Приложение №5 – Свод отраслевых (индустриальных) стандартов по компетенции «Нейросети и большие данные (юниоры)»

УТВЕРЖДЕНО  
Менеджер компетенции  
«Нейросети и большие данные»

II чемпионатный цикл 2025г.

\_\_\_\_\_\_\_\_\_\_\_\_\_/Левицкий И.А./

Индустриальный эксперт

регионального этапа чемпионата

II чемпионатный цикл 2025г.

\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_/

СВОД ОТРАСЛЕВЫХ (ИНДУСТРИАЛЬНЫХ) СТАНДАРТОВ

по компетенции «Нейросети и большие данные (юниоры)»

для юниорской категории

Свод отраслевых (индустриальных) стандартов по компетенции «Нейросети и большие данные (юниоры)» (далее – Свод) создан при взаимодействии с представителями профильной индустрии реального сектора экономики Российской Федерации и нацелен на применение актуальных производственных навыков и стандартов в рамках выполнения Конкурсного задания по компетенции «Нейросети и большие данные (юниоры)». Соблюдение Свода направлено на реализацию тесного взаимодействия Всероссийского чемпионатного движения и индустриальных партнеров чемпионата.

**1. Основные положения**

Свод разделен на два логических раздела: раздел со стандартами по написанию программных продуктов и раздел по оформлению презентации.

Свод используется при оценке Конкурсного задания в тех аспектах Критериев оценки, где это применимо.

При выполнении Конкурсного задания допустимо незначительное отклонение от требований, описанных в данном Своде.

**2. Раздел отраслевых (индустриальных) стандартов по написанию программных продуктов**

**2.1. Стандарт написания кода (PEP/8)**

**2.1.1. Внешний вид кода**

**Отступы**

Необходимо использовать 4 пробела на каждый уровень отступа.

Продолжительные строки должны выравнивать обернутые элементы либо вертикально, используя неявную линию в скобках (круглых, квадратных или фигурных), либо с использованием висячего отступа. При использовании висячего отступа следует применять следующие соображения: на первой линии не должно быть аргументов, а остальные строки должны четко восприниматься как продолжение линии.

Закрывающие круглые/квадратные/фигурные скобки в многострочных конструкциях могут находиться под первым не пробельным символом последней строки списка.

**Табуляция или пробелы**

Пробелы - самый предпочтительный метод отступов.

Табуляция должна использоваться только для поддержки кода, написанного с отступами с помощью табуляции.

В версии Python 3 запрещается смешивание табуляции и пробелов в отступах.

**Максимальная длина строки**

Ограничение длины строки максимум 79 символами.

Для более длинных блоков текста с меньшими структурными ограничениями (строки документации или комментарии), длину строки следует ограничить 72 символами.

Предпочтительный способ переноса длинных строк является использование подразумеваемых продолжений строк Python внутри круглых, квадратных и фигурных скобок. Длинные строки могут быть разбиты на несколько строк, обернутые в скобки. Это предпочтительнее использования обратной косой черты для продолжения строки.

Обратная косая черта все еще может быть использована время от времени. Например, длинная конструкция with не может использовать неявные продолжения, так что обратная косая черта является приемлемой.

**Пустые строки**

Необходимо отделять функции верхнего уровня и определения классов двумя пустыми строками.

Определения методов внутри класса разделяются одной пустой строкой.

Дополнительные пустые строки возможно использовать для разделения различных групп похожих функций. Пустые строки могут быть опущены между несколькими связанными однострочниками (например, набор фиктивных реализаций).

Используйте пустые строки в функциях, чтобы указать логические разделы.

**Кодировка исходного файла**

Кодировка Python должна быть в формате UTF-8.

**Импорты**

Каждый импорт, как правило, должен быть на отдельной строке.  
Импорты всегда помещаются в начале файла, сразу после комментариев к модулю и строк документации, и перед объявлением констант.  
Импорты должны быть сгруппированы в следующем порядке:

* 1. импорты из стандартной библиотеки;
  2. импорты сторонних библиотек;
  3. импорты модулей текущего проекта.

Необходимо вставлять пустую строку между каждой группой импортов.

Рекомендуется абсолютное импортирование, так как оно обычно более читаемо и ведет себя лучше, если импортируемая система настроена неправильно.

Тем не менее, явный относительный импорт является приемлемой альтернативой абсолютному импорту, особенно при работе со сложными пакетами, где использование абсолютного импорта было бы излишне подробным: в стандартной библиотеке следует избегать сложной структуры пакетов и всегда использовать абсолютные импорты. Неявные относительно импорты никогда не должны быть использованы, и были удалены в Python 3.

Шаблоны импортов (from import \*) следует избегать, так как они делают неясным то, какие имена присутствуют в глобальном пространстве имён, что вводит в заблуждение как читателей, так и многие автоматизированные средства.

**Пробелы в выражениях и инструкциях**

Необходимо избегать использования пробелов в следующих ситуациях:

* внутри круглых, квадратных или фигурных скобок;
* перед запятой, точкой с запятой или двоеточием;
* перед открывающей скобкой, после которой начинается; список аргументов при вызове функции;
* перед открывающей скобкой, после которой следует индекс или срез.

Использование более одного пробела вокруг оператора присваивания или любого другого для того, чтобы выровнять его с другим.

Всегда необходимо окружать эти бинарные операторы одним пробелом с каждой стороны: присваивания (=, +=, -= и другие), сравнения (==, <, >, !=, <>, <=, >=, in, not in, is, is not), логические (and, or, not).

Если используются операторы с разными приоритетами, необходимо добавить пробелы вокруг операторов с самым низким приоритетом.

Никогда не используется более одного пробела, и всегда используется одинаковое количество пробелов по обе стороны бинарного оператора.  
Не используются пробелы вокруг знака =, если он используется для обозначения именованного аргумента или значения параметров по умолчанию.  
Не используются составные инструкции, несколько команд в одной строке.  
Допустимо писать тело циклов while, for или ветку if в той же строке, если команда короткая, но если команд несколько, необходимо разделить тело цикла и сам цикл, а также сокращать/разделять длинные строки.

**Комментарии**

Комментарии должны являться законченными предложениями. Если комментарий — фраза или предложение, первое слово должно быть написано с большой буквы, если только это не имя переменной, которая начинается с маленькой буквы.

Если комментарий короткий, можно опустить точку в конце предложения. Блок комментариев обычно состоит из одного или более абзацев, составленных из полноценных предложений, поэтому каждое предложение должно заканчиваться точкой.

Необходимо проставлять два пробела после точки в конце предложения.

**Блоки комментариев**

Блок комментариев обычно объясняет код (весь, или только некоторую часть), идущий после блока, и должен иметь тот же отступ, что и сам код. Каждая строчка такого блока должна начинаться с символа # и одного пробела после него (если только сам текст комментария не имеет отступа).

Абзацы внутри блока комментариев разделяются строкой, состоящей из одного символа #.

**"Строчные" комментарии**

Такой комментарий находится в той же строке, что и инструкция. "Встрочные" комментарии должны отделяться по крайней мере двумя пробелами от инструкции. Они должны начинаться с символа # и одного пробела.

Комментарии в строке с кодом не нужны и только отвлекают от чтения, если они объясняют очевидно.

**Строки документации**

Необходимо описывать документацию для всех публичных модулей, функций, классов, методов. Строки документации необязательны для приватных методов, но лучше написать, что делает метод. Комментарий нужно писать после строки с def.

Для однострочной документации допустимо оставить закрывающие кавычки на той же строке.

**2.1.2. Соглашения по именованию**

**Стили имен**

Существует много разных стилей. Ниже приведены пояснения, какой стиль именования используется, независимо от того, для чего он используется.

Обычно различают следующие стили:

* b (одиночная маленькая буква)
* B (одиночная заглавная буква)
* lowercase (слово в нижнем регистре)
* lower\_case\_with\_underscores (слова из маленьких букв с подчеркиваниями)
* UPPERCASE (заглавные буквы)
* UPPERCASE\_WITH\_UNDERSCORES (слова из заглавных букв с подчеркиваниями)
* CapitalizedWords (слова с заглавными буквами, или CapWords, или CamelCase). Замечание: в случае использования аббревиатуры в таком стиле, необходимо прописывать все буквы аббревиатуры заглавными — HTTPServerError лучше, чем HttpServerError.
* mixedCase (отличается от CapitalizedWords тем, что первое слово начинается с маленькой буквы)
* Capitalized\_Words\_With\_Underscores (слова с заглавными буквами и подчеркиваниями)

Ещё существует стиль, в котором имена, принадлежащие одной логической группе, имеют один короткий префикс. Этот стиль редко используется в Рython.

**Имена, которых следует избегать**

Никогда не применяются символы l (маленькая латинская буква «эль»), O (заглавная латинская буква «о») или I (заглавная латинская буква «ай») как однобуквенные идентификаторы.

В некоторых шрифтах эти символы неотличимы от цифры один и нуля. В случае крайней необходимости использования символа l, рекомендуется вместо него использовать заглавную L.

**Имена модулей и пакетов**

Модули должны иметь короткие имена, состоящие из маленьких букв. Можно использовать символы подчеркивания, если это улучшает читабельность. То же самое относится и к именам пакетов, однако в именах пакетов не рекомендуется использовать символ подчеркивания.

Так как имена модулей отображаются в именах файлов, а некоторые файловые системы являются нечувствительными к регистру символов и обрезают длинные имена, очень важно использовать достаточно короткие имена модулей.

Когда модуль расширения, написанный на С или C++, имеет сопутствующий python-модуль С/С++ модуль начинается с символа подчеркивания, например, \_socket.

**Имена классов**

Имена классов должны обычно следовать соглашению CapWords.

Вместо этого могут использоваться соглашения для именования функций, если интерфейс документирован и используется в основном как функции.

**Имена исключений**

Так как исключения являются классами, к исключениям применяется стиль именования классов. Однако допустимо добавить Error в конце имени, если, конечно, исключение действительно является ошибкой.

**Имена глобальных переменных**

Глобальные переменные используются только внутри одного модуля. Необходимо руководствоваться теми же соглашениями, что и для имен функций.

Необходимо добавлять в модули, которые написаны так, чтобы их использовали с помощью from M import \*, механизм \_\_all\_\_, чтобы предотвратить экспортирование глобальных переменных.

**Имена функций**

Имена функций должны состоять из маленьких букв, а слова разделяться символами подчеркивания — это необходимо, чтобы увеличить читабельность.

Стиль mixedCase допускается в тех местах, где уже преобладает такой стиль, для сохранения обратной совместимости.

**Аргументы функций и методов**

Необходимо использовать self в качестве первого аргумента метода экземпляра объекта.

Необходимо использовать cls в качестве первого аргумента метода класса.

**Имена методов и переменных экземпляров классов**

Необходимо использовать тот же стиль, что и для имен функций: имена должны состоять из маленьких букв, а слова разделяться символами подчеркивания.

Необходимо использовать один символ подчеркивания перед именем для непубличных методов и атрибутов.

Чтобы избежать конфликтов имен с подклассами, необходимо использовать два ведущих подчеркивания.

**Константы**

Константы объявляются на уровне модуля и записываются только заглавными буквами, а слова разделяются символами подчеркивания.

**Разделение ответственности**

Код должен быть упорядоченный, который шаг за шагом объясняет каждую часть программы, предсказуемым и легко понятным образом.

**2.1.3. Принципы разработки программного кода**

При разработке программного продукта необходимо соблюдать следующие правила и принципы:

1. объектно-ориентированный подход - код должен быть организован в виде объектов, что способствует улучшению модульности и повторного использования кода;
2. избегания дублирования кода (DRY - Don't repeat yourself) - каждый элемент кода должен быть представлен единожды и в одном месте. Для этого следует избегать дублирования кода, выделяя общие части в отдельные функции или классы;
3. простота и четкость структуры (KISS - Keep it short and simple) - код должен быть как можно проще и не содержать излишней сложности. Это помогает легче понимать и сопровождать код;
4. минимизация сложности и избыточных абстракций - необходимо избегать ненужных усложнений и чрезмерного использования абстракций, которые могут затруднить понимание кода;
5. использование только необходимых библиотек - необходимо применять только те библиотеки, которые действительно необходимы для выполнения задач, а также избегать ненужного увеличения зависимостей проекта от сторонних библиотек;
6. соблюдать принципы SOLID.

**3. Отраслевые (индустриальные) стандарты по созданию презентации**

**3.1 Общие правила оформления презентаций**

**Общие требования:**

* На слайдах должны быть только тезисы, ключевые фразы и графическая информация (рисунки, графики и т.п.) – они сопровождают подробное изложение мыслей докладчика, но не наоборот;
* Не стоит заполнять слайд большим количеством информации. Наиболее важную информацию желательно помещать в центр слайда.

**Правила шрифтового оформления:**

* Рекомендуется использовать шрифты с засечками (Georgia, Palatino, Times New Roman);
* Размер шрифта: 24-54 пункта (заголовок), 18-36 пунктов (обычный текст);
* Курсив, подчеркивание, жирный шрифт, прописные буквы используются для смыслового выделения ключевой информации и заголовков;
* Не рекомендуется использовать более 2-3 типов шрифта;
* Основной текст должен быть отформатирован по ширине, на схемах – по центру

**Правила выбора цветовой гаммы:**

* Цветовая гамма должна состоять не более чем из 2 цветов и выдержана во всей презентации. Основная цель – читаемость презентации;
* Желателен одноцветный фон неярких пастельных тонов (например, светло-зеленый, светло-синий, бежевый, светло-оранжевый и светло-желтый);
* Цвет шрифта и цвет фона должны контрастировать (текст должен хорошо читаться, белый текст на черном фоне читается плохо);
* Оформление презентации не должно отвлекать внимания от её содержания.

**Графическая информация:**

* Рисунки, фотографии, диаграммы должны быть наглядными и нести смысловую нагрузку, сопровождаться названиями;
* Изображения (в формате jpg) лучше заранее обработать для уменьшения размера файла;
* Размер одного графического объекта – не более 1/2 размера слайда;
* Соотношение текст-картинки – 2/3 (текста меньше, чем картинок).

Практически неоспоримым является факт, что дизайн презентаций оказывает самое непосредственное влияние на мотивацию, скорость восприятия материала, утомляемость и ряд других важных показателей. Поэтому дизайн интерфейса не должен разрабатываться на интуитивном уровне. Требуется научно обоснованный, взвешенный и продуманный системный подход. Существует мнение, что наглядный материал не просто некоторая информация в чувственной форме представления, а информационная модель определенного профессионального опыта, которая должна соответствовать требованиям *эстетики*, *эргономики* и *дизайна*.

**3.2 Типичные недочеты и ошибки при создании презентаций**

В качестве наиболее типичных недочетов и ошибок при конструировании и оформлении электронных презентаций, часто снижающих их эффективность, можно выделить следующие.

***Часто встречающиеся ошибки стиля и оформления:***

* + Орфографические и стилистические ошибки, недопустимые в профессиональной среде;
  + Отсутствие единства стиля страниц:
    - одинаковая гарнитура и размер шрифта для всех заголовков (не менее 24 пунктов);
    - одинаковая гарнитура и размер шрифта для тестовых фрагментов (не менее 18 пунктов);
    - заголовки, номера страниц, кнопки перелистывания должны появляться в одном и том же месте экрана;
    - одинаковая цветовая гамма на всех станицах и т.п.
  + Неудачный выбор цветовой гаммы: использование слишком ярких и утомительных цветов, использование в дизайне более 3 цветов (цвет текста, цвет фона, цвет заголовка и/или выделения); использование темного фона со светлым текстом;
  + Использование разных фонов на слайдах в рамках одной презентации;
  + Отсутствие полей на слайдах;
  + Использование рисунков, фотографий плохого качества и с искажениями пропорций;
  + Отсутствие должного выравнивания текста, использование разного размера шрифта;
  + Низкая контрастность фон / текст;
  + Низкая контрастность гиперссылок (нужно помнить, что у гиперссылки три состояния: выбрана, по гиперссылке еще не переходили, по гиперссылке уже переходили – нужно обращать на цвет гиперссылки во всех состояниях);
  + Отсутствие или неясность связей в схемах или между компонентами материала на слайде;
  + Наличие различных эффектов переходов между слайдами и других раздражающих эффектов анимации, мешающих восприятию информации;
  + Слишком быстрая смена слайдов и анимационных эффектов (при автоматической настройке презентации), отсутствие учета скорости и законов восприятия зрительной информации.