コード自動生成（CodeGeeX 他）

目次

コード自動生成（CodeGeeX 他）

はじめに

プログラミングに付いて回る決まりきったコードを減らす取り組みは汎用機（メインフレーム）のマクロアセンブラの時代からありました。よく使われるものの一つにに画面生成マクロがあり、項目毎に表示位置や入出力、文字種類、桁数等の固有情報を指定するだけで業務画面が完成します。

現代ではフレームワークや共用ライブラリを使うことが定番になり、制御（IO や判定、繰り返し）用のコードを削減する Java の Stream や C#の Linq、Python の内包表記、ラムダ式、…等の簡明な書き方を言語仕様に取り込み続けています。また、SQL クエリを使えば論理的な条件を記述するだけで制御コードを書かずにデータ間の関連付けや加工ができます。

次に向かおうとしているのは Auto Coding や Code Generator と呼ぶコードの自動生成です。これはインターネット上の大量の情報を学習データとし、大量の CPU/GPU とメモリ、ストレージを使って処理・作成した事前学習済言語モデルからコードを生成しようという新しいアプローチで、それまでのテンプレートに変動パラメータを埋め込むソース生成とは全く異なるものです。

1. コードの自動生成が生まれた経緯

2018 年に Google 社が自然言語処理[1]の論文 BERT(Bidirectional Encoder Representations from Transformers：Transformer を活用した双方向的エンコード表現)とその実装を発表したのを端緒に、2019 年に OpenAI 社が Transformer をベースにインターネット上の大量の情報で事前学習した言語モデル GPT-2(Generative Pre-trained Transformer-2)を公開して自然な会話ができる AI[2]として話題になりました。2020 年に次版の GPT-3 の独占的ライセンス契約を Microsoft 社と結び Microsoft 社はこれを使って無料の開発ツール CodeGPT（開発環境 VSCode の機能拡張）の提供を始めました。BERT も GPT もテキストを変換（Transform：翻訳、要約、他言語生成、...）のための言語モデルとしてつくられており、開発ツールとして使うためには変換前後のテキストを開発環境で取り扱う機能が必要で CodeGPT がこれを行います。2023 年 4 月現在 CodeGPT は以下の機能を持っています。

・GPT-4 へのアクセス
・ChatGPT とのチャット
・コメントからコードを生成
・ソースコードのリファクタリング
・ソースコードからコメントを生成
・その他

CodeGPT は無償で使えますが GPT-3 へのアクセスには OpenAI 社への登録が必要なため、色々な組織が Transformer ベースの事前学習済言語モデル（BERT、GPT 以外にも T５、DeBERTa 等々）と言語モデルを使うためのツールを競って開発しており Auto-Coding や Code-Genarator 等のタグを付けてオープンソースで公開しています。因みに GPT-2 もオープンソース（修正 MIT ライセンス）で公開[3]されています。

---

[1] 言語処理の方法は「bert gpt attention」等の文言で検索すると解説サイトが沢山でてきます
[2] GPT-2,GPT-3,GPT-4 は大量の学習データを使い言語データを整理した事前学習済言語モデルです
一般に AI と呼ばれている ChatGPT は GPT-3.5 に会話インタフェースを付加したものです
[3] GitHub/openai/gpt-2 https://github.com/openai/gpt-2

2. AI を使うリスク

AI 全般に関して SF 的な漠然とした不安を語る論調がありますが、コードの自動生成に関しては黎明期特有のもう少し具体的なリスクがあります。

2.1. 著作権等

コード生成は情報源としてコード生成用の言語モデルを使います。生成用の言語モデルは GitHub のリポジトリ等に登録されたソースコードから事前学習したものが多いのですが、各ソースには著作権があり、コードによっては特許申請・登録されたものが混じっているかもしれません。

日本の著作権法では学習データとして使う場合は著作権者の承諾は不要のよう[4]ですが、AI 利用の切っ掛けになった GPT-2 からは学習データの一部を抽出することができるという報告[5]があり、実際にこれを行うツールが GitHub で公開されています。また、自動生成ツールが提案したコードが事前学習データに使った商用禁止ライセンスのソースと完全一致した場合、偶然の一致と主張できるか否か（小さく分割されたトークンから問合せ内容に基づいて新たに組み立てているので問題なさそうな気はしますが...）、海外で運用・利用されているサイトのソースで作った言語モデルから自動生成したソースの著作権の帰属先がどう判定されるのか、現状では分かりません。

大半の言語モデルは事前学習に使用するソースのライセンスに言及していませんが、モデルによってはライセンスフリーのソースだけを使っていると宣言しているものもあります。

2.2. 漏洩

開発環境に組み込んでコードの自動補完を受けるためにエディターで編集中のコードの全体か一部を言語モデルに送りますが、送られたコードは言語処理をするために事前学習データと同一の機能を使ってエンコードされ言語モデルに取り込まれる（学習データとして使われる）可能性があります。ツールによっては取込みを停止することもできますが、IP アドレスを含むアクセス記録は残ります。

また、CodeGPT やその他の代替ツールはリファクタリングの機能を持っています。この機能を使うためには自分で開発しているソースを言語モデル側に送る必要がありますが、転送経路や送った先で保護される保証はありませんし、言語モデルに取り込まれる可能性は自動補完と同様です。

2.3. 生成されるコードの精度

コード生成で使われる言語モデルは各種のプログラミング言語用に調整（ファインチューニング）されていると推測しますが、特定のバージョンで追加されたり廃止されたクラスや記法をどのように適用するのか明示されたものは見当たりません。また、言語モデルは既に誰かが記述した蓄積を基にしたものなので、調整次第ですが、より新しい機能（クラス、メソッド）やアルゴリズムではなく最大公約数的なソースの提案になる可能性が強くなります。更に「ChatGPT に生成させたコードは生成する回答が間違っている可能性が高い」ので使用を当面禁止としたコミュニティもあります。

---

[4] 総務省「AI を用いたクラウドサービスに関するガイドブック」4.1.2 留意すべき事項 3）データ収集と著作権
https://www.soumu.go.jp/menu_news/s-news/01ryutsu06_02000305.html

[5] Extracting Training Data from Large Language Models https://arxiv.org/abs/2012.07805

3. コード生成の範囲やタイプ

コード生成／コード補正を行う各種のツールは、それぞれ以下のレベルを目標にしています。
① クラス全体を生成する
② Spring や FLASK、React 等のフレームワークに組み込み可能なコードを自動生成する
③ メソッドや処理ブロックを生成する
④ 入力中の文や式の補完し、次の文や式を提案する

①～③のようにある程度高度な機能を実現するにはツールに仕様を伝える方法が難しくなります。③は連続する複数行のコメントで実現している例もありますが、決まりきった書き方を覚える必要があるのであれば直接 コードを覚えた方が効率がよいかもしれません。

①、②が満足できるレベルで実現できるようになればプログラミング言語を習得する必要がなくなり是非実現して貰いたいですが、現状は④のレベルが現実的で、しかも言語モデルを処理するために開発環境として必要なマシン性能もかなり高いものになります。

4. コード生成ツールの試用

2023 年 4 月時点で実際に動作するコード自動生成のツール（言語モデル×開発環境のツール）には、有償（ツール or/and 言語モデル）、今だけ無償で利用者登録要 、等のものがありますが、本資料では無償で利用者登録不要の CodeGeeX を使ってコード生成のイメージを確認します。

4.1. CodeGeeX について（注意点）

CodeGeeX は GitHub に登録されているオープンソースで、以下のように紹介されています[6]。
「CodeGeeX は、20 を超えるプログラミング言語の大規模なコード コーパスで事前トレーニングされた、130 億のパラメーターを持つ大規模な多言語コード生成モデルです。2022 年 6 月 22 日の時点で、CodeGeeX は 1,536 個の Ascend 910 AI プロセッサのクラスターで 8,500 億を超えるトークンでトレーニングされています。」
そして、「CodeGeeX にはいくつかのユニークな機能があり」として以下の記載があります。
●多言語コード生成: CodeGeeX は、Python、C++、Java、JavaScript、Go など、＜以下略＞
●カスタマイズ可能なプログラミング アシスタント: CodeGeeX は、VS Code 拡張機能マーケットプレースで無料で入手できます。コード補完、説明、要約などをサポートし、＜以下略＞
●オープンソースとクロスプラットフォーム: すべてのコードとモデルの重みは、研究目的で公開し＜以下略＞

＜2023/4/1 現在 CodeGeeX：ライセンス条項＞

●コードは Apache-2.0 ライセンス適用で商業使用、改造、配布が許可されますが、データモデルに関しては以下の記載があり、**商用、軍用での使用と中国の弱体化に資する目的で使うことを禁じ**ています。https://github.com/THUDM/CodeGeeX/blob/main/MODEL_LICENSE

[6] THUDM/CodeGeeX https://github.com/THUDM/CodeGeeX

【CodeGeeX/MODEL_LICENSE】

3. Restriction

You will not use, copy, modify, merge, publish, distribute, reproduce, or create derivative works of the Software, in whole or in part, for any commercial, military, or illegal purposes.

You will not use the Software for any act that may undermine China's national security and national unity, harm the public interest of society, or infringe upon the rights and interests of human beings.

項番２のリスクで触れたようにこのツールは cn ドメインと通信し、下図のような内容を送信します。

```
POST https://tianqi.aminer.cn/api/v2/multilingual_code_generate_adapt HTTP/1.1
Accept: application/json, text/plain, */*
Content-Type: application/json
User-Agent: axios/0.24.0
Content-Length: 1077
Host: tianqi.aminer.cn
Connection: close

{"lang":"Java","prompt":"class Test4 {\r\n    /**\r\n     * 小数点付きのkingakuからStreamを作成する\r\n     * null と
数値以外を除外する\r\n     * Streamを10進数に変換する\r\n     * 変換した10進数を集計する。\r\n     * @param kingaku\r\n
* @return sum\r\n     */\r\n    BigDecimal sumValueOfArray(String[] kingaku) {\r\n        return
Arrays.stream(kingaku).map(BigDecimal::new).map(BigDecimal::valueOf).sum();        //return
Arrays.stream(kingaku).mapToDouble(e -> Double.parseDouble(e)).sum(); //return
Arrays.stream(kingaku).mapToLong(e -> Long.parseLong(e)).sum(); //return Arrays.stream(kingaku).mapToInt(e ->
Integer.parseInt(e)).sum(); //return Arrays.stream(kingaku).mapToBigDecimal(e -> new BigDecimal(e\r
\n","n":1,"apikey":"68cf004321e94b47a91c2e45a8109852","apisecret":"e82b86a16f9d471ab215f653060310e3","temperat
ure":0.2,"top_p":0.95,"top_k":0,"suffix":"\r\n    }\r
\n}","isFimEnabled":true,"ext":{"uid":null,"mid":"2190e459854fc99895569dd5ebeaaffacad1a5bb9691b15ff6af57ffabf2
b5cb","ide":"VSCode","ideVersion":"1.77.3"}}
```

※ 自分でこのツールを試す場合は、私用の範囲で、他国の弱体化に使わず、かつ**セキュリティに注意**して利用してください。http の**アクセス記録／IP アドレスはあちらに記録**されます。

（本資料は自由に使って構いませんが、部分転載等の際は注意書きを忘れずに添付してください）

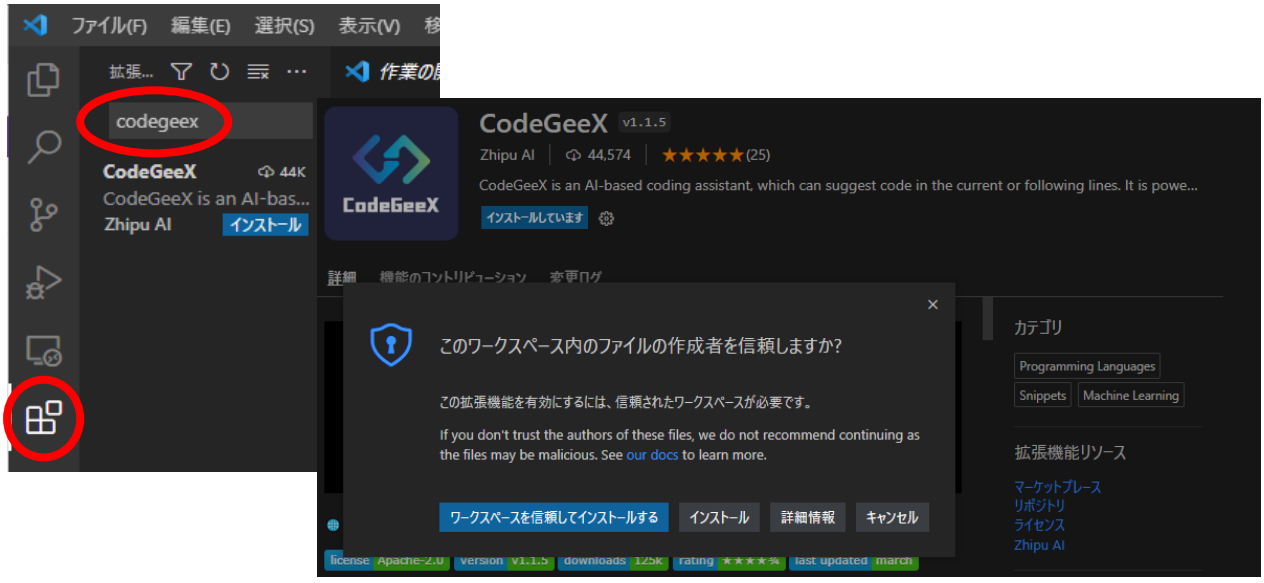
4.2. CodeGeeX のインストール

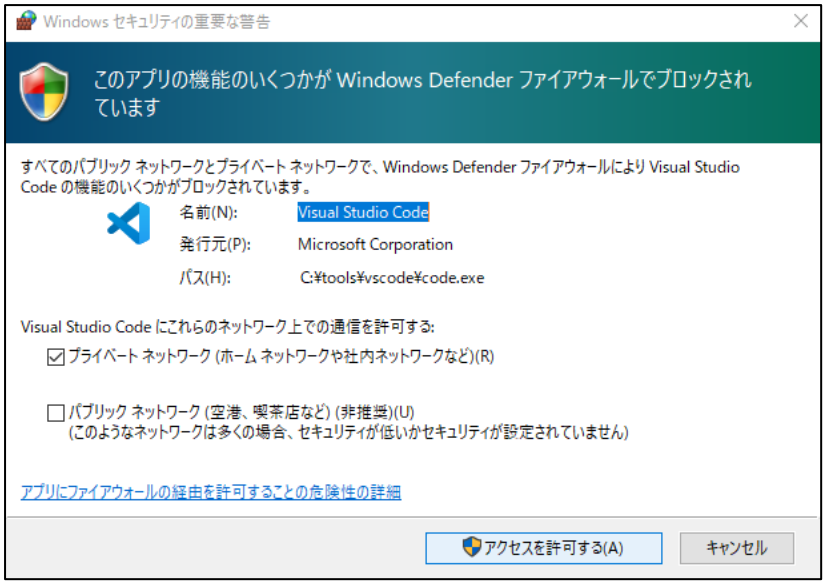
VSCode の機能拡張としてインストールします。

① アクティビティバーの機能拡張（Ctrl＋Shift＋X）を選択し、検索窓に”codegeex”と打鍵し［インストール］を押下

② ［ワークスペースを信頼してインストールする］を押下

③ ファイアウォールのブルック解除を確認されるので、［アクセスを許可する］を押下





4.3. CodeGeeX の使用

インストールするとデフォルトで CodeGeeX のコード補完が有効になっています。

以下、言語を Java にしてコードを入力していきます（アクセス修飾子等の省略可能な部分は省略）。

4.3.1. 英文のコメントからコード生成

① メソッドの宣言を書いた後にコメント（3 行目）を書くとコードの提案が始まります。

```
1  class Test{
2      int selElementsAndSum(Map<String, Integer> map, String[] keys){
3      // Select elements of Map that match in keys and return sum of values.
           int sum = 0; // Initialize sum to zero.
4
5  }
6
```

TAB キーで CodeGeex の提案したコードが確定（Tab 以外でクリア）します。1 行づつ進めると...

```
1  class Test{
2      int selElementsAndSum(Map<String, Integer> map, String[] keys){
3      // Select elements of Map that match in keys and return sum of values.
4
5          int sum = 0; // Initialize sum to zero.
6          for(String key: keys){ // Iterate over keys.
7              if(map.containsKey(key)){ // Check if key is in map.
8                  sum += map.get(key); // If key is in map, add its value to sum.
9              }
10         }
11         return sum; // Return sum.
12     } // Close method.
13 }
```

上記のようなコードが完成します。

コード自動生成（CodeGeeX 他）

② クラス内に行コメントを記述するとコメントの続きやメソッド全体が提案されたりもします。

```
class Test2{
    // Select elements of Map that match in keys and return sum of values.
    public static int sumValues(final Map<String, Integer> map, final Set<String> keys) {
        int sum = 0;
        for (final String key : keys) {
            final Integer value = map.get(key);
            if (value!= null) {
                sum += value;
            }
        }
        return sum;
    }
}

    ....
```

提案を受け入れる場合は、行毎に TAB キーで確定していきます。

③ メソッドコメントを書き始めると、コメントの補完とメソッド宣言以降も提案してきます。
@return の記述では、入力パラメータの設定値による分岐も提案してきます。

```
class Test2{
/**
 * Select elements of Map that match in keys and return sum of values.
 * @param map indata
 * @param keys keys to be tested for match.  If null, all keys are tested.
 * @return int sum of values of matching elements.  0 if map is empty or keys are null.    */
```

メソッド宣言を入力すると、処理の提案が始まります。

```
class Test{ Untitled-1    class Test2{ Untitled-2
class Test2{
/**
 * Select elements of Map that match in keys and return sum of values.
 * @param map indata
 * @param keys keys to be tested for match.  If null, all keys are tested.
 * @return sum of values of matching elements.  If keys is null, returns the sum of all values.  If
 */
public int sumValueOfMap(Map<String, Integer> map, String[] keys){
    int sum = 0;    //sum of values of matching elements.  If keys is null, returns the sum of all v
    if(keys == null){ //if keys is null, test all keys.  If keys is non-null, return null if no matc
```

分岐した処理ブロックの途中で提案が停止（下部ステータスバーに「No Suggestion」）したら"}"で処理ブロックを閉じると提案が続きます。

```
class Test2{
/**
 * Select elements of Map that match in keys and return sum of values.
 * @param map indata
 * @param keys keys to be tested for match.  If null, all keys are tested.
 * @return sum  of values of matching elements.  If keys is null, returns the sum of all values.  If keys is
 */
public int sumValueOfMap(Map<String, Integer> map, String[] keys){
    int sum = 0;    //sum of values of matching elements.  If keys is null, returns the sum of all values.
    if(keys == null){ //if keys is null, test all keys.  If keys is non-null, return null if no matching ele
        for(String key:map.keySet()){ //for each key in map.  If key is in map, add its value to sum.
            if(map.get(key)!=null){ //if the value in map.get(key) is not null, add it to
                sum += map.get(key); //sum.  If map.get(key) is null, the value in map.get
                }                                                       //(key) is not in map.  So, the value in
        }
        return sum; //return sum.  If keys is null, return sum.  If keys is non-null, return null if no matc
    }else{ //keys are not null, test only keys.  If keys are in map, add its value to sum.  If keys are not
        for(String key:map.keySet()){ //for each key in map.  If key is in map, add its value to sum.
            if(map.containsKey(key)){ //if key is in map, add its value to sum.  If key is in map,
                sum += map.get(key); //sum.  If key is in map, NaN is not in sum.  So, NaN is
                }
        }
        return sum; //return sum.  If keys are not null, return sum.  If keys are in map, add its value to s
    }
}
import java.util.*; //import java.util.Collections; //import java.util.Collections; //import java.util.HashM
import java.util.Map; //import java.util.HashMap; //import java.util.Map; //import java.util.Set;
```

4.3.2.　日本語のコメントからコード生成

日本語は単語の区切りが解り難かったり動詞の活用や助詞の使い方が柔軟な分、プログラミングには使い難いのですが、日本語で書いた仕様が通らないと使いものにならないので試してみました。

① ファイルの先頭から日本語でコメント「List の keys で指定した値と一致する Map の要素を選択し、値の合計を返す。」を書くとそれらしいライブラリの import 文とテストコードが書かれ出します。

```
//Listのkeysで指定した値と一致する Map の要素を選択し、値の合計を返す。
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.function.Function;  // for Integer::valueOf (from Guava library) 描述: https://codegeex.
import java.util.stream.Collectors;  // for toString (from Guava library) 描述: https://codegeex.cn
import static java.util.stream.Collectors.groupingBy;  // for (key, value) -> value (from Guava library)
import static java.util.stream.Collectors.summingInt;  // for int (from Guava library) 描述: https://code
import static java.util.stream.IntStream.rangeClosed;  // for int (from Gua  library) 描述: https://
public class MapTest {  // the class name must be MapTest (not Map<Integer, Integer> or Map<String, Integer> or M
    public static void main(String[] args) {  // the main method must be named main (not main$Main) 创建一个名
        Map<Integer, Integer> map = new HashMap<>();  // a new map (not a new HashMap) 描述: https
        map.put(1, 1);  // add a new entry (not replace an existing entry) 描述: https (key = 1,
        map.put(2, 2);  // add a new entry (not replace an existing entry) 描述: https (key = 2,
```

※ 一見 日本語が解釈できていそうな感じもしますが、英字の List や Map に反応しただけかもしれません。そして、なぜテスト用のコードにしようとツールが判断したのかも分かりません

コード自動生成（CodeGeeX 他）

② “List の keys で指定した値”の部分を「List の keys の値と一致する Map の要素を選択し、値の合計を返す。」と変え、メソッド宣言を手書きすると、以下のように提案してきます。

```
//Listのkeysの値と一致する Map の要素を選択し、値の合計を返す。
import java.util.AbstractMap;
import java.util.HashMap;
import java.util.Map; // for AbstractMap.SimpleEntry<K,V> and HashMap<K,V> classes. //
import java.util.Map.Entry; // for SimpleEntry<K,V> class. // Note: SimpleEntry<K,V>
import java.util.Scanner; // for the function readLine(). // Note: readLine() is not part
import java.util.Set; // for the function keys(). // Note: keys() is not part of the JDK.
import java.util.TreeSet; // for the function keys(). // Note: keys() is not part of the

int sumValueOfMap(Map<String, Integer>, List<String> keys){
    int sum = 0; // the value of the sum. // Note: sum is not part of the function. //
    for(Entry<String, Integer> e: keys.toMap().entrySet()){ // for each entry in keys. //
        sum += e.getValue(); // add the value of the entry. // Note: sum is not part of
    } // for each entry in keys. // Note: // Note: // Note: // Note: //
    return sum; // return the sum. // Note: sum is part of the function. // Note: sum is

}
```

③「List の keys で指定した値と一致する Map の要素を選択し、値の合計を返す。」に変えメソッド宣言を指定すると、JDK 1.8 で追加された Stream を使ってブロック内の処理を書き始めます。

```
//Listのkeysで指定した値と一致する Map の要素を選択し、値の合計を返す。
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.function.Function; // for Integer::valueOf (from Guava library) 描述: https://codegeex.
import java.util.stream.Collectors; // for toString (from Guava library) 描述: https://codegeex.cn

int sumValueOfMap(Map<Strint, Integer> map, List<String> keys){
    return map.entrySet() // Map.Entry<K,V>[] 型: https://codegeex.cn 描述: https://codegeex.cn 制DB: https:
        .stream() // Stream<Map.Entry<K,V>> 型: https://codegeex.cn 描述: https://code
        .map(e -> e.getValue().equals(Integer.valueOf(e.getKey().toString()))? e.get
        Value(): null) // Function<V,V> 型: https://codegeex.cn 描述: https://codegeex.
        .filter(Objects::nonNull) // Predicate<V> 型: https://codegeex.cn 描述: https://
        .mapToInt(Integer::intValue) // Function<V,Integer> 型: https://codegeex.cn 描述:
        .sum(); // int 型: https://codegeex.cn 描述: https://codegeex.cn 制DB: https
}
public class SumValueOfMap { // Main class: https://codegeex.cn 描述: https://codegeex.cn 制DB
```

※ コメントの書き方というよりはコメントに合わせてメソッド宣言を書くのが有効でした。それに、メソッドの呼び出し側の都合から考えればメソッド名や引数を自動生成させることはないでしょう

4.3.3. メソッドコメントの書き方による変化

Stream を使った以下のコードが生成できるか試してみました。

```
-------------------------------------------------------------------------
  BigDecimal sumValueOfArray(String[] kingaku) {
       return Arrays.stream(kingaku)
                    .filter(Objects::nonNull)
                    .filter(s -> s.matches("¥¥d+¥¥.*¥¥d*"))  //数値と"."*1 以外を除外
                    .map(BigDecimal::new)                     //文字列 10 進数変換
                    .reduce(BigDecimal.ZERO, BigDecimal::add);//集計
  }
-------------------------------------------------------------------------
```

① メソッドコメントとして実行したい内容を記述

```
class Test4 {
    /**
     * 小数点付きのkingakuからStreamを作成する。
     * nullと数値以外を除外する。
     * Streamを10進数に変換する。
     * 変換した10進数を集計する。
     * @param kingaku
     * @return sum
     */
    BigDecimal sumValueOfArray(String[] kingaku) {
        BigDecimal sum = BigDecimal.ZERO; //initial value for the sum. （初期値） 数値以外。Strea
        for (String number : kingaku) { //for each number in the array. 数字節. 数値
            sum = sum.add(new BigDecimal(number)); //add the number to the sum. 数字節. 数

    }
```

ここで、下部ステータスバーに「No Suggestion」が出て止まるので、14 行目"}"を打鍵し閉じる

```
        }
        return sum; //return the sum. 数字節. 数 返却。数字節. 数 取引
```

return が提案されて一連の処理完成します。

【結果】"null と数値以外を除外する"という部分が反映されません

② コメントの句点（「。」）を最終行だけに付けそれ以外の行から句点を削ると一気に以下の提案となりましたが、「null と数値以外を除外」はやはりコード化されていません。

```
class Test4 {
    /**
     * 小数点付きのkingakuからStreamを作成する
     * nullと数値以外を除外する
     * Streamを10進数に変換する
     * 変換した10進数を集計する。
     * @param kingaku
     * @return sum
     */
    BigDecimal sumValueOfArray(String[] kingaku) {
        return Arrays.stream(kingaku).map(BigDecimal::new).map(BigDecimal::valueOf).sum();

    }
}
```

11 行目の行末にはコメントとして以下の出力がされました。

```
.sum();         //return Arrays.stream(kingaku).mapToDouble(e ->
Double.parseDouble(e)).sum(); //return Arrays.stream(kingaku).mapToLong(e ->
Long.parseLong(e)).sum(); //return Arrays.stream(kingaku).mapToInt(e ->
Integer.parseInt(e)).sum(); //return Arrays.stream(kingaku).mapToBigDecimal(e ->
new BigDecimal(e
```

③「null と数値以外を除外する」を「数値を選択する」に変えると、提案が変更されましたが...

```
class Test4 {
    /**
     * 小数点付きのkingakuからStreamを作成する
     * 数値を選択する
     * Streamを10進数に変換する
     * 変換した10進数を集計する。
     * @param kingaku
     * @return sum
     */
    BigDecimal sumValueOfArray(String[] kingaku) {
        return Arrays.stream(kingaku) //ArrayList<String> stream = Arrays.stream(kingaku) //stream.mapToDouble(Integer::pa
            .map(this::toBigDecimal) //Stream<BigDecimal> toBigDecimalStream = stream.mapToDouble(Integer::
            mapToObj(this::toString) //String to String stream = stream.mapToDouble(Integer::parseInt).mapToObj(this
            .map(number -> new BigDecimal(number)) //BigDecimal to BigDecimal stream = stream.mapToBigDecimal(this
            .reduce(BigDecimal.ZERO, BigDecimal::add)) //return stream.reduce(BigDecimal.ZERO,
            .map(number -> number.movePointRight(number.scale())) //return stream.mapToDouble(this::toBigDecimal);
            .map(number -> number.setScale(number.scale())) //return stream.mapToBigDecimal(this::toBigDecimal
            .map(number -> number.movePointRight(number.scale())) //return stream.mapToBigDecimal(this::toBigDec
            .map(number -> number.abs()) //return stream.mapToBigDecimal(this::toBigDecimal) //return stream.map
            .max(BigDecimal.ZERO).get().setScale(number.scale())) //return stream.max(this::compareTo
            .orElseThrow(() -> new IllegalStateException("Unknown error.")) //return stream.max(BigDecimal.ZERO).get().
            toBigDecimalStream //return toBigDecimalStream.max(BigDecimal::compareTo); //return toBigDecimalStream.
            .reduce(BigDecimal.ZERO, BigDecimal::add)); //return toBigDecimalStream.reduce(BigDecimal
            //.add); //return toBigDecimalStream.reduce(BigDecimal.ZERO, BigDecimal::add); //return to
            //BigDecimalStream.reduce(BigDecimal.ZERO, BigDecimal::add); //return toBigDecimalStream.red
```

④ コンパイルが通らなそう...

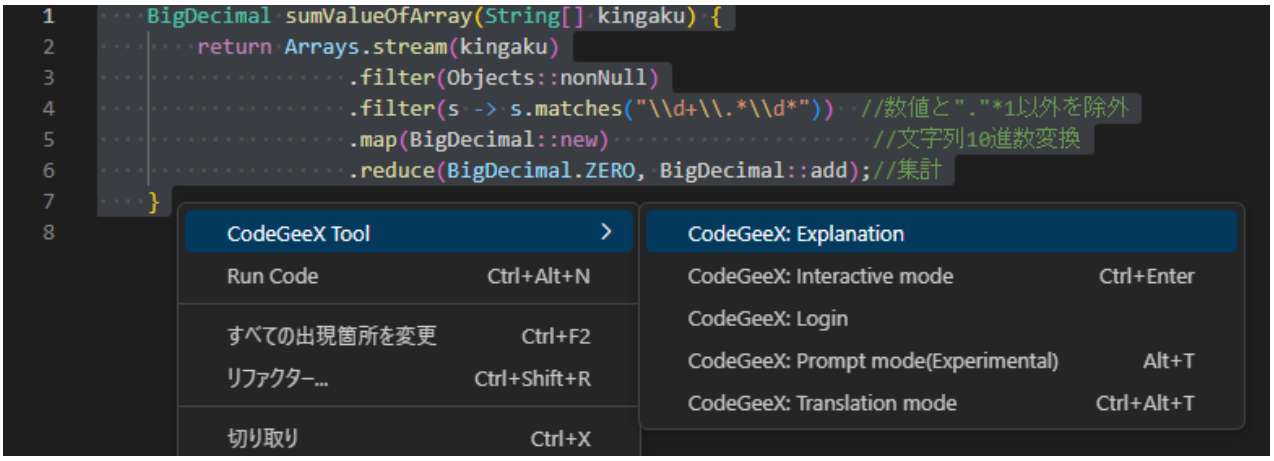
```
class Test4 {
    /**
     * 小数点付きのkingakuからStreamを作成する
     * Streamから数字の値を選択し、10進数に変換する
     * 変換した10進数を集計する。
     * @param kingaku
     * @return sum
     */
    BigDecimal sumValueOfArray(String[] kingaku) {
        return new BigDecimal(kingaku.length).movePointLeft(2).map(i -> new BigDecimal(i)).reduce(BigDecimal.ZERO, BigDecimal::add);
            .TEN);  //.movePointLeft(2); a.movePointLeft(2); a.movePointLeft(2, BigDec
    }
```

⑤ なぜか、10 進数にした後に Long⇒Integer と変換してから計算して 10 進数に戻している

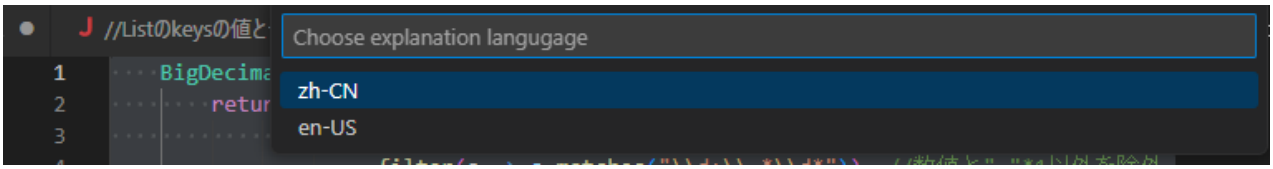
```
class Test4 {
    /**
     * 小数点付きのkingakuからStreamを作成する
     * Streamから数字の値を選択し、10進数に変換する
     * 変換した10進数を集計して返す。
     * @param kingaku
     * @return sum
     */
    BigDecimal sumValueOfArray(String[] kingaku) {
        return Arrays.stream(kingaku) // Stream<String> 作成(範囲限定) 或 Stream<String> 作成(終端) 以降
            .map(this::toBigDecimal) // Stream<BigDecimal> 作成(範囲限定) 或
            .mapToLong(this::toBigDecimal) // LongStream<BigDecimal> 作成(範囲限定
            .mapToInt(BigDecimal::intValueExact) // IntStream<BigDecimal> 作成(範囲
            .sum()) // BigDecimal 作成(終端) 。 IntStream.sum() 作成(終
            .toBigDecimal(); // BigDecimal 作成(終端) 。 toString() 作成(

```

4.3.4. 処理内容からコメントを生成

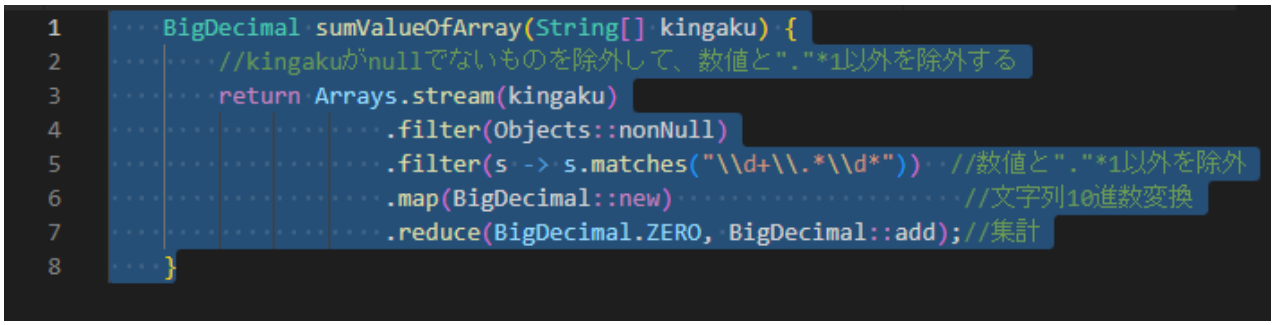
CodeGeeX のコード説明を作成する機能を実行してみました。



使用言語の選択肢は今のところ 2 つだけですが...



10 秒程度すると、2 行目が挿入されました。内容はコードと逆になってしまっていますが、一連の操作から CodeGeeX が日本語を学習したのかも（学習は拒否 disclaim を選択したハズ）



再度 言語に en-US を選択して実行すると、2～7 行目が追加されました。
このコメントで基にしたコードが再現できるか否かは確認していませんが、このコメントを書くよりもコードを書いた方が手間がなさそうな粒度です。

5. コード自動生成のその他のツール

5.1. ブラウザで試せる Codeium

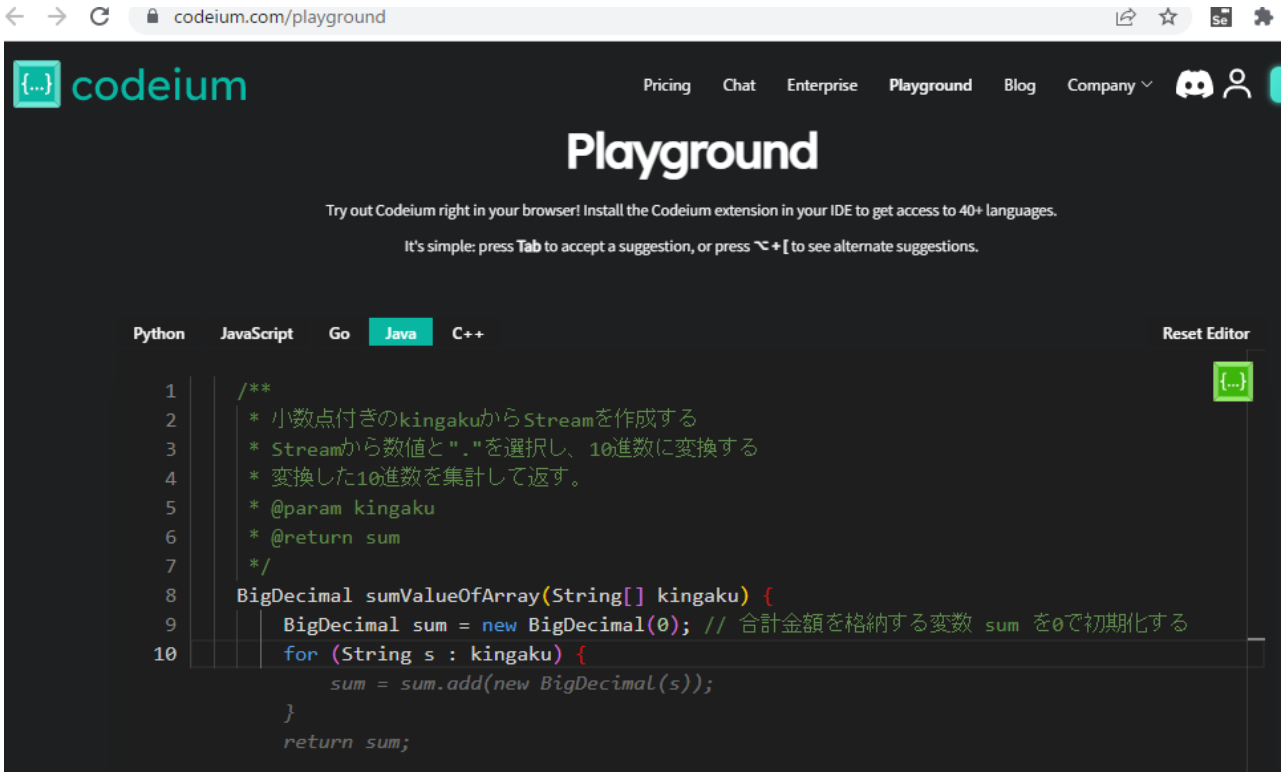
ブラウザの機能拡張ですがインストールしなくても試すことができ、日本語も使えます。
https://codeium.com/playground

プログラミング言語を選び、画面中央のエディター部分にコメントを入力するとコードが生成されます。Enter を押下すると 1 行づつコードが生成され、Tab でコードが確定します。

＜使用例＞以下のコードを貼り付けて Enter を押下

```
/**
 * 小数点付きの kingaku から Stream を作成する
 * Stream から数値と”.”を選択し、10 進数に変換する
 * 変換した 10 進数を集計して返す。
 * @param kingaku
 * @return sum
 */
BigDecimal sumValueOfArray(String[] kingaku) {
```

※ 9 行目以降が生成されたコードです。 9 行目の行コメントも日本語で出力されました。



5.2. オープンソースの言語モデルを利用する Hugging Face

The AI community building the future.（未来を築く AI コミュニティ）、機械学習に関するアプリやツールを開発しているアメリカの企業です。事前学習済モデルにアクセスするための API も開発していて GitHub で公開されている BERT や GPT- 2 を含むオープンソースの 18 万以上（2023/4 時点）の事前学習済モデルがこの API を通して使えるようになっています[7]。

VSCode 用の機能拡張[8]は無料（MIT ライセンス）で公開されていて、補完機能等が使えます。

5.3. スタックオーバフロー検索（CROKAGE、Que2Code 他）

スタックオーバフロー[9]（Stack OverFlow）は世界中のプログラマーがコードに関するアドバイスを求めて投稿し、有識者がコード例や改善案を回答するサイトです。正式ドキュメントだけでは分からない具体的な（殆どはそのまま動作可能な）実装例が実際の開発者から寄せられるので、現段階では「コードの自動生成」よりもこちらのコード例を参考にする方が開発の助けになるかもしれません。

このサイトに登録されているコード例（スニペット）を探すためのツールも幾つか開発されており、CROKAGE や Que2Code 等が GitHub に登録されています。

6. 現状の評価

ある時 Web 翻訳の精度が格段に向上したことに気づき驚いた記憶があります。その頃、海外向け製品のマニュアル校正で担当部分に間違いを見つけ、米国の編集者に「表 xx の yy 番目に列を追加し、その列見出しに…」と修正の指示をしたいのですが、そんな英語力はありません。社内に翻訳サイトがあったのですがインターネットに公開されている他の Web 翻訳と同様、英訳、和訳と往復させると単語の数が減っていくような代物です。ところが半ば自棄で検索サイトの翻訳機能を試したところ、関係代名詞を使いこなした上に英訳和訳を往復させても 2 往復目以降は表現が変化しません。これがちょうど BERT が公開された頃の出来事で、BERT（技術要素でいうと他の言語モデルでも使われている Embedding、Attention、Transformer）により単語と熟語レベルの翻訳から句点で区切られた文という単位で翻訳できるように人知れず進化していたわけです。

事前学習済言語モデルは数値化したトークン[10]とトークンの関連（重み）付けで保存された情報で、コード生成を行う場合は既存のソースコードから事前学習した言語モデルをトークン化した自然言語（ソースコメント）で一致度を基準に検索します。これにより自然言語の文からソースコードの文や式に変換できるのですが、仕組みの観点から以下の点を考慮する必要があります。

① ソースコード 1 文を作るのにコメント 1 文を必要とするのであれば、コードを書いた方が早い
② ローカルで作成したクラスやメソッドは事前学習に含まれないので、初期は候補に含まれない

---

[7] https://huggingface.co/models

[8] HF Code Autocomplete
https://marketplace.visualstudio.com/items?itemName=HuggingFace.huggingface-vscode

[9] Stack Overflow https://stackoverflow.com/search?q=stack+overflow+search+engine

[10] トークンは英語系では単語（空白等の区切り文字）、日本語の場合はもう少し細かくなるようです
…詳細は「トークン 日本語 Embedding」で検索

①のコメントと生成されるコードの関係は、学習が進むと 4.3.1 ②のようにコメントからメソッドや for each ブロック程度は生成できるようです。課題は日本語の場合はどう書くか、for each 以外の処理ブロックは生成させられないのか等があります。

②については、ローカルのクラスやメソッドを使っているコードを学習（言語モデルへの登録）すれば提案候補に上がるようになると考えられます。また、設計工程を VSCode 等の開発環境を使って行えるようにすれば、設計工程の用語が言語モデルに取り込まれて設計／仕様から直接コードを生成することが可能になるかもしれません。

7. 今後

＜背景＞

現状 Ruby 以外の言語では日本人は言語仕様に関わることができておらず、スタックオーバフロー等の国際的なコミュニティへの書き込みも見かけません。言語仕様やフレームワークの策定に関われないので頻繁に行われるバージョンアップへの追随が困難になります。新機能を使う予定がなくともOS／ミドルウェア、ライブラリのセキュリティ対応や互換維持のためのバージョンアップが必要になります。これに伴って API が変わったり、廃止になったり、記法の変更になったりで古いアプリの改修が必要になり、現状維持のためだけの後ろ向きな仕事が発生します。

＜その他の考慮点＞

日本語を使ってコード再生成ができればバージョンの問題を軽減できる可能性があります。但し、古いアプリをバージョンアップさせるケースではコメントと生成したコードが混在する現状のツールでは困難で、仕様からコードを生成させるような今よりも高度な機能が必要になります。

＜おすすめ＞

事前学習済言語モデルを使ったコード生成ツールは定番（慣用）コードの入力を省いてくれます。これだけでも有用性が高いですが、オープンソースの事前学習済言語モデルを取得して自前の開発物で強化学習を行えば、自組織が開発したクラスやメソッド、コードパターンを熟知した言語モデルが作れ、セキュリティも維持できます。

現状のツールが提示する内容は完全に満足できるものではなくスピードも遅いですが、ツールの開発は活発に進められており、いつか BERT が登場したときのように、気が付いたら景色が変わっていたということになるかもしれません。組織としても個人としても試してみる価値はあります。

※ツール及び環境の進展が早く、資料の内容を最新の情報に更新・維持し続けるのは困難なので、新しい追記が必要な情報は別の資料として纏めます。

この資料は 2023 年 4 月末時点の情報で作っています。

以上