. Використовує оптимізаційні техніки, таких як векторизація, асинхронність та інші алгоритми, для зменшення впливу GIL на продуктивність.	**3**	1	1	1
3. Розширені знання Python.				
3.1 Область видимості. Ідентифікує локальні та глобальні змінні. Розуміє особливості областей видимості функцій та класів, використання ключових слів global та self. Ідентифікує процедуру оголошення функцій всередині інших функцій та наслідки створення вкладеної області видимості. Знає поняття часу життя змінних та поняття іменованого простору імен, де змінні та інші об'єкти зберігаються та знаходяться.	**7**	1	3	3
3.2 Python Enhancement Proposal 8 (PEP8) Усвідомлює важливість стилевого форматування та зрозумілості коду для покращення співпраці та читабельності Використовує правила форматування, дотримується правил іменування змінних (snake_case), констант (UPPER_CASE) та функцій (lowerCamelCase) Користується коментарями для пояснення складного або незрозумілого коду, докстрінгами (docstrings) для документування модулів, класів та функцій. Приділяє увагу анотації типів (type hints) для покращення зрозумілості та інтеграції інструментів статичного аналізу коду.	**7**	3	2	2

3.3 Робота з вбудованими бібліотеками мови Python (os, sys, shitil, itertools, collections etc.) Використовує модуль os для робота з операційною системою та оточенням, доступом до файлової системи та роботи файлами та директоріями Вміє взаємодіяти з інтерпретатором Python (доступ до аргументів командного рядка, керування потоками введення/виведення, отримання інформації про виконувану програму). Виконує операції з файлами та директоріями за допомогою модуля shutil Здійснює генерацію та обробку ітераторів із допомогою спеціальних функцій модуля itertools (комбінування, перестановки, повторення, фільтрація, об'єднання послідовностей тощо) Використовує при роботі зі структурами даних для ефективного зберігання та обробки даних спеціалізовані контейнери даних модуля collections (namedtuple, deque, Counter, defaultdict, OrderedDict тощо) Застосовує інші вбудовані модулі: модуль datetime для роботи з датами та часом, модуль random для генерації випадкових чисел, модуль math для математичних операцій та функцій тощо.	**8**	3	3	2
3.4 Принципи роботи DRY, YAGNI, KISS, Zen of Python Використовує рефакторинг для поліпшення коду згідно з принципами DRY та YAGNI. Розуміє необхідність розробки за допомогою функцій, класів та модулів для уникнення дублювання коду. Уникає надмірного написання коду або включення функціональності, яка на цей момент не потрібна. Фокусується на поточних вимогах та функціоналісті проєкту та уникає зайвого розширення або узагальнення Дотримується принципів розділення функцій і обов'язків між різними компонентами системи для поліпшення читабельності, підтримки і можливості повторного використання. Приділяє особливу увагу принципам, які повинні керувати розробкою коду в Python та сформульовані у документі PEP 20 (Python Enhancement Proposal 20).	**6**	2	1	3
3.5 Концепція MVC (Model-View-Controller) Ідентифікує архітектурний шаблон MVC, який визначає розподіл функціональності програми на моделі (Model), представлення (View) та контролери (Controller). Вміння створювати модель, яка представляє дані та бізнес-логіку додатка (наприклад, зберігання в базі даних) та забезпечує логіку обробки цих даних. Практикує створення представлень, які відповідають за візуалізацію даних, шляхом відображення даних з моделі та забезпечення коректної інтерактивності для користувача. Здатний створити контролер, який здійснює керування взаємодією між моделлю та представленням. Може організувати відповідну обробку дій користувача, взаємодію з моделлю для отримання/оновлення даних та оновлення представлення для відображення змін. Вміє правильно розділити функціональність між моделлю, представленням та контролером. Знає фреймворки, які підтримують побудову додатків за патерном MVC, такі як Django або Flask та вміє використовувати функціональність фреймворків для реалізації компонентів MVC. Розуміє інші шаблони архітектури програмного забезпечення, які доповнюють або розширюють концепцію MVC, наприклад, MVVM (Model-View-ViewModel).	**6**	2	2	2
3.6 AsincIO (асинхронне введення-виведення Asynchronous I/O) Володіє основними поняттями, таких як асинхронні функції та корутини, вміє визначати та викликати асинхронні функції та корутини з використанням ключових слів async та await. Розуміє роль петлі подій у виконанні асинхронного коду, вміє створювати та запускати петлю подій, створювати та планувати асинхронні задачі. Може визначати та викликати корутини-генератори (async generators) з використанням ключового слова async в оголошенні функції. Розуміє концепції синхронізації та паралелізму у контексті асинхронного програмування. Використовує asyncio.Lock та інші засоби синхронізації для управління доступом до ресурсів у багатопотоковому середовищі. Ідентифікує основні поняття, пов'язані з мережевими операціями, таких як сокети та протоколи. Застосовує асинхронні операції вводу-виводу з модулем asyncio для роботи з мережевими ресурсами. Використовує сторонні бібліотеки, які підтримують асинхронну розробку (такі як aiohttp для роботи з HTTP, aiomysql для роботи з MySQL тощо) для реалізації асинхронних операцій	**6**	1	3	2
3.7 Розуміння роботи API (типи, логіка роботи REST-застосунків) Знає основні принципи REST, такі як ресурси, URL-шляхи, HTTP-методи та параметри запиту. Розуміє структуру URL-шляхів та їх використання для ідентифікації ресурсів і підресурсів. Володіє навичками читання та створення даних у форматах JSON і XML, ідентифікує переваги та недоліки кожного формату і вміє обирати відповідний формат для конкретного випадку використання. Здатний використовувати різні методи автентифікації, такі як токени, сесії та OAuth. Вміє налаштовувати та використовувати механізми автентифікації та авторизації у REST-застосунках. Розуміє рівні доступу та особливості використання ролей користувачів для обмеження доступу до ресурсів.	**6**	1	2	3
3.8 Патерни розробки Знає про різні типи патернів проєктування, такі як структурні, поведінкові та породжувальні патерни. Розуміє основні принципи, які лежать в основі кожного патерну. Вміє впізнавати проблеми, для яких патерни проєктування є відповіддю, і використовувати патерни для їх розв'язання. Використовує популярні патерни розробки, такі як Singleton, Factory, Observer, Strategy, Decorator та інші. Вміє вибрати та використовувати відповідний патерн для конкретної ситуації розробки. Ідентифікує переваги та недоліки кожного патерну та здатний прийняти обґрунтоване рішення про їх використання.	**6**	3	1	2

3.9 Робота з базами даних SQL Має базові знання SQL syntax, вміє будувати запити (queries) для отримання інформації з бази даних, модифікації даних та виконання інших операцій. Вміє працювати з таблицями та схемами, розуміє зв'язки між таблицями. Здатний працювати з ключами та індексами для забезпечення ефективного доступу до даних. Розуміє типи з'єднань (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN) та може використовувати їх для об'єднання даних з різних таблиць та отримання з'єднаних даних Може будувати та виконувати SELECT-запити для отримання певних даних з таблиць та фільтрацію даних на основі певних умов, здійснювати їх агрегацію та групування Вміє працювати зі схемами баз даних, встановлювати права доступу користувачів тощо	**5**	0	2	3
3.10 Робота з базами даних NoSQL Знає різні інструменти та бібліотеки для роботи з NoSQL базами даних, такі як MongoDB, Cassandra, Redis, Couchbase Розуміє різницю між ACID (Atomicity, Consistency, Isolation, Durability) транзакціями, що характерні для реляційних баз даних, та BASE (Basically Available, Soft state, Eventually consistent) підходом, що використовується в NoSQL базах Вміє працювати зі специфічними структурами даних, які забезпечуються NoSQL базами, такими як дерева, графи, документи тощо Має навички тестування взаємодії NoSQL баз даних з іншими компонентами, такими як бекенд, фронтенд, API	**4**	0	2	2
3.11 Python for Data Science Здатний використовувати Python в якості інструменту для обробки, аналізу та візуалізації даних. Використовує бібліотеку NumPy для наукових обчислень, яка надає підтримку масивів та матриць, а також виконання різних обчислень та маніпуляцій з даними. Вміє зчитувати, маніпулювати та аналізувати структуровані дані з використанням DataFrame бібліотеки Pandas. Застосовує бібілотеку SciPy для наукових обчислень та чисельного аналізу: чисельна оптимізація, інтеграція, апроксимація та розв'язання диференціальних рівнянь. Практикує використання бібіліотек Seaborn та Matplotlib для візуалізації даних. Вміє створювати різноманітні типи графіків, діаграм та інші візуалізації для представлення даних. Здатний виконувати Big Data обробку великих обсягів даних з використанням бібілотеки PySpark. Виконує розподілені обчислення, маніпулює та аналізує великі набори даних з використанням Spark. Використовує різні алгоритми машинного навчання для класифікації, регресії, кластеризації та інших задач за допомогою інструментарія бібілотеки Scikit-learn. Здійснює глибоке навчання з використанням TensorFlow та Keras, користуючись нейронними мережами та глибокими моделями для розв'язання задач зображень, природної мови та інших.	**4**	3	0	1
4. Знання экосистеми Python				
4.1 Управління пам'яттю в Python Розуміє автоматичне управління пам'яттю в Python з використанням механізму збору сміття, його роль у вивільненні невикористовуваної пам'яті. Використовує об'єктну модель пам'яті Python, де об'єкти створюються, зберігаються та видаляються. Вміє працювати з посиланнями на об'єкти та розуміє вплив цього на передачу аргументів функцій. Ідентифікує відмінності між передачею аргументів за значенням та за посиланням. Знає мутабельні та імутабельні типи даних та їхні наслідки для роботи з пам'яттю. Вміє працювати з контекстними менеджерами (with statement) для автоматичного вивільнення ресурсів, таких як файли або з'єднання з базою даних. Використовує генератори та ітератори для роботи з великими наборами даних без необхідності зберігати всю інформацію в пам'яті одночасно. Застосовує методи оптимізації пам'яті, такі як використання слабих посилань (weak references), кешування або генераторів списків.	**5**	2	1	2
4.2 Знання одного з інструментів web framework (Django, Flask, AIOHTTP або Sanic) Розуміє концепції веб-фреймворка та його ролі у розробці веб-додатків. Знає архітектуру та основні компоненти веб-фреймворка, такі як модель-представлення-контролер (MVC), маршрутизація, обробники (handlers), шаблони тощо. Вміє встановлювати веб-фреймворк та його залежності з використанням інструментів управління пакетами, таких як pip або conda. Здатний налаштувати основні параметри та конфігурацію веб-фреймворку для потреб проекту. Може визначити маршрути та налаштувати обробники для різних URL-адрес. Використовує шаблони для відображення динамічного контенту на сторінках. Застосовує статичні файли, таких як CSS та JavaScript, для оформлення веб-сторінок. Може здійснювати взаємодію з базами даних з використанням веб-фреймворку, використовуючи ORM (Об'єктно-реляційне відображення). Вміє створювати запити до бази даних, виконувати CRUD-операції (створення, читання, оновлення, видалення) тощо. Використовує механізм сесій, який надається SQLAlchemy, для взаємодії з базою даних. Здатний створювати та застосовувати міграційні скрипти для оновлення структури бази даних без втрати даних. Може створювати та керувати сесіями та транзакціями в SQLAlchemy для забезпечення цілісності та консистентності даних.	**6**	1	3	2

4.3 Розуміння роботи фреймворків тестування (Unittest або PyTest) Розуміння концепцій Patching та Mocking Розуміє значення тестування в процесі розробки програмного забезпечення. Ідентифікує типи тестів, такі як модульні тести, інтеграційні тести та функціональні тести. Вміє створювати тести з використанням класів TestCase, методів assert та інших функціоналів, які надає Unittest. Практикує використання механізму фікстур для підготовки тестового середовища та завантаження необхідних даних перед виконанням тестів в фреймворку PyTest. Володіє навичками створення маркерів для тестів з метою їх категорізації. Здатний параметризувати тести, щоб виконати їх з різними вхідними даними, використовує декоратори. Розуміє сутність Patching та Mocking, їх призначення та як вони використовуються для контролю над поведінкою об'єктів та функцій під час тестування. Використовує функцію patch з бібліотеки unittest.mock для заміни об'єктів та функцій в тестових сценаріях як за допомогою декоратора, так і через контекстний менеджер. Здатний створювати замінники за допомогою функцій Mock або MagicMock з бібліотеки unittest.mock. Розуміє процес встановлення поведінки замінників та перевірки викликів до них. Практикує налаштування поведінкм замінників (повернення певних значень, підкидання винятків, виклик функцій та перевірка переданих аргументів). Розуміє важливість правильного управління ресурсами та використання підходу "з patch до моків" для забезпечення належного відновлення стану після виконання тестування. Знає різні способи генерації звітів з результатами тестування. Здатний проводити аналіз та інтерпретацію результатів тестів для виявлення проблем та вдосконалення якості коду.	**4**	0	3	1
5. Знання Web-технологій. Основи HTML та CSS, JavaScript				
5.1 Загальні питання веб технологій Розуміє клієнт-серверну архітектуру. Знає основні мережеві протоколи, розуміє принцип роботи протоколу HTTP, його основні методи, статусні коди та їх значення. Вміє працювати з заголовками HTTP, формувати HTTP-запити, вказуючи метод, шлях, заголовки та дані. Може обробляти HTTP-відповіді, читати статусні коди, заголовки та вміст відповіді. Розуміє принципи взаємодії із зовнішніми API з використанням HTTP-запитів Має навички використання інструментів розробника браузера для відстеження запитів мережі та аналізу заголовків та вмісту відповідей. Вміє використовувати cookie, LocalStorage для зберігання даних користувача	**5**	1	1	3
5.2 Загальна структура HTML-документа. Теги та атрибути. Семантика Знає основні елементи HTML-документу та вміє створювати з них правильну структуру. Знає основні HTML-теги, розуміє їх призначення та особливості, вміє обирати та використовувати найбільш доречні HTML-теги для розмітки різних типів контенту. Використовує семантичні елементи для більш чіткого та структурованого опису вмісту вебсторінки. Розуміє роль та значення атрибутів та їх вплив на поведінку та відображення елементів, вміє додавати та налаштовувати атрибути для елементів HTML. Розуміє важливість валідного HTML-коду та його ролі для сумісності, доступності та оптимізації вебсторінки. Здатний перевіряти та виправляти помилки валідації HTML за допомогою валідаторів та інструментів розробника.	**5**	0	3	2
5.3 Основні CSS-властивості та селектори. CSS-анімації. Flexbox. Grid Розуміє принципи каскадності та спадкування стилів. Вміє працювати з css-властивостями, css-селекторами, та їх ієрархією. Ідентифікує поняття блокової моделі, потоку, позиціонування. Має досвід використання CSS-технік flexbox та grid, вміє налаштовувати гнучку сітку для різних екранів та пристроїв, використовує медіазапити для створення адаптивних вебсторінок. Працює з різними одиницями вимірювання	**3**	0	2	1
5.4 Основи JavaScript Знає синтаксис JavaScript, змінні, оператори та умовні конструкції. Ідентифікує основні поняття, такі як функції, масиви та об'єкти. Вміє маніпулювати DOM (Document Object Model) за допомогою JavaScript, змінювати вміст та стилі елементів, додавати та видаляти елементи. Може здійснювати обробку подій та взаємодію з користувачем. Розуміє процес здістнення асинхронних запитів до сервера за допомогою AJAX (Asynchronous JavaScript and XML). Використовує вбудовані функції JavaScript, такі як fetch(), для отримання та відправлення даних на сервер.	**5**	0	3	2
5.5 Основи Bootstrap (шаблони, CSS, компоненти, JavaScript) Знає основні можливості Bootstrap, такі як використання готових шаблонів та компонентів для швидкої розробки веб-інтерфейсу. Розуміє базові CSS-класи та структуру стилів у Bootstrap. Здатний використовувати JavaScript-компоненти Bootstrap для створення інтерактивності та покращення функціональності на веб-сторінці.	**2**	0	1	1
5.6 Розуміння роботи API (типи, логіка роботи REST-застосунків) Знає основні принципи REST, такі як ресурси, URL-шляхи, HTTP-методи та параметри запиту. Розуміє структуру URL-шляхів та їх використання для ідентифікації ресурсів і підресурсів. Володіє навичками читання та створення даних у форматах JSON і XML, ідентифікує переваги та недоліки кожного формату і вміє обирати відповідний формат для конкретного випадку використання. Здатний використовувати різні методи автентифікації, такі як токени, сесії та OAuth. Вміє налаштовувати та використовувати механізми автентифікації та авторизації у REST-застосунках. Розуміє рівні доступу та особливості використання ролей користувачів для обмеження доступу до ресурсів.	**4**	1	1	2
6. Інструментарій розробника				

6.1 Системи контролю версій - GIT Розуміє git flow - методологію роботи з гілками для ефективного управління процесом розробки. Здатний створювати репозиторії та ініціалізувати проєкти з використанням Git. Розуміє та використовує команди Git (commit, push, pull, fetch, checkout, add, status, revert та інші). Вміє працювати з гілками (branches) для розробки функціональності відокремлено від основної гілки (наприклад, розробка нових функцій у власних гілках та їх об'єднання в основну гілку). Знайомий з процедурою створення пулл-реквестів для обговорення та злиття змін з різних гілок.	**9**	3	3	3
6.2 Навички використання IDE Вміє працювати з популярними IDE, такими як Visual Studio Code, IntelliJ IDEA, Eclipse, або WebStorm. Використовує редактор коду, налаштування розширень та плагінів для поліпшення продуктивності розробки. Застосовує вбудовані інструменти IDE для налагодження коду, аналізу помилок, рефакторингу та контролю якості коду.	**8**	3	3	2
6.3 Навички використання командного рядка (Terminal) Має базове розуміння навігації по файловій системі через командний рядок. Використовує команди для перегляду, створення, зміни, видалення та архівації файлів та тек через командний рядок. Застосовує через командний рядок команди для роботи з Git, таких як клонування репозиторію, створення гілок, комітів та інші.	**7**	3	1	3
6.4 Знання Linux-середовища, Docker Розуміє основні концепції та принципи роботи з операційною системою Linux. Вміє працювати з командним рядком Linux, виконувати команди, керувати файловою системою, управляти користувачами та правами доступу. Працює зі скриптовими мовами, такими як Bash, для автоматизації рутинних завдань та конфігурації середовища. Вміє створювати, налаштовувати та управляти Docker контейнерами для окремих мікросервісів. Знає Docker-команди для роботи з образами, контейнерами, мережами, томами та іншими компонентами Docker.	**7**	3	2	2
7. Розмовна англійська				
7.1 Правильна побудова фрази з узгодженням часу Вміє використовувати відповідні часові форми дієслів і структури речень для точного вираження часу в комунікації. Здатний правильно використовувати часові форми, такі як Present Simple, Present Continuous, Past Simple, Past Continuous, Future Simple, для передачі правильного значення в повідомленні.	**8**	2	3	3
7.2 Професійна it термінологія в розмові з замовником Знає і використовує професійні IT-терміни та скорочення, які використовуються в IT-індустрії, для чіткого та зрозумілого спілкування зі замовником. Вміє перекладати складні технічні концепції на доступну мову для замовника, не знайомого з IT-галуззю.	**7**	3	2	2
7.3 Професійна it термінологія у діловому звіті Знає і використовує спеціалізовану IT-термінологію та фразеологію при написанні ділових звітів. Вміє передати інформацію про технічні питання та проєкти зрозуміло та конкретно, використовуючи адекватні IT-терміни та абревіатури.	**7**	3	2	2
7.4 Професійна термінологія для Python Знає та розуміє специфічну термінологію, що використовується в контексті Python-розробки Розуміє та використовує специфічні терміни, патерни, алгоритми, фреймворки, бібліотеки та інші інструменти, які використовуються в Python-розробці.	**5**	1	1	3
7.5 Неформальне спілкування (small talks) Вміє вести неформальну розмову, розпочинати діалоги та підтримувати невимушену атмосферу. Знає загальновживану термінологію та фрази, які використовуються у неформальних розмовах, таких як загальні питання про погоду, цікаві події, особисті захоплення, спорт, фільми тощо.	**5**	3	1	1

Коментарі до таблиці

Оцінки компетентностей вказані станом на 2023/2024 рр., по шкалі від 0 до 3.

«3» - компетентність вкрай важлива, якщо кандидат нею не володіє, його подальше просування на вакансію неможливе.

«2» - компетентність, знання якої обов'язково знадобиться кандидату при подальшому професійному зростанні.

«1» - некритична компетентність, якою можна знехтувати.

«0» - неактуальна компетентність.

Зауваження від компаній

IT-Jim

3.1 - Область видимості не розширені, а базові знання Python. Перенести в блок 1.

SoftServe

2.5.1 - Ці компетентності не так важливі саме Junior розробнику, їх освоєння буде важчим, ніж пунктів з "Розширені знання Python"

3.2 - Базова тема, яку важливо давати на самому початку вивчення мови, щоб не будувались звички писати "поганий" код, слід перенсти її до "базових знань з програмування".

3.4 - гарно поєднується з 3.2

3.9 та 3.10 можна поєднати в 1 компетенцію, по навичкам роботи з БД, знати основні різниці між SQL та noSQL базами

3.11 - Data Science - інший напрямок, який можно розибити на окрмеі компетенції, пов'язні з чисельними методами, штучним інтелектом, візуалізацією даних. тощо

4.1 - Для Junior можна зменшити перелік вимог та поєднати з 1.1

5.1 - Можливо можна поєднати з 3.7, додати ще декілька слів про застарілиі підходи (soap) та більш новий QraphQL

5.5 - Поєднати з 5.3

5.6, 3.7 та можливо 5.1 можна в 1 компетенці.

6.3 - Поєднати з 6.4

7.4 - Поєднати з 7.3

7.5 - Поєднати з 7.2