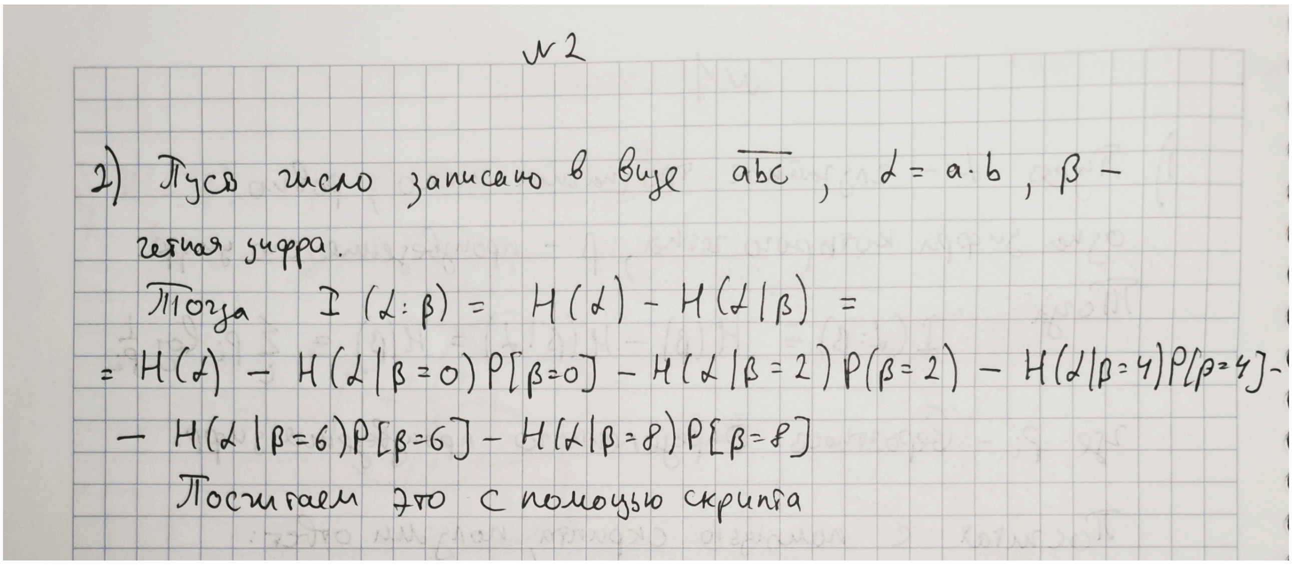


№2

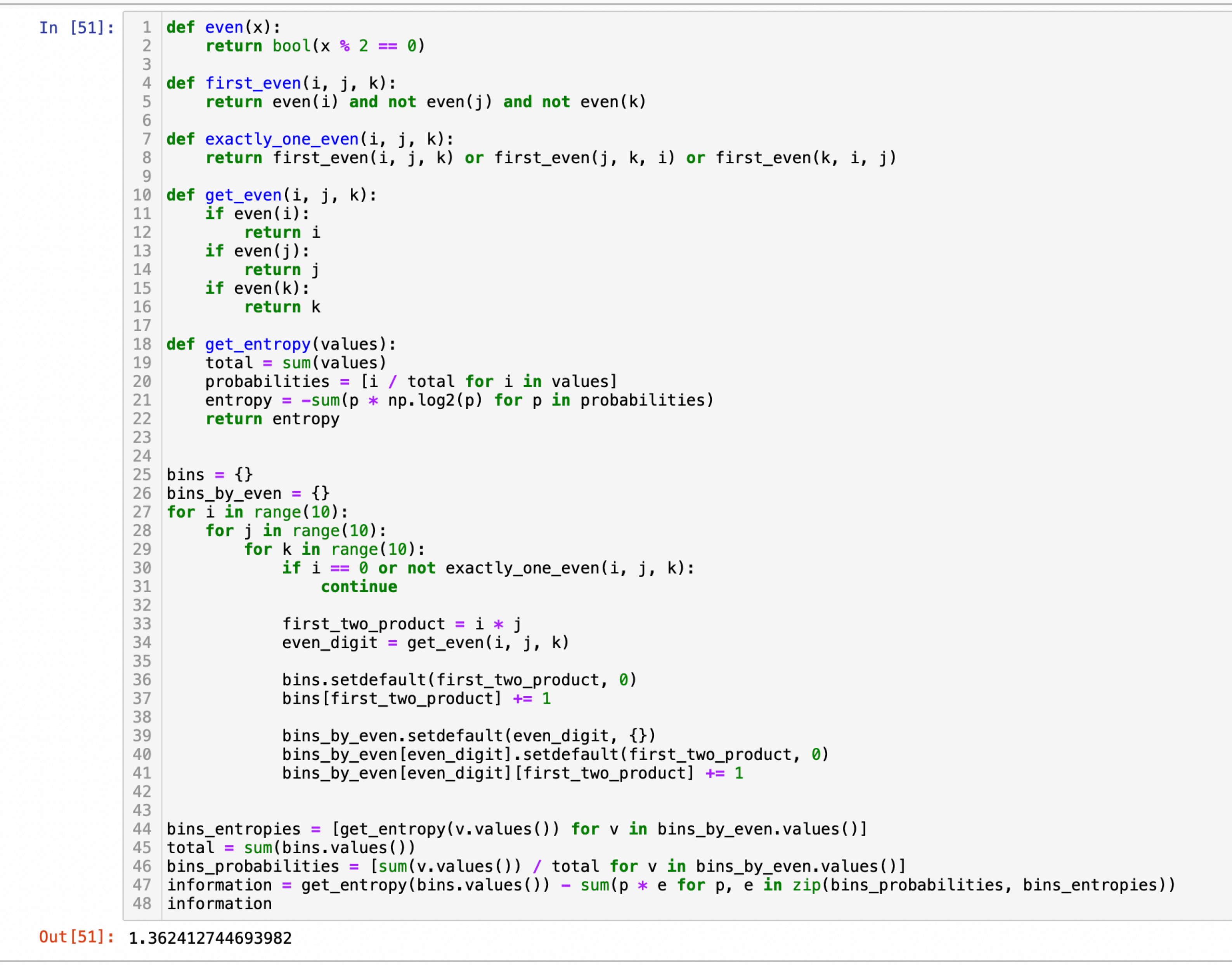
2) Пусть число записано в виде $\overline{abc}$, $\alpha = a \cdot b$, β – четная цифра.

Тогда $I(\alpha : \beta) = H(\alpha) - H(\alpha | \beta) =$

$= H(\alpha) - H(\alpha | \beta = 0) P[\beta = 0] - H(\alpha | \beta = 2) P(\beta = 2) - H(\alpha | \beta = 4) P[\beta = 4] -$

$- H(\alpha | \beta = 6) P[\beta = 6] - H(\alpha | \beta = 8) P[\beta = 8]$

Посчитаем это с помощью скрипта



In [51]:

```python
def even(x):
    return bool(x % 2 == 0)

def first_even(i, j, k):
    return even(i) and not even(j) and not even(k)

def exactly_one_even(i, j, k):
    return first_even(i, j, k) or first_even(j, k, i) or first_even(k, i, j)

def get_even(i, j, k):
    if even(i):
        return i
    if even(j):
        return j
    if even(k):
        return k

def get_entropy(values):
    total = sum(values)
    probabilities = [i / total for i in values]
    entropy = -sum(p * np.log2(p) for p in probabilities)
    return entropy


bins = {}
bins_by_even = {}
for i in range(10):
    for j in range(10):
        for k in range(10):
            if i == 0 or not exactly_one_even(i, j, k):
                continue

            first_two_product = i * j
            even_digit = get_even(i, j, k)

            bins.setdefault(first_two_product, 0)
            bins[first_two_product] += 1

            bins_by_even.setdefault(even_digit, {})
            bins_by_even[even_digit].setdefault(first_two_product, 0)
            bins_by_even[even_digit][first_two_product] += 1


bins_entropies = [get_entropy(v.values()) for v in bins_by_even.values()]
total = sum(bins.values())
bins_probabilities = [sum(v.values()) / total for v in bins_by_even.values()]
information = get_entropy(bins.values()) - sum(p * e for p, e in zip(bins_probabilities, bins_entropies))
information
```

Out[51]:

```
1.362412744693982
```