Задание 1.
Исходные данные:
Даны значения величины заработной платы заемщиков банка (zp) и значения их поведенческого кредитного скоринга (ks): zp = [35, 45, 190, 200, 40, 70, 54, 150, 120, 110], ks = [401, 574, 874, 919, 459, 739, 653, 902, 746, 832]. Используя математические операции, посчитать коэффициенты линейной регрессии, приняв за X заработную плату (то есть, zp - признак), а за y - значения скорингового балла (то есть, ks - целевая переменная). Произвести расчет как с использованием intercept, так и без.
Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x1 = np.array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])
>>> y1 = np.array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])
>>> b1 = np.cov(x1, y1, ddof=1)[0, 1] / np.var(x1, ddof=1)
>>> b0 = y1.mean() - b1 * x1.mean()
>>> print(b0, b1)
444.1773573243596 2.620538882402765
>>>
>>> x1
array([ 35,  45, 190, 200,  40,  70,  54, 150, 120, 110])
>>>
>>> z1 = b0 + b1 * x1
>>> print(z1)
[535.89621821 562.10160703 942.07974498 968.2851338  548.99891262
 627.61507909 585.68645697 837.25818968 758.64202321 732.43663439]
>>>
>>> e1 = y1 - z1
>>> print(e1)
[-134.89621821   11.89839297  -68.07974498  -49.2851338   -89.99891262
  111.38492091   67.31354303   64.74181032  -12.64202321   99.56336561]
>>> e1.mean()
2.2737367544323207e-14
>>>
>>> n = x1.shape[0]
>>> m = 1
>>> k1 = m
>>> k2 = n - m - 1
>>> print(k1, k2)
1 8
>>>
>>> alpha = 0.05
>>> import scipy
>>> from scipy import stats
>>> t = stats.f.ppf(1 - alpha / 2, k1, k2)
>>> print(t)
7.57088209969174
>>>
>>> def sum_of_squares(samples: np.ndarray) -> float:
...     return ((samples - samples.mean()) ** 2).sum()
...
```

```
>>> r1 = sum_of_squares(e1) / sum_of_squares(y1)
>>> r1
0.2123613364706314
>>>
>>> f = (r1 / k1) / ((1 - r1) / k2)
>>> print(f)
2.1569417176048833
>>>

Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import scipy
>>> x1 = np.array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])
>>> y1 = np.array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])
>>> b1 = np.cov(x1, y1, ddof=1)[0, 1] / np.var(x1, ddof=1)
>>> b0 = y1.mean() - b1 * x1.mean()
>>> print(b0, b1)
444.1773573243596 2.620538882402765
>>>
>>> x1
array([ 35,  45, 190, 200,  40,  70,  54, 150, 120, 110])
>>>
>>> z1 = b0 + b1 * x1
>>> print(z1)
[535.89621821 562.10160703 942.07974498 968.2851338  548.99891262
 627.61507909 585.68645697 837.25818968 758.64202321 732.43663439]
>>>
>>> e1 = y1 - z1
>>> print(e1)
[-134.89621821   11.89839297  -68.07974498  -49.2851338   -89.99891262
  111.38492091   67.31354303   64.74181032  -12.64202321   99.56336561]
>>>
>>> e1.mean()
2.2737367544323207e-14
>>>
>>> def standard_error_slope(
...         x1: np.ndarray,
...         y1: np.ndarray,
...         z1: np.ndarray,
... ) -> float:
...     n = x1.shape
...     upper = ((y1 - z1) ** 2).sum()
...     lower = ((x1 - x1.mean()) ** 2).sum()
...     return np.sqrt(upper / lower)
...
>>> s_slope = standard_error_slope(x1, y1, z1)
>>> s_slope
1.360707628564636
>>>
>>> alpha = 0.05
```

```
>>> n = x1.shape[0]
>>> from scipy import stats
>>> t1 = stats.t.ppf(alpha / 2, df=n - 2)
>>> t2 = stats.t.ppf(1 - alpha / 2, df=n - 2)
>>> print(t1, t2)
-2.306004135033371 2.3060041350333704
>>>
>>> b1_lower, b1_upper = (b1 + t1 * s_slope, b1 + t2 * s_slope)
>>> b1_lower, b1_upper
(-0.5172585356387378, 5.758336300444267)
>>>
```

Задание 2.
Исходные данные:
В каких случаях для вычисления доверительных интервалов и проверки статистических гипотез используется таблица значений функции Лапласа, а в каких - таблица критических точек распределения Стьюдента?
Решение:
Для начала формируем основную и дополнительную гипотезы. Далее начинаем их проверять.
Метод Стьюдента, очень прост в использовании и кажется на первый взгляд, более удобным и если у нас мало статестических данных, то он нам очень хорошо подходит, из-за своей простоты, но если у нас датасет, то тут уже метод Стьюденса, будет давать уже большие погрешности. Здесь уже хорошо подойдёт метод Лапласа, так как он прямо идеально создан, для датасетов и именно он в данной ситуации даст наименьшую погрешность или ошибку, что даст возможность наиболее точнее определить данные, которые содержаться в датасете.