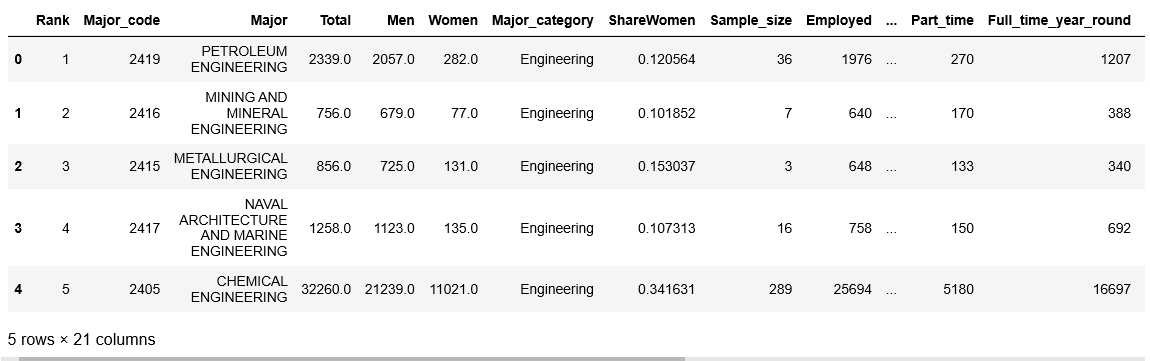
**Визуализация данных в Pandas**

Проанализируем данные по специальностям выпускников колледжей, полученные в результате исследования, которое послужила основой для гида по выбору колледжа.

* **Rank** - Рейтинг по медианному доходу (набор данных упорядочен по этому столбцу)
* **Major\_code** – Код специальности
* **Major** - Специальность
* **Major\_category** - Категория
* **Total** – Количество человек
* **Men** – Впускники мужчины
* **Women** – Выпускники женщины
* **ShareWomen** - Женщины как доля от общего числа
* **Employed** -. Количество занятых
* **Median** - Средняя заработная плата работников, занятых полный рабочий день и работающих круглый год.
* **Low\_wage\_jobs** - Количество низкооплачиваемых рабочих мест в сфере услуг.
* **Full\_time** - Количество занятых 35 часов и более
* **Part\_time** - Количество занятых менее 35 часов

После вызова read\_csv() создается DataFrame — главная структура данных, используемая в pandas. Отобразим первые пять строк.



Магическая команда %matplotlib настраивает Jupyter Notebook для отображения графиков с помощью [Matplotlib](https://python-scripts.com/matplotlib). По умолчанию используется стандартный графический бэкенд от Matplotlib, и ваши графики отображаются в отдельном окне.

## Создание простого Pandas графика в Python

Рассматриваемый набор данных содержит несколько столбцов, связанных с доходами выпускников по каждой специальности:

* **«Median»** — средний заработок работников, которые заняты полным рабочим днем, круглый год;
* **«P25th»** — 25-й процентиль заработка;
* **«P75th»** — 75-й процентиль заработка;
* **«Rank»** — рейтинг специалиста по среднему заработку.

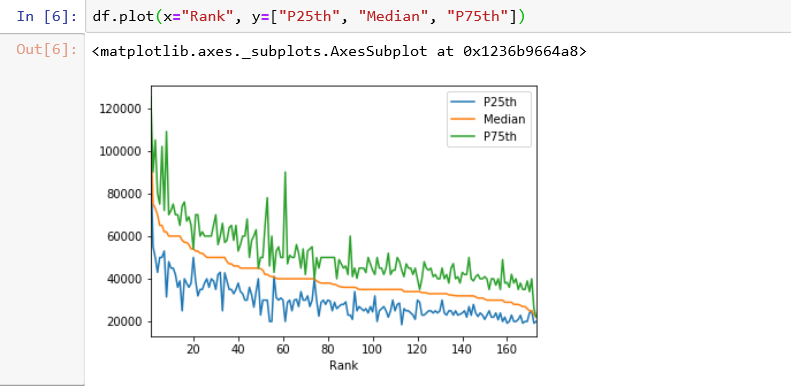
Начнем с графика, который отображает эти столбцы. Сначала вам потребуется настроить Jupyter Notebook для отображения графиков с помощью магической команды %matplotlib:

**На заметку**: Вы можете изменить бэкенд Matplotlib, передав аргумент в магическую команду %matplotlib.

К примеру, бэкенд inline популярен для Jupyter Notebooks, потому что он отображает график в самом блокноте сразу под ячейкой, которая создает график:



Теперь вы готовы к **созданию первого графика**. Это можно сделать с помощью метода .plot():



Метод .plot() возвращает линейный график, который содержит данные из каждой строки структуры DataFrame. Значения по оси х представляют рейтинг каждого учреждения, а значения "P25th", "Median" и "P75th" показываются на оси у графика.

При рассмотрении графика можно сделать следующие выводы:

* Средний заработок уменьшается по мере уменьшения рейтинга. Это ожидаемо, потому что рейтинг определяется средним доходом;
* У некоторых специальностей большой разрыв между 25-м и 75-м процентилями. Люди с этими степенями могут зарабатывать значительно меньше или значительно больше среднего дохода;
* У других специальностей очень небольшой разрыв между 25-м и 75-м процентилями. Люди с такими степенями получают зарплату, очень близкую к средней.

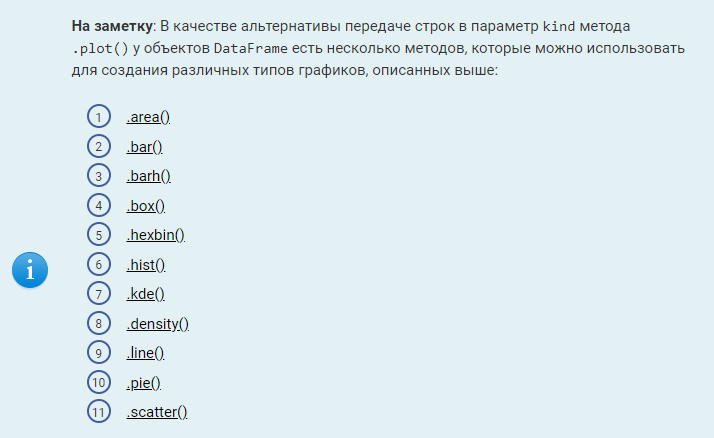
Первый график уже дает понять, что в датасетах можно найти много интересного. У некоторых специальностей есть широкий диапазон заработков, у других он довольно узкий. Для **обнаружения этих различий** будет использоваться другие типы графиков.

У метода .plot() есть несколько необязательных [параметров](https://python-scripts.com/def-args-kwargs). В частности, параметр kind принимает одиннадцать различных строковых значений и определяет, какой тип графика создается:

1. **area** для графиков с накоплением;
2. **bar** для вертикальной гистограммы;
3. **barh** для горизонтальной гистограммы;
4. **box** для графиков с боксами;
5. **hexbin** для шестнадцатеричных графиков;
6. **hist** для гистограмм;
7. **kde** для графика оценки плотности ядра;
8. **density** является альтернативным названием для kde;
9. **line** для линейных графиков;
10. **pie** для круговых графиков;
11. **scatter** для графиков рассеяния.

Тип графика по умолчанию является line. Линейные графики наподобие того, что был создан выше, дают хороший обзор данных. Их можно использовать для обнаружения общих трендов. С их помощью не получится сделать глубокий анализ, однако они могут помочь выяснить, на какую область стоит обратить внимание.

Если вы не укажите параметр для метода .plot(), она создаст линейный график с индексом по оси x и всеми числовыми столбцами по оси y. Данный тип графика подходит для наборов данных с несколькими столбцами, но для нашего набора данных по специальностям колледжа с его несколькими числовыми столбцами это выглядит довольно беспорядочно.



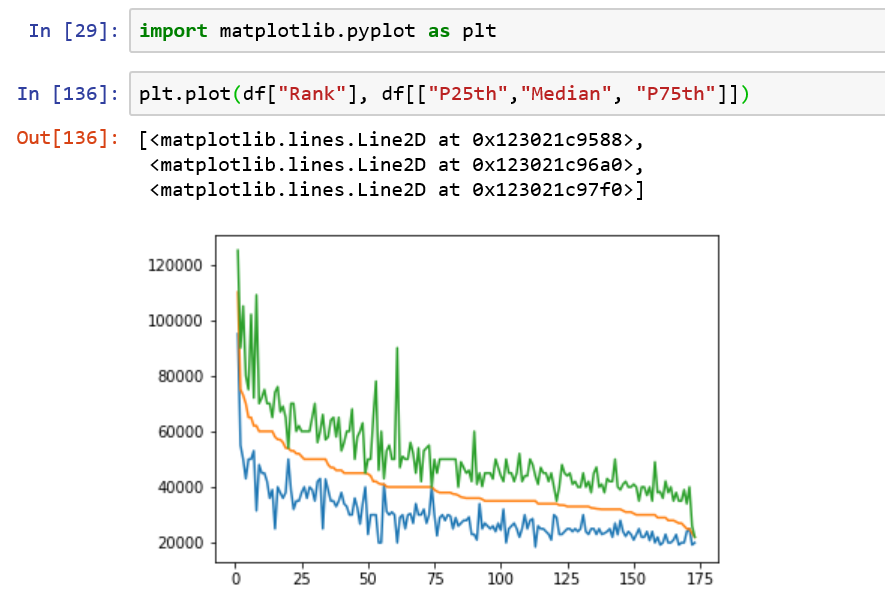
Мы будем использоваться интерфейс .plot() и передавать строки параметру kind. Однако при желании вы можете посмотреть сами, как работают вышеуказанные методы.

Теперь после создания нашего первого **pandas графика** можно подробнее разобрать, как именно работает [метод](https://python-scripts.com/object-oriented-programming-in-python#methods) .plot().

## Библиотека Matplotlib в Python

При вызове метода .plot() для объекта DataFrame Matplotlib незаметно создает наш график.

Чтобы убедиться в этом, воспользуемся двумя фрагментами кода. Сначала создадим график с помощью Matplotlib, используя два столбца из структуры DataFrame:



Сначала импортируется модуль matplotlib.pyplot и переименовывается как plt. Затем вызывается метод plot() и передается столбец "Rank" объекта DataFrame как первый аргумент, а также столбец "P75th" как второй аргумент.

Результатом является линейный график, на котором отображается 75-й процентиль по оси у против рейтинга по оси х.

Сначала импортируется модуль matplotlib.pyplot и переименовывается как plt. Затем вызывается метод plot() и передается столбец "Rank" объекта DataFrame как первый аргумент, а также столбец "P75th" как второй аргумент.

Результатом является линейный график, на котором отображается 75-й процентиль по оси у против рейтинга по оси х.

Теперь, когда вы знаете, что метод .plot() объекта DataFrame является оболочкой для pyplot.plot() от библиотеки Matplotlib, давайте рассмотрим различные типы графиков и способы их создания.

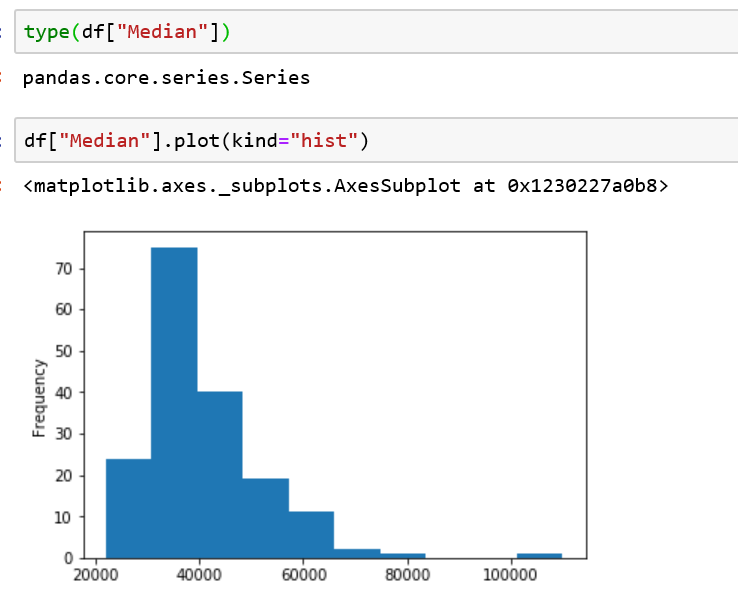
## **Изучение данных с помощью графиков в Python**

Следующие графики дадут общий обзор определенного столбца указанного набора данных. Сначала мы рассмотрим распределение собственности с помощью гистограммы. Затем мы познакомимся с некоторыми инструментами для исследования выбросов.

### Распределения и гистограммы в Pandas

DataFrame не единственный [класс](https://python-scripts.com/python-class) в pandas с методом .plot(). Часто встречаемый в pandas объект [Series](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html) предоставляет похожую функциональность.

Вы можете представить каждый столбец из **DataFrame** как объект **Series**. Далее дан пример использования столбца "Median" из структуры DataFrame, созданной на основе данных специальностей колледжей:



Гистограмма является хорошим способом визуализировать, как значения распределяются по набору данных. Гистограммы разбивают значения на интервалы (**bins**) и отображают количество данных, чьи значения находятся в определенном интервале.

Гистограмма показывает данные в десяти интервалах от $20,000 до $120,000, и ширина каждого интервала составляет $10,000. Форма гистограммы отличается от [нормального распределения](https://en.wikipedia.org/wiki/Normal_distribution), у которого симметричная форма колокола с пиком посередине.

Однако у гистограммы средних данных есть пики слева ниже $40,000. Хвост тянется далеко вправо и сообщает о том, что есть отрасли, в которых определенные специальности могут рассчитывать на более высокие доходы.

## **Выбросы в гистограмме**

Вы наверняка заметили одинокий маленький прямоугольник с правой части распределения? Кажется, что у некоторых данных есть своя собственная категория. Специалисты в данной области получают отличную зарплату по сравнению не только со средней зарплатой, но и с зарплатой, занявшей второе место. Хотя это не ее основная цель, гистограмма может помочь обнаружить такой выброс. Рассмотрим данный выброс более подробно:

* Какие специальности представляет данный выброс?
* Какова его граница?

В отличие от первого обзора, здесь нужно сравнить только несколько пунктов с данными, но получить о них более подробную информацию. Для этого гистограмма является отличным инструментом. Сначала выбираем пять специальностей с самым высоким средним доходом. Вам предстоит выполнить два пункта:

1. Сортировка столбца "Median" путем использование метода [.sort\_values()](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html), указывая название столбца, который нужно отсортировать, а также порядок сортировки ascending=False;
2. Получение пяти первых элементов списка через использования метода .head().

Создаем новую структуру DataFrame под названием top\_5:



Теперь у вас есть структура **DataFrame** меньшего размера, содержащая только пять самых прибыльных специальностей. В качестве следующего шага вы можете создать график, на котором будут показаны только основные специалисты с пятью самыми высокими средними зарплатами:

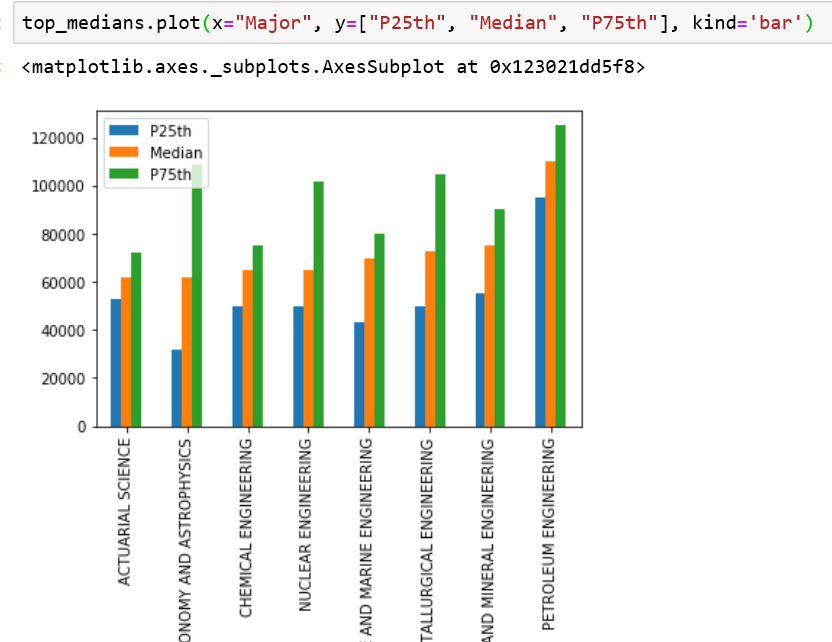
Обратите внимание, что параметры rot и fontsize используются для вращения и изменения размера ярлыков оси x, чтобы они были видны. Вы увидите график с 5 полосами.

Этот график показывает, что средняя заработная плата специалистов по нефтяной инженерии более чем на $20,000 выше, чем у остальных. Доходы специальностей, занимающих второе-четвертое места, относительно близки друг к другу.

Если у вас есть точка данных с более высокими или низкими значениями, чем остальные, вы, вероятно, захотите изучить ее подробнее. Например, вы можете посмотреть столбцы, содержащие связанные между собой данные.

Рассмотрим все специальности, чья средняя зарплата превышает $60,000. Сначала необходимо отфильтровать основные категории с помощью маски df[df["Median"] > 60000]. Затем вы можете создать еще одну гистограмму, показывающую все три столбца заработков:



25-й и 75-й процентили подтверждают вышесказанное: выпускники по специальности нефтяной инженерии получали больше всего.

Почему нас интересуют выбросы в этом наборе данных? Студенты при выборе специальности отталкиваются от определенных причин. Однако выбросы также очень интересны с точки зрения анализа. Они могут указывать не только на прибыльные отрасли, но и на **поврежденные данные**.

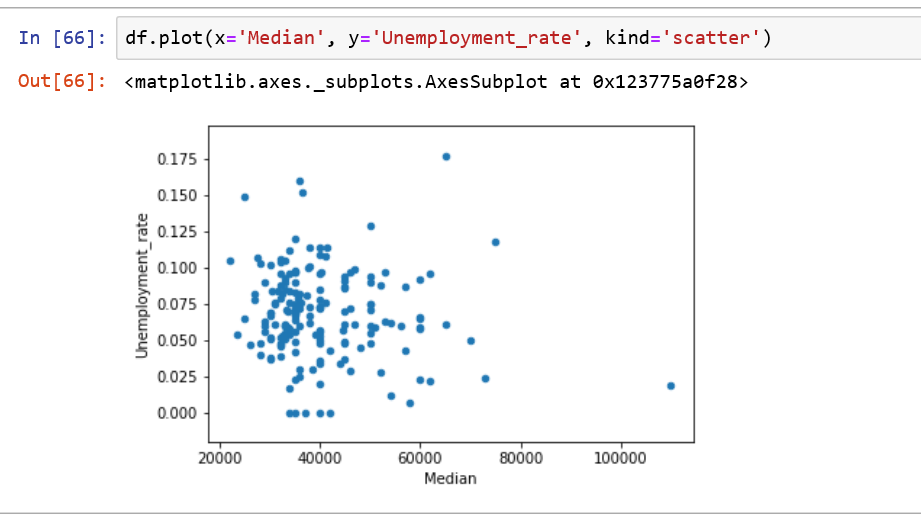
Поврежденные данные могут быть вызваны какими либо техническими ошибками или человеческим фактором, такие как отказ датчика, ошибка при ручном вводе данных или участие пятилетнего ребенка в фокус-группе, предназначенной для детей в возрасте от десяти лет и старше.

Даже если данные верны, вам может показаться, что они сильно отличаются от остальных и создают больше шума, чем пользы. Предположим, вы анализируете данные о продажах небольшого издателя. Вы группируете доходы по регионам и сравниваете их с тем же месяцем предыдущего года. Затем совершенно неожиданно издатель выпускает бестселлер.

Такая приятная неожиданность делает ваш отчет бесполезным. С учетом данных о бестселлерах продажи повсюду растут. Выполнение того же анализа без выброса даст более ценную информацию, позволяющую увидеть, что в Нью-Йорке показатели продаж значительно улучшились, но в Майами они ухудшились.

## **Проверка корреляции данных**

Зачастую требуется посмотреть, связаны ли два столбца с наборами данных. Если вы выберете специализацию с более высоким средним заработком, будет ли у вас меньше шансов остаться без работы? На первом этапе создайте **диаграмму рассеяния** с этими двумя столбцами:



Беглый взгляд на эту фигуру показывает, что нет значительной корреляции между заработком и уровнем безработицы.

Диаграмма рассеяния — отличный инструмент для получения первого впечатления о возможной корреляции, но не является окончательным доказательством наличия связи между данными. Для обзора корреляций между разными столбцами вы можете использовать метод [.corr()](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html). Если вы предполагаете, что есть корреляция между двумя значениями, тогда в вашем распоряжении есть [несколько инструментов](https://realpython.com/numpy-scipy-pandas-correlation-python/), чтобы проверить свою догадку и измерить, насколько сильна корреляция.

Имейте в виду, что даже если между двумя значениями существует корреляция, это еще не означает, что изменение одного значения приведет к изменению другого. Другими словами, корреляция не предполагает причинной связи.

## **Анализ категориальных данных**

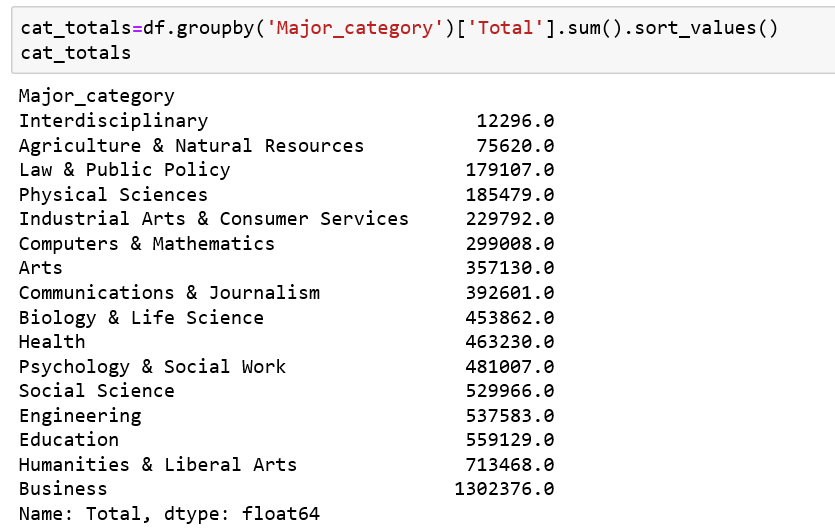
Для обработки больших объемов информации, человеческий разум сознательно и бессознательно сортирует данные по категориям. Этот прием часто бывает полезным, однако он далеко не безупречный.

Иногда мы делим объекты по категориям, которые при внимательном рассмотрении не так уж похожи. В этом разделе вы познакомитесь с некоторыми инструментами для изучения категорий и проверки их целесообразности.

Многие наборы данных уже содержат явную или неявную категоризацию. В текущем примере 173 специализации разделены на 16 категорий.

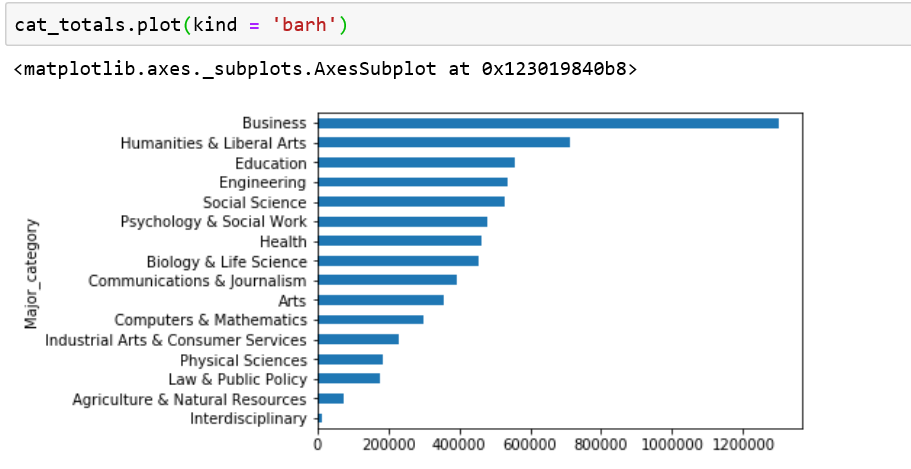
## **Группировка данных в Pandas**

Категории в основном используются для группирования и агрегирования данных. Вы можете использовать метод .groupby(), чтобы определить, насколько популярна каждая из категорий из набора специализаций:



С помощью метода [.groupby()](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html) создается объект DataFrameGroupBy, а с помощью метода [.sum()](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.core.groupby.GroupBy.sum.html) вы создадите Series.

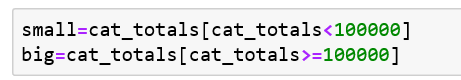
Давайте нарисуем график с горизонтальными полосами, которые будут представлять все категории из cat\_totals:

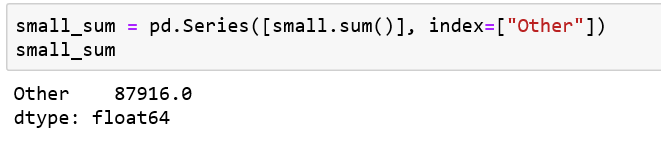


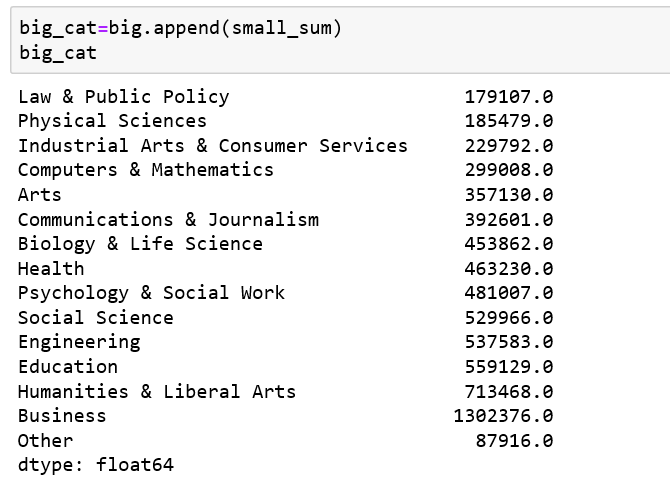
## **Определение соотношений на графике**

Вертикальные и горизонтальные гистограммы часто являются хорошим выбором, если вы хотите увидеть разницу между категориями. Если вас интересуют соотношение данных, то круговые диаграммы — отличный инструмент. Поскольку cat\_totals содержит только несколько категорий, создание круговой диаграммы с помощью cat\_totals.plot(kind="pie") приведет к созданию нескольких крошечных фрагментов с перекрывающимися текстовыми ярлыками.

Для решения этой проблемы можно объединить более мелкие категории в одну группу. Объединим все категории с общим количеством данных меньше 100 000 в категорию под названием «Other» («Другое»), затем создадим круговую диаграмму:

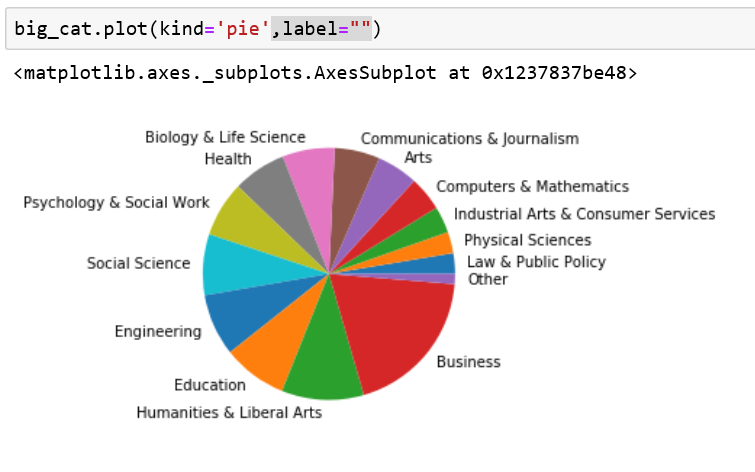






Обратите внимание, что мы добавили аргумент label="". По умолчанию, pandas добавляет ярлык с названием столбца. Это зачастую имеет смысл, но в данном случае это будет лишним.

Теперь вы должны увидеть круговой график наподобие следующего:



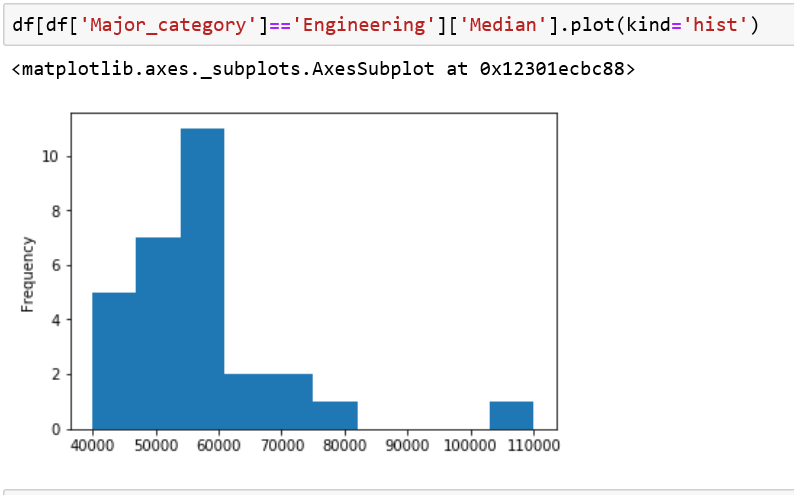
Категория "Other" по-прежнему небольшая. Это хороший признак, значит, мы сделали правильный выбор объединив маленькие категории.

### **Анализ данных внутри категории**

Иногда нужно проверить, имеет ли смысл определенная категоризация. Являются ли элементы категории более похожими друг на друга, чем на остальную часть набора данных?

Опять же, распределение — хороший инструмент для первого обзора нашего набора данных. Как правило, мы ожидаем, что распределение категории будет похоже на [нормальное распределение](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), но с меньшим диапазоном.

Создадим гистограмму, показывающую распределение среднего дохода для инженерных специальностей:



Диапазон основных средних доходов немного меньше, начиная с $40,000. Распределение ближе к нормальному, хотя его пик все же слева. Итак, даже если вы решили выбрать специальность в инженерной категории, было бы разумно более тщательно проанализировать все варианты.

### **Заключение**

В этом руководстве вы узнали, как визуализировать набор данных с помощью Python и библиотеки pandas. Вы увидели, как некоторые базовые графики могут дать представление о данных и помочь выбрать направление для анализа.

В этой теме вы узнали, как:

* Сделать анализ распределения набора данных с помощью гистограммы;
* Найти корреляцию с помощью диаграммы разброса;
* Анализировать категории с помощью гистограмм и круговых диаграмм;
* Определить, какой график больше всего подходит для текущей задачи.

Используя метод .plot() и структуру DataFrame, вы обнаружили много возможностей для визуализации данных.