

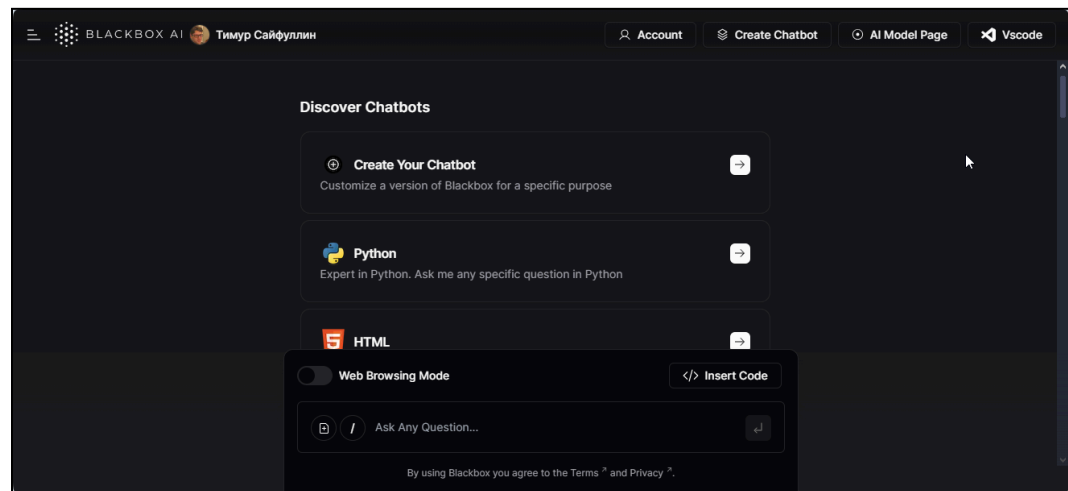
## **Бонус 3. ПРОГРАММИРОВАНИЕ С ПОМОЩЬЮ AI ИНСТРУМЕНТОВ**

<b>ЗНАКОМСТВО С НОВЫМИ ИНСТРУМЕНТАМИ</b>	<b>2</b>
<b>ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ</b>	<b>5</b>
<b>ОШИБКИ AI</b>	<b>7</b>
<b>ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ</b>	<b>10</b>
<b>РЕШЕНИЕ КОМПЛЕКСНОЙ ЗАДАЧИ</b>	<b>11</b>

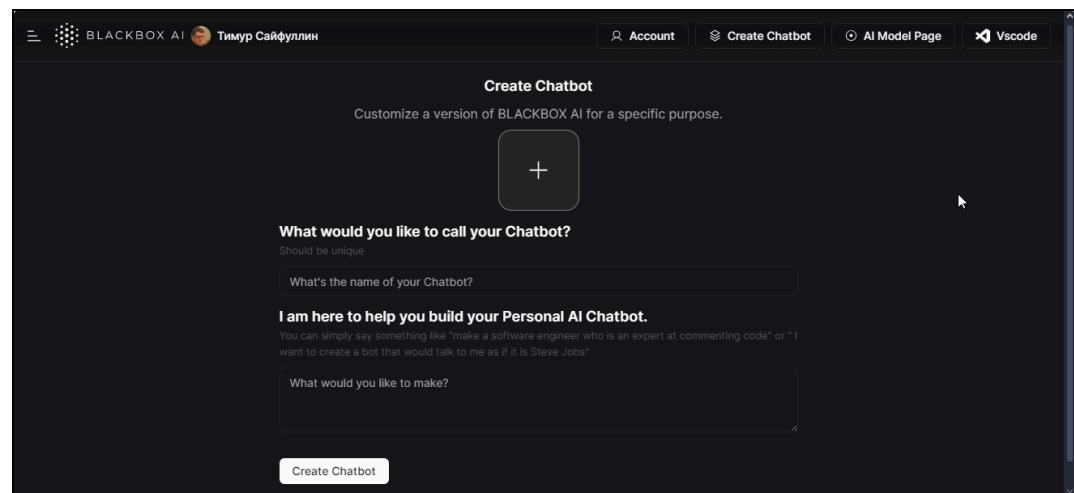


## ЗНАКОМСТВО С НОВЫМИ ИНСТРУМЕНТАМИ

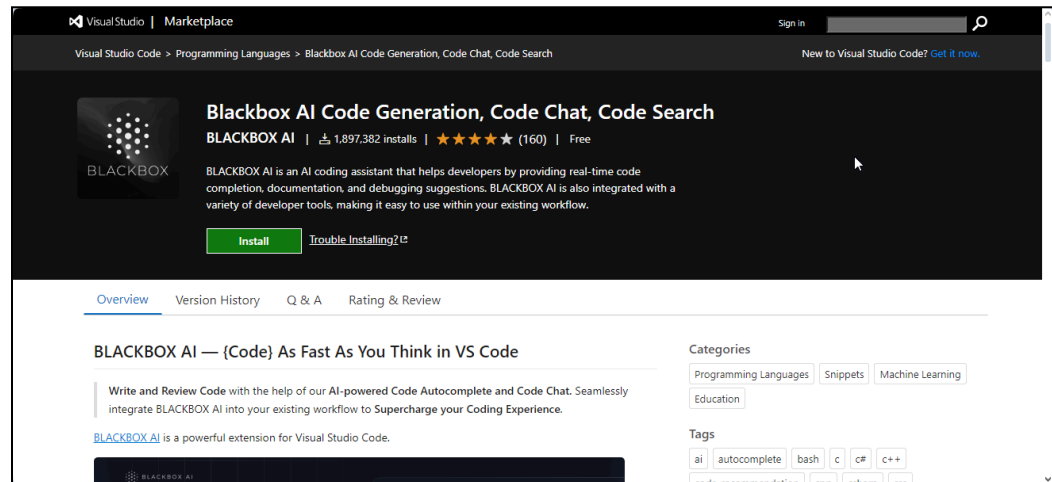
1. ChatGPT - [ChatGPT](#)
2. Blackbox - [Blackbox: AI](#)
  - а. Главный экран, где можно формировать запрос или использовать готовых чатботов под разные тематики



- б. Возможность создать своего чатбота

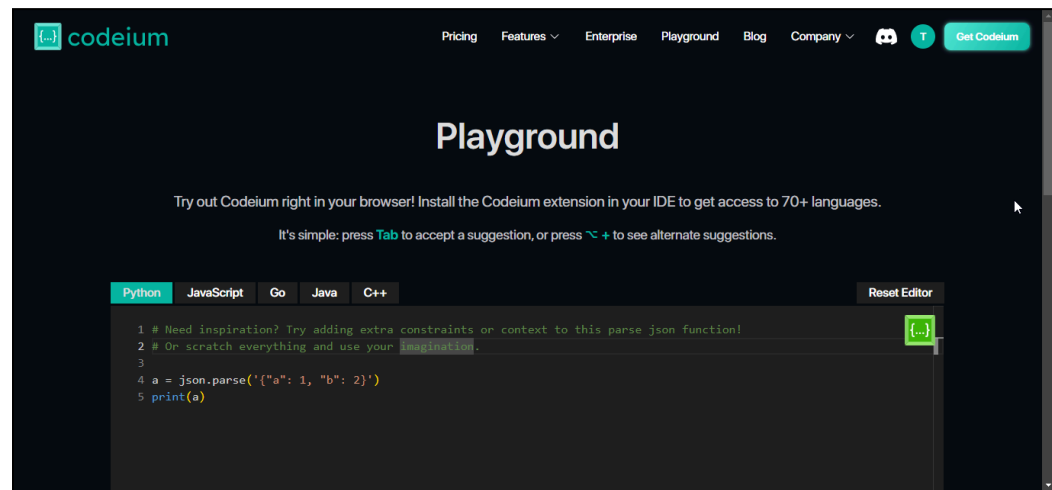


- с. Возможность встроить AI помощника в excode



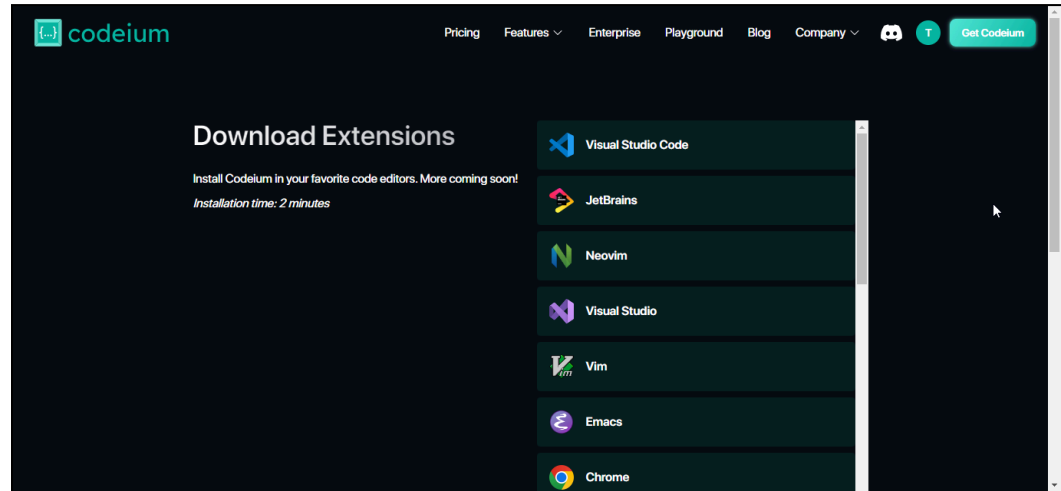
### 3. Codeium - [Chrome Tutorial](#) | [Codeium · Free AI Code Completion & Chat](#)

#### а. Возможность писать код в браузере





b. Возможность установить в популярный IDE



c. Возможность установить AI как расширение в браузере и использовать в Jupyter lab или Colab





## ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

Валидация домашней работы:

возьмем код из вашей домашней работы и проверим правильно ли он написан, используя ChatGPT.

1.

import re

Python

```
if __name__ == '__main__':
    text = "br@ya.com Contact us at info@example.com or support@example.com for assistance."
    pattern = r"(?:\b|^)\w*\w*\.\w*\b"
    arrayEmail = re.findall(pattern, text, re.I)
    print(arrayEmail)
```

2.

import re

Python

```
def highlight_keywords(text_find, keywords_find):
    for keyword in keywords_find:
        pattern = re.escape(keyword)
        info_find = re.findall(pattern, text_find, re.I)

        for data in info_find:
            text_find = text_find.replace(data, "*" + data + "*")

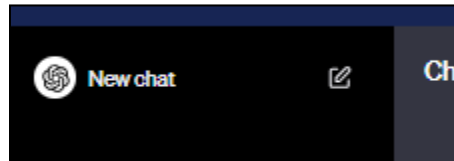
    return text_find

if __name__ == '__main__':
    text = "This is a sample text. We need to highlight Python and programming."
```



```
keywords = ["this", "python", "programming"]  
highlighted_text = highlight_keywords(text, keywords)  
  
print(highlighted_text)
```

Теперь давайте попросим ChatGPT решить задачу с нуля. Для этого возьмем условие задания и попросим ChatGPT написать код. Однако мы создадим новый New Chat, так как наши старые запросы могут повлиять на будущее.



1) Напишите функцию `extract_emails(text)`, которая извлекает все адреса электронной почты из заданного текста и возвращает их в виде списка.

Пример использования:

```
text = "Contact us at info@example.com or support@example.com for assistance."  
emails = extract_emails(text)  
print(emails) # Вывод: ['info@example.com', 'support@example.com']
```

2) Напишите функцию `highlight_keywords(text, keywords)`, которая выделяет все вхождения заданных ключевых слов в тексте, окружая их символами `*`. Функция должна быть регистронезависимой при поиске ключевых слов.

Пример использования:

```
text = "This is a sample text. We need to highlight Python and programming."  
keywords = ["python", "programming"]  
highlighted_text = highlight_keywords(text, keywords)  
print(highlighted_text)
```

```
# Вывод: "This is a sample text. We need to highlight *Python* and *programming*."
```



## ОШИБКИ AI

Несмотря на то, что ChatGPT может решать наши задачи, хоть и не с первого раза. Он может допускать ошибки как и человек. Мы можем получить решение и даже не заметить этого. Давайте приведем пример.

Есть задание:

На вход дается число  $n$  и число  $d$ , где  $n$  - это просто последовательность чисел, а  $d$  - это делитель. Нам нужно вернуть произведение всех цифр числа  $n$  кратных  $d$ . Если таких чисел нет, то вернуть  $-1$ .

Вот ответ, который выдает нам ChatGPT :

```
def произведение_кратных_цифр(n, d):
    # Преобразуем число n в строку для работы с отдельными цифрами
    str_n = str(n)

    # Инициализируем переменную для хранения произведения
    произведение = 1

    # Перебираем цифры числа n
    for цифра in str_n:
        # Проверяем, кратна ли цифра d
        if int(цифра) % d == 0:
            произведение *= int(цифра)

    # Если не было кратных цифр, возвращаем -1
    if произведение == 1:
        return -1
    else:
        return произведение

# Пример использования
n = 12345
d = 3
результат = произведение_кратных_цифр(n, d)
print(результат)
```

There was an error generating a response



Python

```
def произведение_кратных_цифр(n, d):  
    # Преобразуем число n в строку для работы с отдельными цифрами  
    str_n = str(n)  
  
    # Инициализируем переменную для хранения произведения  
    произведение = 1  
  
    # Перебираем цифры числа n  
    for цифра in str_n:  
        # Проверяем, кратна ли цифра d  
        if int(цифра) % d == 0:  
            произведение *= int(цифра)  
  
    # Если не было кратных цифр, возвращаем -1  
    if произведение == 1:  
        return -1  
    else:  
        return произведение  
  
# Пример использования  
n = 12345  
d = 3  
результат = произведение_кратных_цифр(n, d)  
print(результат)
```

Заметим, что при  $n = 11111$  и  $d = -1$ , мы должны получать 1, но получаем -1. И при  $d = -1$  мы тоже получаем -1

Вот тут ChatGPT нас подвел.

Плюс код представленный ChatGPT недостаточно оптимизирован, так как использует строки и занимает много памяти. Лучшим решением будет не использовать строки, а брать последнее число из  $n$  (через  $\%$ ) и смещать через  $//$

Часть кода:





```
number = n
while number > 0:
    digit = number % 10 # Получаем последнюю цифру числа
    number //= 10       # Убираем последнюю цифру числа
```

Теперь решим ту же задачку через Blackbox - (тут он решает верно).

Python

```
def product_of_multiples(n, d):
    """Return the product of all digits of n that are multiples of d, or -1 if
    there are no such digits."""
    product = 1
    found = False
    while n > 0:
        digit = n % 10
        if digit % d == 0:
            product *= digit
            found = True
        n //= 10
    return product if found else -1
```



## ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

Попробуйте решить, используя инструменты AI:

1. Есть набор чисел - 9,3,7. И целевое значение 13  
Нужно написать алгоритм который скажет можно ли из набора чисел получить 13, используя + и -  
Например,  $9+3+7$  не равно 13  
 $9-3+7$  равно 13
2. Напишите программу, которая выводит часть последовательности 1 2 2 3 3 4 4 4 5 5 5 5 5 ... (число повторяется столько раз, чему равно). На вход программе передаётся неотрицательное целое число  $n$  — столько элементов последовательности должна отобразить программа. На выходе ожидается последовательность чисел, записанных через пробел в одну строку.  
Например, если  $n = 7$ , то программа должна вывести 1 2 2 3 3 4.
3. Попросить ai решить задачу используя 5 разных способов:  
  
Даются два списка. Нужно избавить первый список от элементов присутствующих во втором, сохраняя дубликаты и порядок.



## РЕШЕНИЕ КОМПЛЕКСНОЙ ЗАДАЧИ

Давайте теперь решим комплексную задачу с помощью ChatGPT.

Нам необходимо выстроить полноценный ETL процесс. Сначала нужно подключиться к базе данных, забрать данные, обработать их или трансформировать и загрузить в новую базу данных.

Поначалу непонятно, что нужно спросить у ChatGPT, так как мы не знаем вводных, какие данные к нам приходят, их тип данных или что мы должны получить в итоге. Поэтому давайте попросим ChatGPT описать пайплайн (последовательность шагов) решения данной задачи и подробно разберем каждый этап.

```
import pandas as pd

# Шаг извлечения данных (Extract)
def extract_data(file_path):
    # Чтение данных из CSV файла
    data = pd.read_csv(file_path)
    return data

# Шаг трансформации данных (Transform)
def transform_data(data):
    # Пример преобразования: удаление дубликатов
    transformed_data = data.drop_duplicates()
    return transformed_data

# Шаг загрузки данных (Load)
def load_data(data, target_table_name):
    # Пример загрузки данных в базу данных SQLite
    engine = create_engine('sqlite:///database.db')
    data.to_sql(target_table_name, engine, index=False, if_exists='replace')

# Задаем пути к источнику данных и целевой таблице
file_path = 'source_data.csv'
target_table_name = 'target_table'

# Вызываем каждый шаг ETL процесса
data = extract_data(file_path)
transformed_data = transform_data(data)
load_data(transformed_data, target_table_name)
```

ChatGPT может помочь разобраться с решением. Давайте вернемся к нашим задачам из пункта 5 и попросим ai подробно объяснить нам свое решение.



После этого давайте возьмем сложный код и попробуем понять, что он делает, используя ChatGPT:

Python

```
from pymle_master.pymle.models import CKLS, OrnsteinUhlenbeck
from pymle_master.pymle.TransitionDensity import EulerDensity, OzakiDensity
#ShojiOzakiDensity
from pymle_master.pymle.fit.AnalyticalMLE import AnalyticalMLE
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
from tqdm import tqdm
import matplotlib.ticker as ticker
from scipy.stats import norm
import scipy.stats as st
```

```
def ou(X, kappa, alpha, sigma, dW, dt):
    dX = kappa * (alpha - X) * dt + sigma * dW
    return dX
```

```
kappa = 3
alpha = 2
sigma = 0.5
```

```
N = 1000
delta = 1 / N
```

```
param_bounds = [(0, 8), (0, 4), (0.01, 1)]
guess = np.array([2, 1, 0.4])
```

```
model = OrnsteinUhlenbeck()
```

```
M = 50
param = []
```

```

theta1 = []
theta2 = []
theta3 = []
for j in range(M):
    dW = np.random.normal(0, np.sqrt(1 / N), N)
    X_ou = np.zeros(N)
    X_ou[0] = 5
    for i in range(1, N):
        X_ou[i] = X_ou[i - 1] + ou(X_ou[i - 1], kappa, alpha, sigma, dW[i], delta)
    param = AnalyticalMLE(X_ou, param_bounds, delta,
density=OzakiDensity(model)).estimate_params(guess)
    #print(param)
    theta1.append(param.params[0])
    theta2.append(param.params[1])
    theta3.append(param.params[2])

#print(param)
print(st.t.interval(alpha=0.95, df=len(theta1)-1, loc=np.mean(theta1),
scale=st.sem(theta1)) )
print(st.t.interval(alpha=0.95, df=len(theta2)-1, loc=np.mean(theta2),
scale=st.sem(theta2)) )
print(st.t.interval(alpha=0.95, df=len(theta3)-1, loc=np.mean(theta3),
scale=st.sem(theta3)) )

```