|  |  |  |
| --- | --- | --- |
| Название этапа | Название листинга | Комментарии |
| Просто функция | simple\_func\_1 | Рассказываем, что в Python есть функции, и что они дают достаточно много простора для программиста. Функции можно понимать как подпрограммы, которые мы вызываем в тот момент, когда они нам нужны, чтобы они не исполнялись сразу. Моя любимая аналогия – с блендером, у которого есть какие-то функции, которые были заложены еще на заводе, однако мы ими пользуемся только в случае крайней необходимости |
| Система аргументов | def\_argument\_2 | Рассказываем про аргументы в функции, что они делают, и почему они так важны. Аргументы – переменные «на вход» функции, которые и позволяют использовать функции самыми разными способами, потому что функция делает полезную работу вне зависимости от того, что было заложено в ее инициализации |
| Позиционные аргументы | pos\_arguments\_3 | Один из самых очевидных способов получения аргумента – это позиционные аргументы, которые срабатывают просто по месту. Здесь почти что нечего добавить, кроме того, что такой способ представляется наиболее логичным, потому что применяется в большинстве других интерпретируемых языков, таких например, как PHP или JavaScript |
| Именованные аргументы | naming\_arg\_4 | Именнованные аргументы – куда более изысканные вариант, потому что он позволяют напрямую указать, в качестве какого аргумент вы передаете данные. Может быть применено для уменьшения путаницы в программе, однако может визуально увеличивать иницализацию функции и путаницу с параметрами по умолчанию |
| Дефолтный параметры | default\_param\_5 | Параметры по умолчанию действительно очень полезны, потому что вы можете поставить в функцию параметр по умолчанию и вызывать ее без аргументов, если вам необходимо достигнуть достаточной гибкости в выполнении программы. |
| Различные варианты вызова функций, которые выглядят одинаково | equalent\_6 | Здесь просто показываем, что какая-то визуальная путаница в Python вполне возможна. Внимательно присматривайтесь к тому, как выглядит функция! |
| Возвращение значение | return\_type\_7 | Однако функции не выглядели настолько интересно, если бы они не могли передавать значения, например друг в друга. Здесь вы вполне можете посмотреть на функции высшего порядка, если вы понимаете о чем я. А если нет, то это будет в расширенном функциональном программировании через несколько занятий. |
| Разбиение функций | drob\_8 | Функции можно и нужно дробить на отдельные! Здесь можно показать код из листинга drob.py, в котором у нас достаточно удачно разделяются функции на несколько отдельных, и потом происходит их вызов в функции, которая является функцией для композирования отдельных элементов |
| Необязательные аргументы и документация | unnesos\_arg\_9 | При инициализации функции стоит писать документации, только если ее работа не является абсолютно очевидной. Не стоит бессмысленно делать перевод ее названия, стоит описать данные, которые приходят к ней на вход, и те, которые она возвращает и какого типа. Также можно задавать необязательные аргументы на основании конструкции if else. |
| Возвращение словаря | Return\_slovar\_10 | Функции могут возвращать любые типы данных. Они вполне эффективно применяются и для обработки словаря и его последующего изменения. |
| Использование конструкции while | while\_return\_slovar\_11 | Также функцию можно вызывать и в конструкции while. Так, к примеру, делают для процесса обучения нейросетей. |
| Использование словаря в качестве аргумента | slovar\_na\_vhod\_12 | Попробуем использовать словарь в качестве аргумента. В качестве реального аргумента мы можем использовать считывание данных из базы данных, в которых все записано по столбикам id Имя Почта |
| Иммутабельность данных | immutable\_data\_13 | Пора немного поговорить о вечном. Мы говорим, что в многие программы в функциональной парадигме разделяют свои данные на мутабельные и немутабельные. Мутабельные структуры данных в Python это списки( мутабельные это те, которые могут изменяться), а немутабельные – это кортежи, в которых значения не могут изменяться. Чаще всего кортежи могут применяться в том случае, если вы не хотите, чтобы ваши данные вдруг изменились во время исполнения программы. |