3. Требования к коду

Вопрос 329. Какие документы устанавливают основные требования к коду Python?

Ответ: это документы PEP8, PEP257

Вопрос 330. Как именуются константы в Python и можно ли их изменять?

Ответ: константы в Python пишутся заглавными буквами, можно использовать нижнее подчеркивание. Менять их значение можно в процессе выполнения программы.

Вопрос 331. Как именуются переменные в Python?

Ответ: все буквы в названии переменной маленькие, можно использовать нижнее подчеркивание.

Вопрос 332. Как именуются функции в Python?

Ответ: пишутся с помощью маленьких букв, должны отражать, что делает функция.

Вопрос 333. Как именуются классы в Python?

Ответ: каждое слово начинается с заглавной буквы. Нижние подчеркивания не используются.

Вопрос 334. Какая максимальная длина строки должна быть согласно PEP8?

Ответ: 79 символов

Вопрос 335. Порядок импортов и их сортировка?

Ответ: вначале импортируются модули стандартной библиотеки Python, потом модули скачанных библиотек, потом свои собственные модули. Они отделяются друг от друга строкой. Импортируется чаще всего через from, импортируемые компоненты должны быть в алфавитном порядке.

Вопрос 336. Можно ли применять бексплеши для переноса?

Ответ: если нет других способов переноса, то можно

Вопрос 337. Какие кавычки должны быть в программе?

Ответ: двойные или одиночные, но везде только одного из этих двух типов.

Вопрос 338. Стоит ли всегда использовать ключевое слово else в условной конструкции?

Ответ: В функции если else можно заменить на GuardBlock, то лучше так и поступить

Вопрос 339. Какой конструкцией должен быть закрыт код в исполняемом ру-файле?

Ответ: конструкцией if __name__ == “__main__”

_____

Вопрос 340. Если в программе возможен выбор между списками и кортежами, что предпочтительнее выбирать?

Ответ: кортежи

_____

Вопрос 341. Какая специфика использования f-строк?

Ответ: применяется только подстановка переменных и нет логических или арифметических операций, вызов функций подобной динамики.

_____

Вопрос 342. Как должны быть названы переменные?

Ответ: переменные названы в соответствии с их смыслом, по-английски, нет однобуквенных названий и транслита. В названии переменной не должен содержаться ее тип. При необходимости применять аннотацию типов.

_____

Вопрос 343. О чем документ PEP257?

Ответ: о документированных строках – Docstrings

_____

Вопрос 344. Что такое документированная строка?

Ответ: Документационная строка — это строковый литерал, являющийся первой инструкцией в определении модуля, функции, класса или метода. Такая строка становится доступна при обращении с специального атрибута _ _ doc _ _ этого объекта.

_____

Вопрос 345. Для каких объектов применяется Docstrings?

Ответ: Docstring применяется для всех функций, классов и библиотек.

_____

Вопрос 346. Расскажите основные правила документирования?

Ответ: Для согласованности всегда используйте """тройные двойные кавычки""" вокруг документационной строки. Документационная строка — это «фраза», заканчивающаяся точкой. Если весь docstring не помещается в строку, вы можете вынести закрывающие кавычки на отдельную линию.

_____

Вопрос 347. Стоит ли документировать каждую строчку кода?

Ответ: Комментировать каждую строчку кода считается плохим тоном. Используйте комментарии, когда нужно: указать на участок кода, на который стоит обратить внимание; пояснить сложные алгоритмы или логику; указать на код, который нужно доработать; указать на код, который хочется позже разобрать.

Вопрос 348. Какие существуют инструменты для Python для проверки кода на соответствие стандартам?

Ответ: используют линтеры - в программировании линтерами принято называть инструменты для анализа кода, которые помогают находить места, где код не соответствует указанному стандарту.

Flake8 – для проверки PEP8, mypy – для проверки PEP257

Вопрос 349. В чем суть принципов YAGNI, KISS, DRY?

Ответ: Принцип YAGNI (You Ain't Gonna Need It) гласит о том, что не следует создавать функциональность, которая может быть не нужна в настоящий момент или в будущем.

Принцип KISS (Keep It Simple, Stupid) предполагает, что решение должно быть максимально простым и понятным.

Принцип DRY (Don't Repeat Yourself) говорит о том, что необходимо избегать повторения кода и использовать модульность для разделения функциональности на различные части.

Вопрос 350. Что такое аннотация типов и для чего она нужна?

Ответ: Чтобы держать типизацию под контролем — применяют аннотации типов данных (_Type Hints_, дословный перевод с английского — «подсказки типов»). Python не оставляет аннотации совсем без внимания: он считывает _Type Hints_ и сохраняет их в словарь `__annotations__`. Содержимое этого словаря можно вывести на экран:

print(_ _ annotations _ _ _) >>> {'name': <class 'str'>, 'var_for_bool': <class 'bool'>}

Вопрос 351. Какая библиотека существует для аннотации типов и как она работает?

Ответ: Библиотека (модуль) typing нужна для аннотации типов в коде программы, чтобы вы и другие программисты, которые читают ваш код, понимали, что принимает и отдает та или иная функция. Аргумент функции принимает строки или числа, а переменная может содержать число или None. В таких случаях для аннотирования типов данных применяются компоновщики.

С помощью компоновщика можно указать для переменной несколько возможных типов данных.

Вопрос 352. Что аннотирует типы данных, которые могут содержать 2 типа данных?

Ответ: Метод Optional

```
from typing import Optional

text: Optional[str] - переменная text ожидает str или None
text = None - проблем нет
```

Вопрос 353. Что используется, если переменная должна принимать данные нескольких разных типов?

Ответ: применяют аннотацию Union

```
from typing import Union


def hundreds(x: Union[int, str]) -> str:
    return str(x * 100)

hundreds(100)
hundreds('сто')
```

Вопрос 354. Что применять, когда не нужно ограничивать возможные типы переменной?

Ответ: Метод Any – но лучше его никогда не применять

```
x: Any
x = 12210
x = 'Строка'
x = True
x = None
```

Вопрос 355. Как производится аннотация коллекций?

Ответ: Если используется версия Python 3.9+ тогда импортировать typing не нужно, а сразу без импорта записать dict, list, tuple, set. Если версия питона ниже есть from typing import Sequence, Dict, List, Tuple, Set

---

Вопрос 356. Что использовать если функция передаётся в качестве аргумента в другую функцию или в метод и там вызывается?

Ответ: такому аргументу присваивается тип Callable.

```
from typing import Callable


def printer() -> None:
    print("Вызови меня!")
def returner(word: str) -> str:
    return word

def app(printed_inside: Callable[[], None], returned_inside:
Callable[[str], str]) -> None:
    printed_inside()
    print(returned_inside('Нет, вызови меня!'))

app(printer, returner)
app(printer, printer)
```
