メディアアート・プログラミング2

東京藝術大学 芸術情報センター開設科目 後期金曜4限 第5週

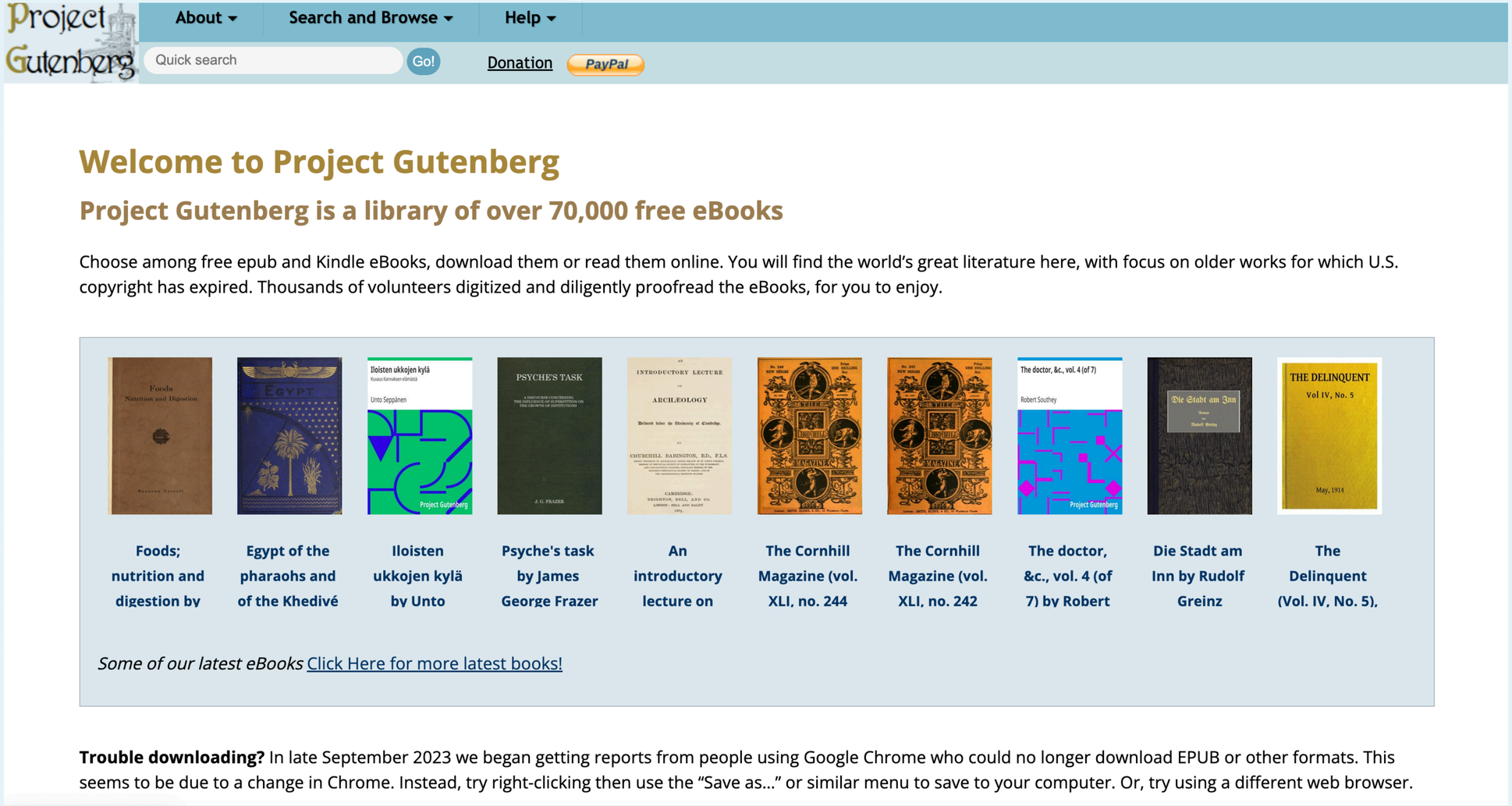
2023.10.31 松浦知也（matsura.tomoya@noc.geidai.ac.jp teach@matsuuratomoya.com）

配列としてのテキストデータ

青空文庫

https://www.aozora.gr.jp/

Project Gutenberg



https://www.gutenberg.org/

図書カード：No.49866

作品名： 変身
作品名読み： へんしん
原題： DIE VERWANDLUNG
著者名： カフカ フランツ

［ファイルのダウンロード｜いますぐXHTML版で読む］

作品データ

分類： NDC 943
作品について： Wikipedia「変身 (カフカ)」
文字遣い種別： 新字新仮名
備考： この作品には、今日からみれば、不適切と受け取られる可能性のある表現がみられます。その旨をここに記載した上で、そのままの形で作品を公開します。（青空文庫）

https://www.aozora.gr.jp/cards/001235/card49866.html

工作員データ

入力：　kompass
校正：　青空文庫

ファイルのダウンロード

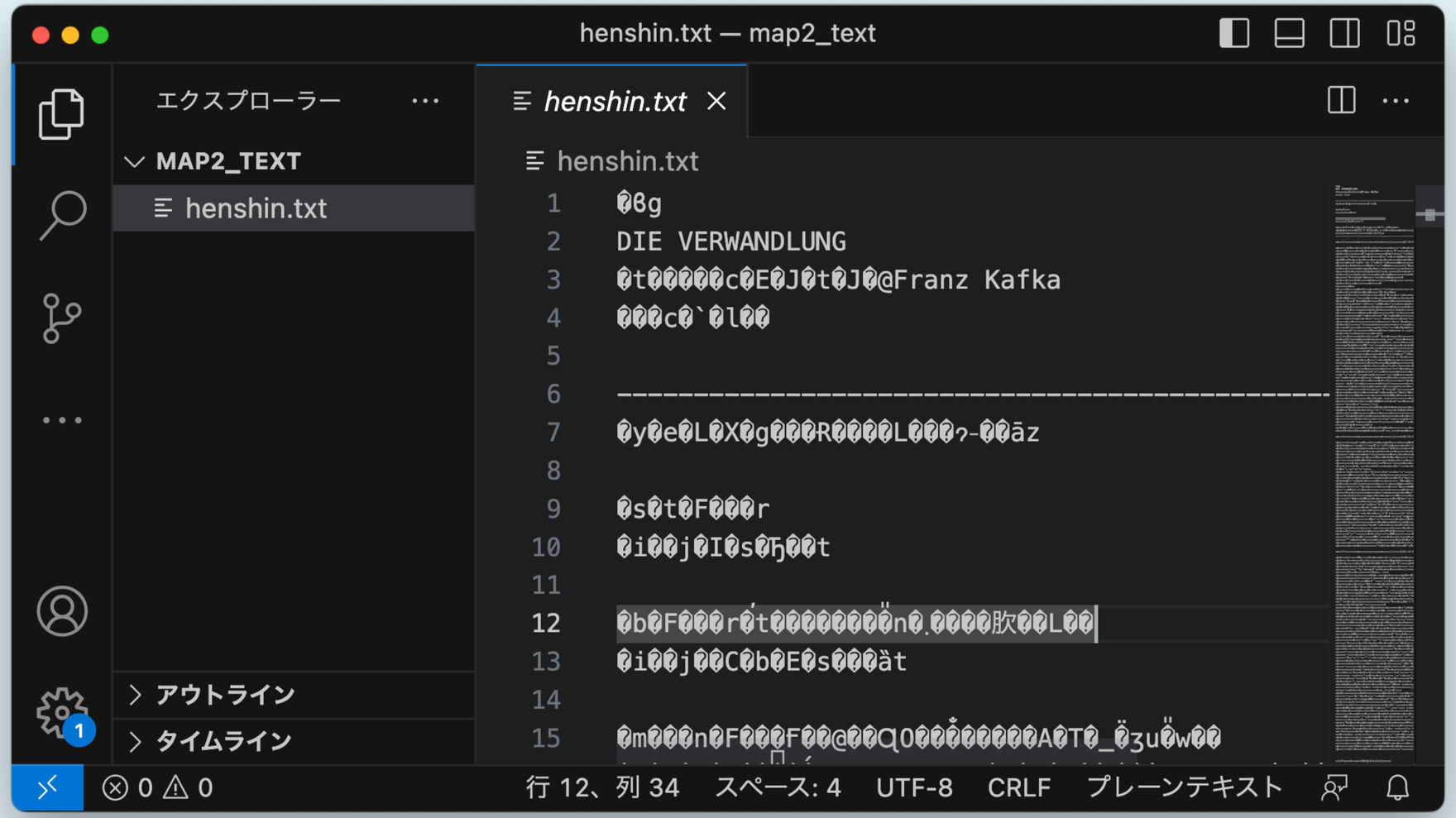
ファイル種別	圧縮	ファイル名（リンク）	文字集合／符号化方式	サイズ	初登録日	最終更新日
テキストファイル(ルビあり)	zip	49866_ruby_41853.zip	JIS X 0208／ShiftJIS	47691	2011-01-01	2016-02-22
XHTMLファイル	なし	49866_41897.html	JIS X 0208／ShiftJIS	119433	2011-01-01	2016-02-22

●ファイルのダウンロード方法・解凍方法

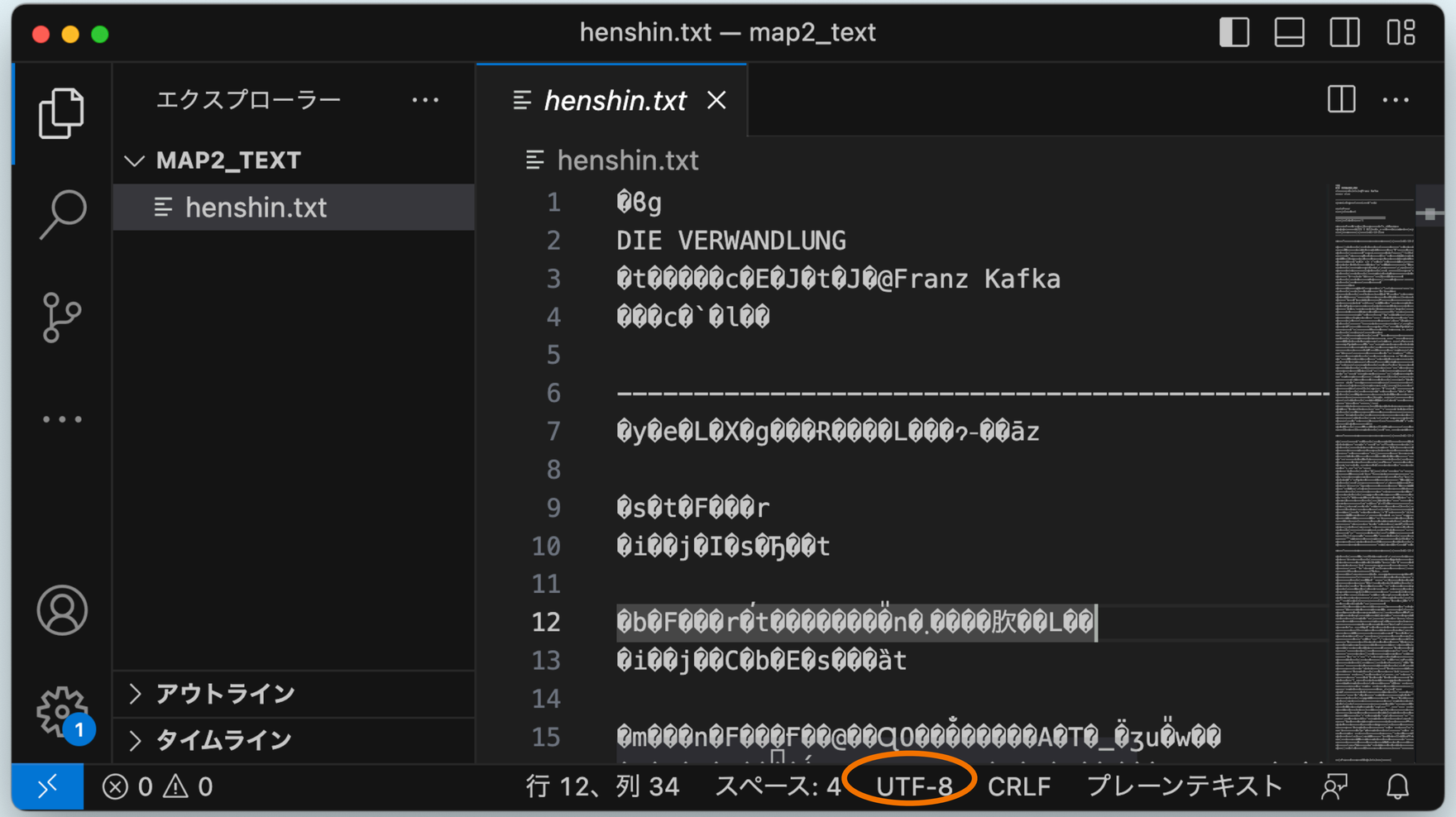
関連サイトデータ

▲

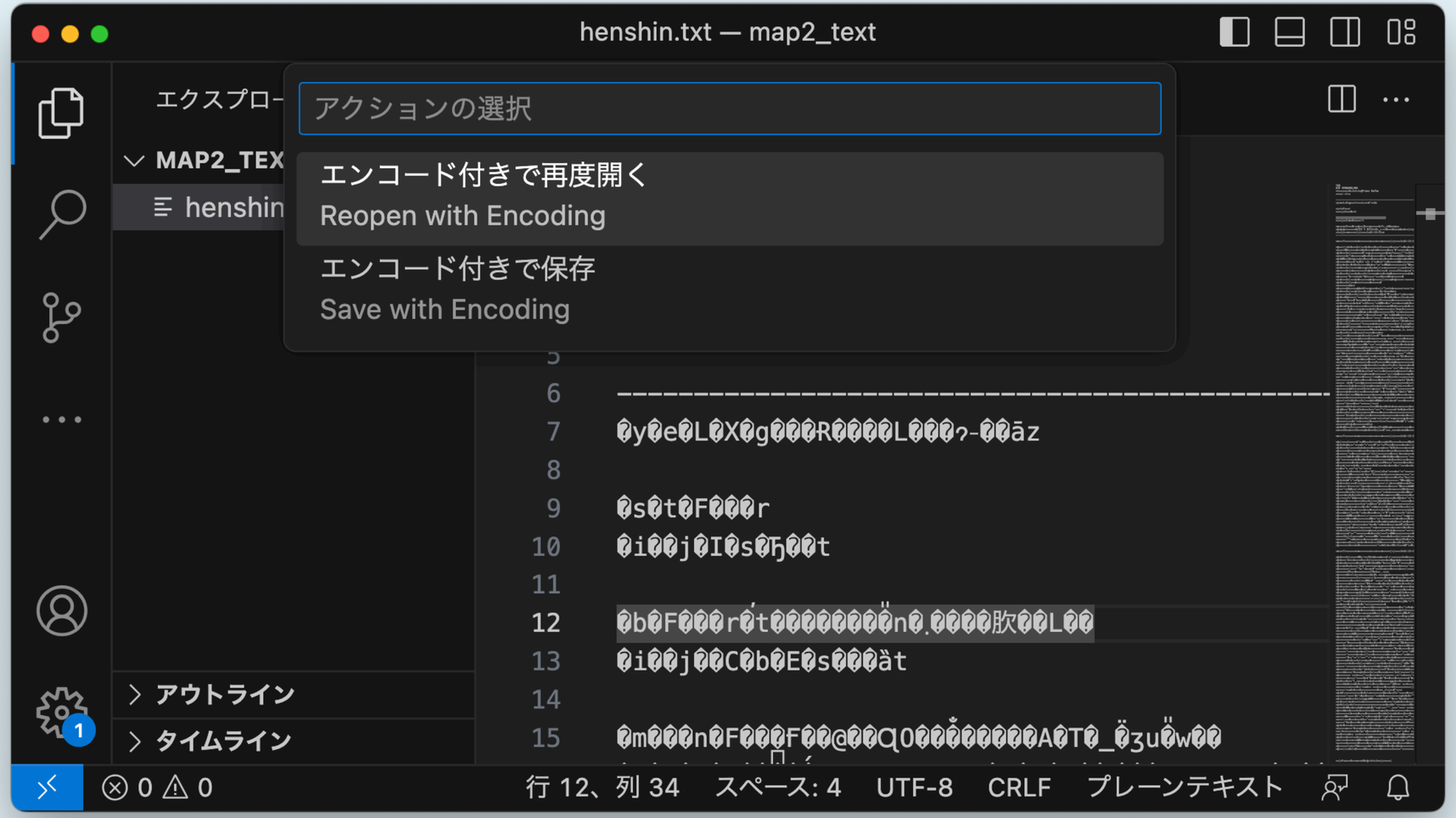
●作家リスト：公開中　[あ] [か] [さ] [た] [な] [は] [ま] [や] [ら] [わ] [他]
●作家リスト：全　[あ] [か] [さ] [た] [な] [は] [ま] [や] [ら] [わ] [他]
●トップ　●インデックス／全　●作家別作品リスト



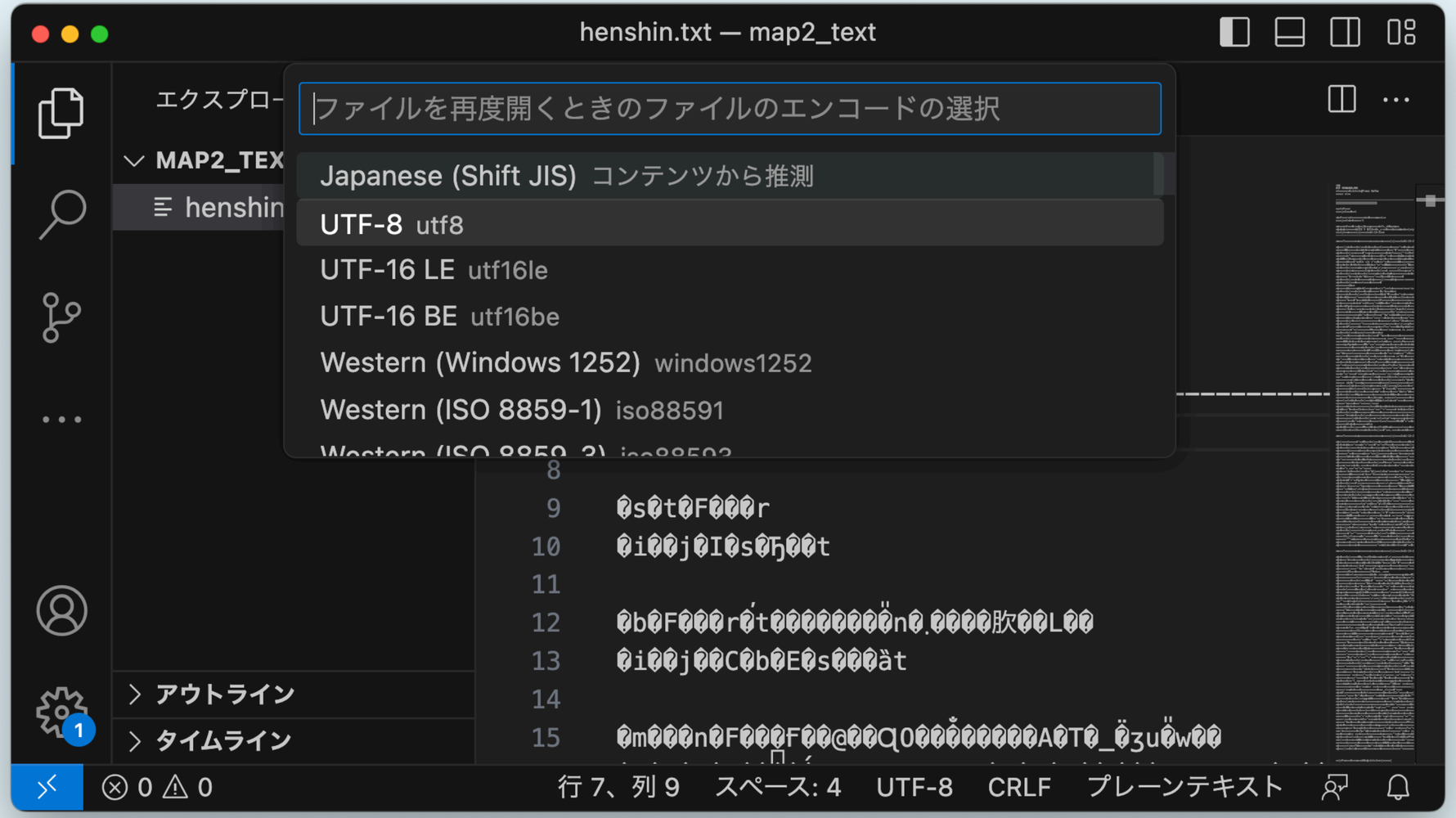
適当なフォルダに入れてVSCodeで開く



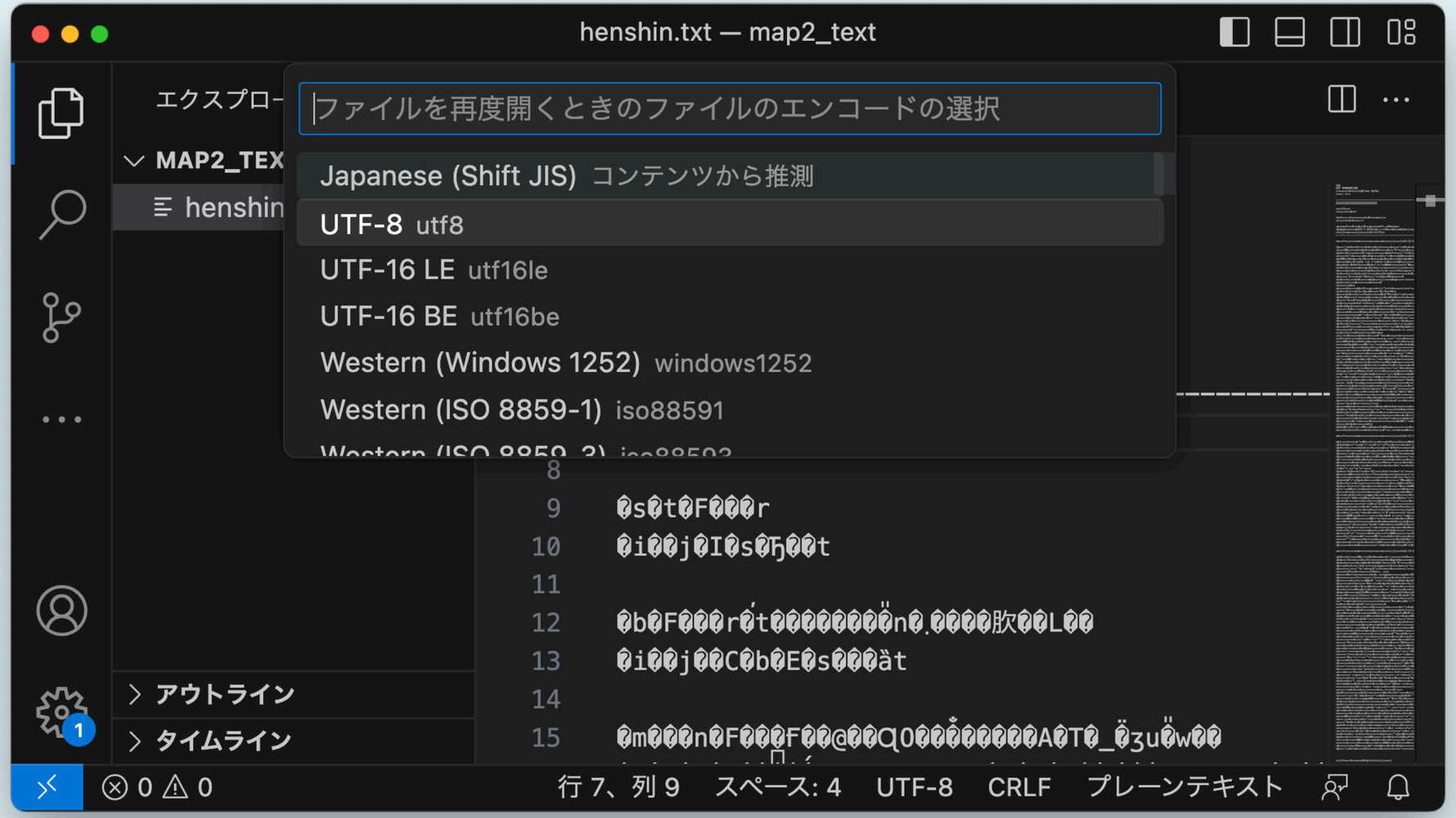
青空文庫は文字コードがShift-JISなので一度UTF-8に変換する



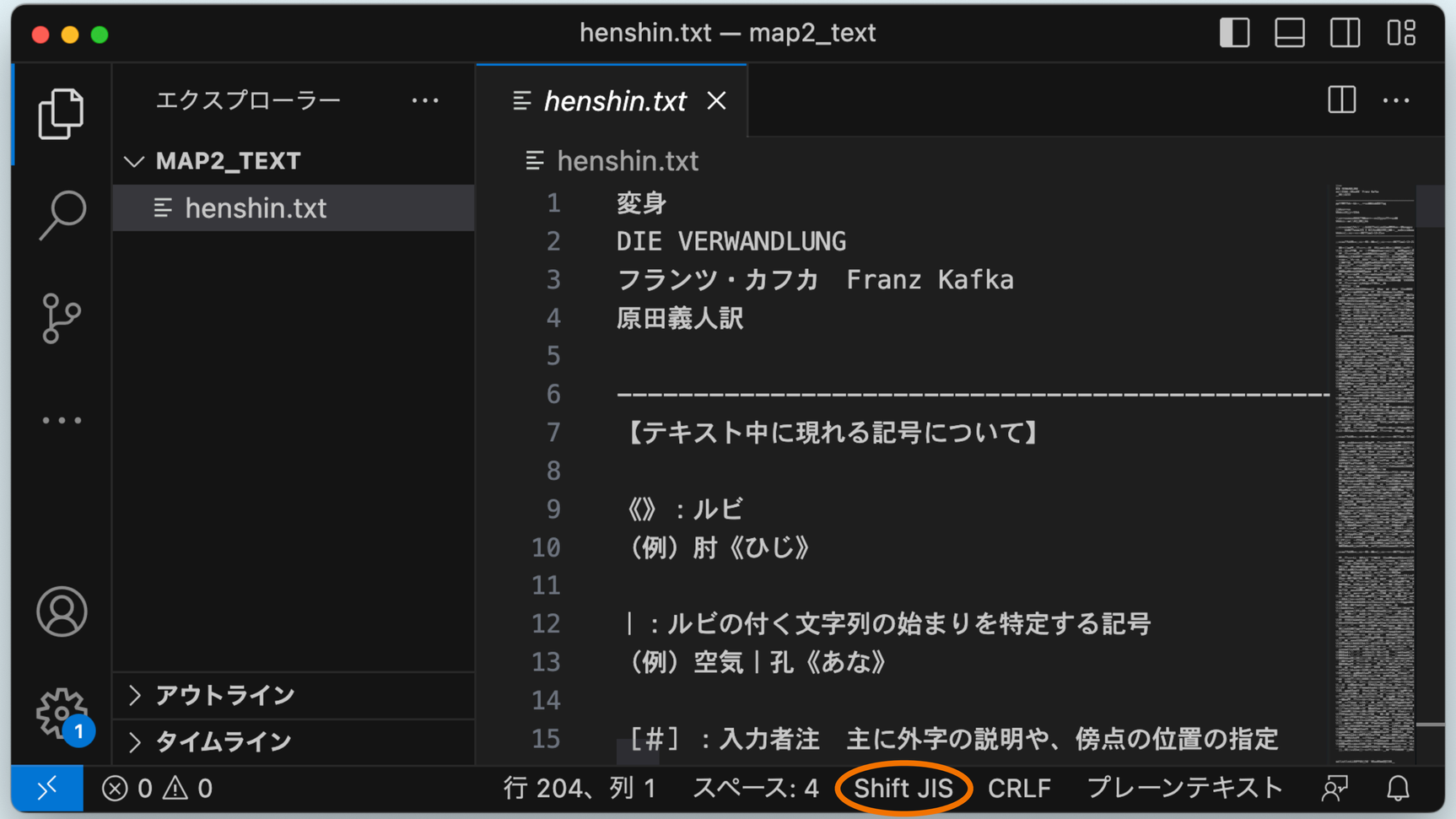
「エンコード付きで再度開く」を選択



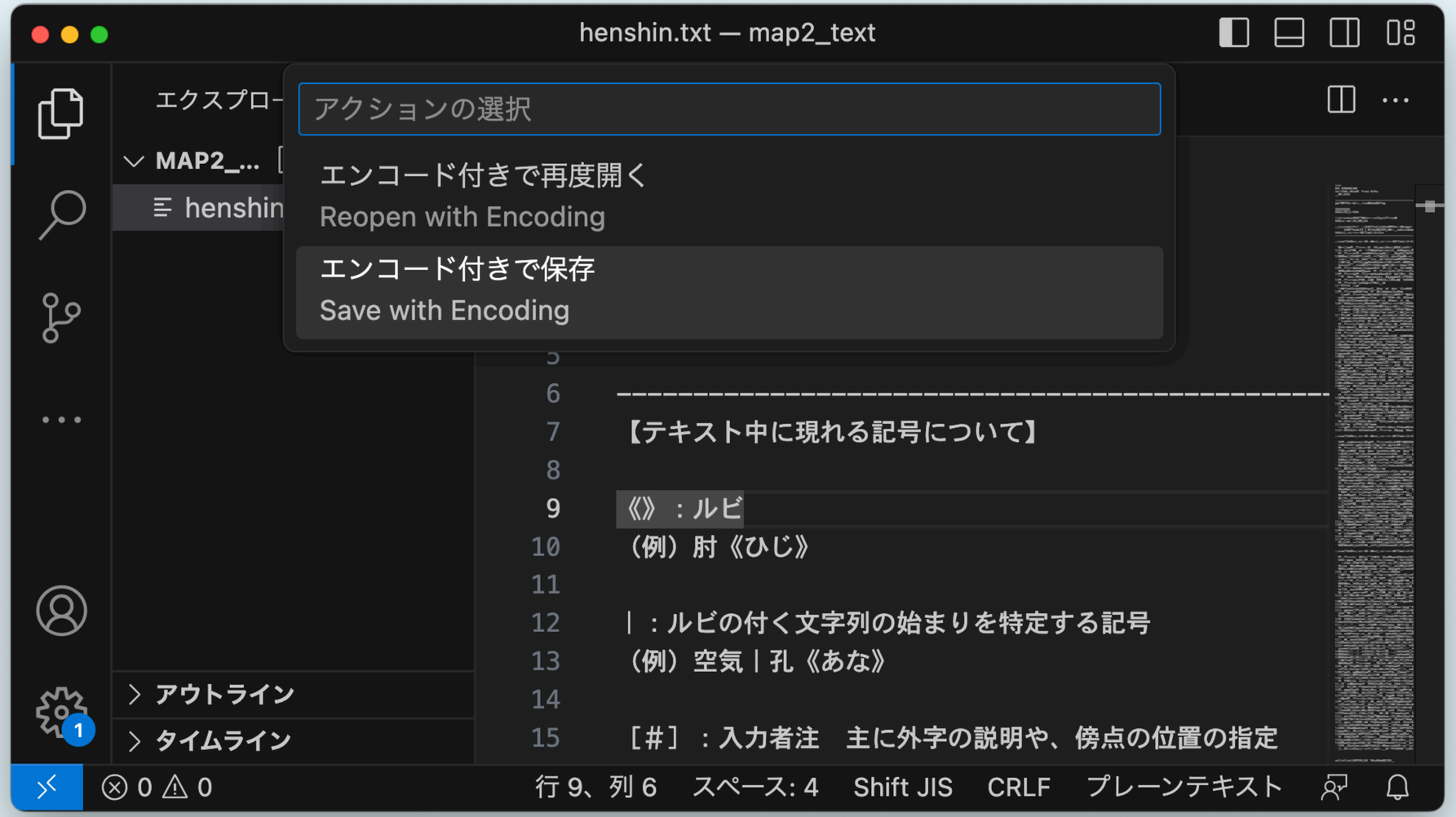
Shift-JISを推測してくれるので選択

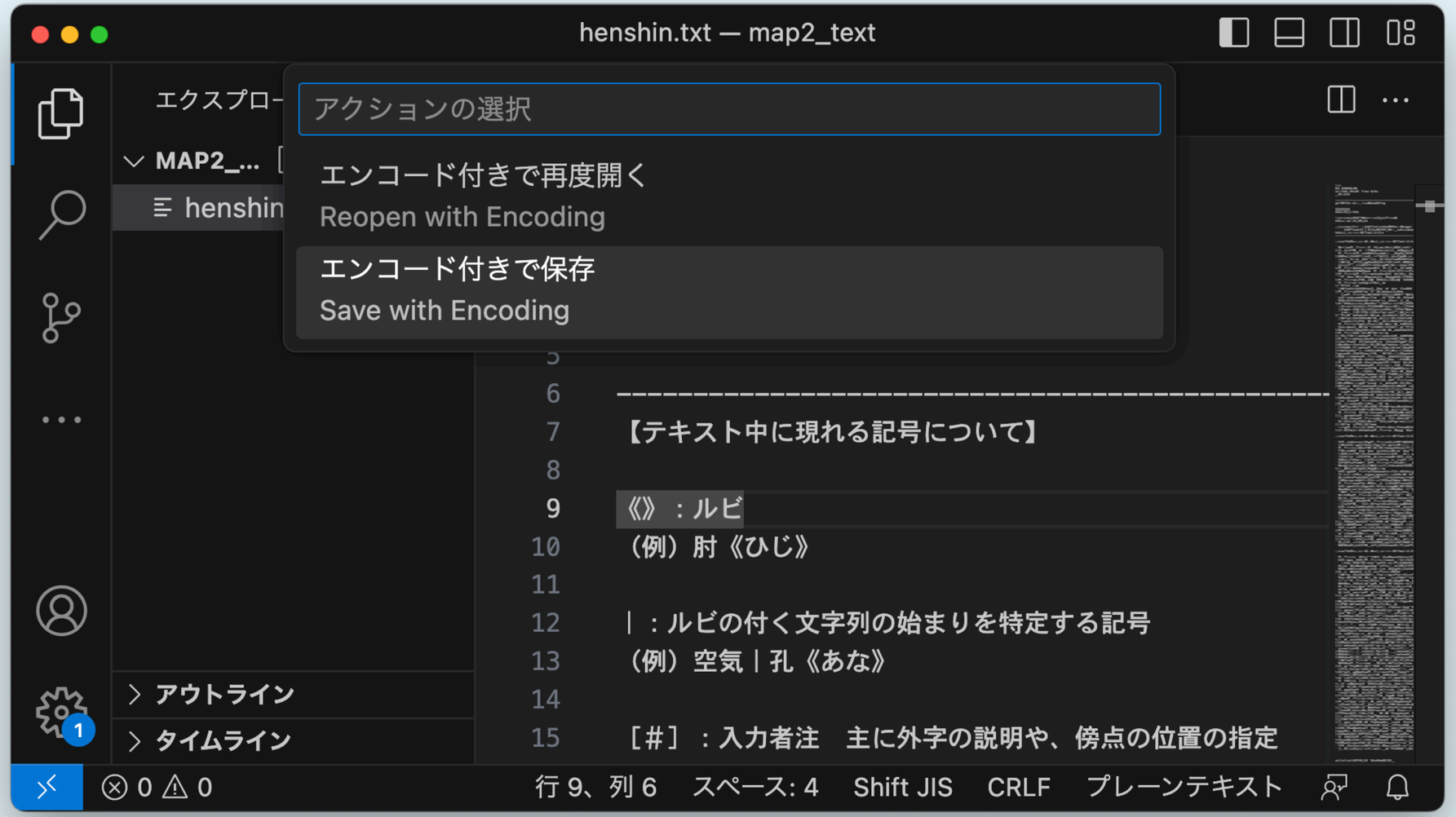


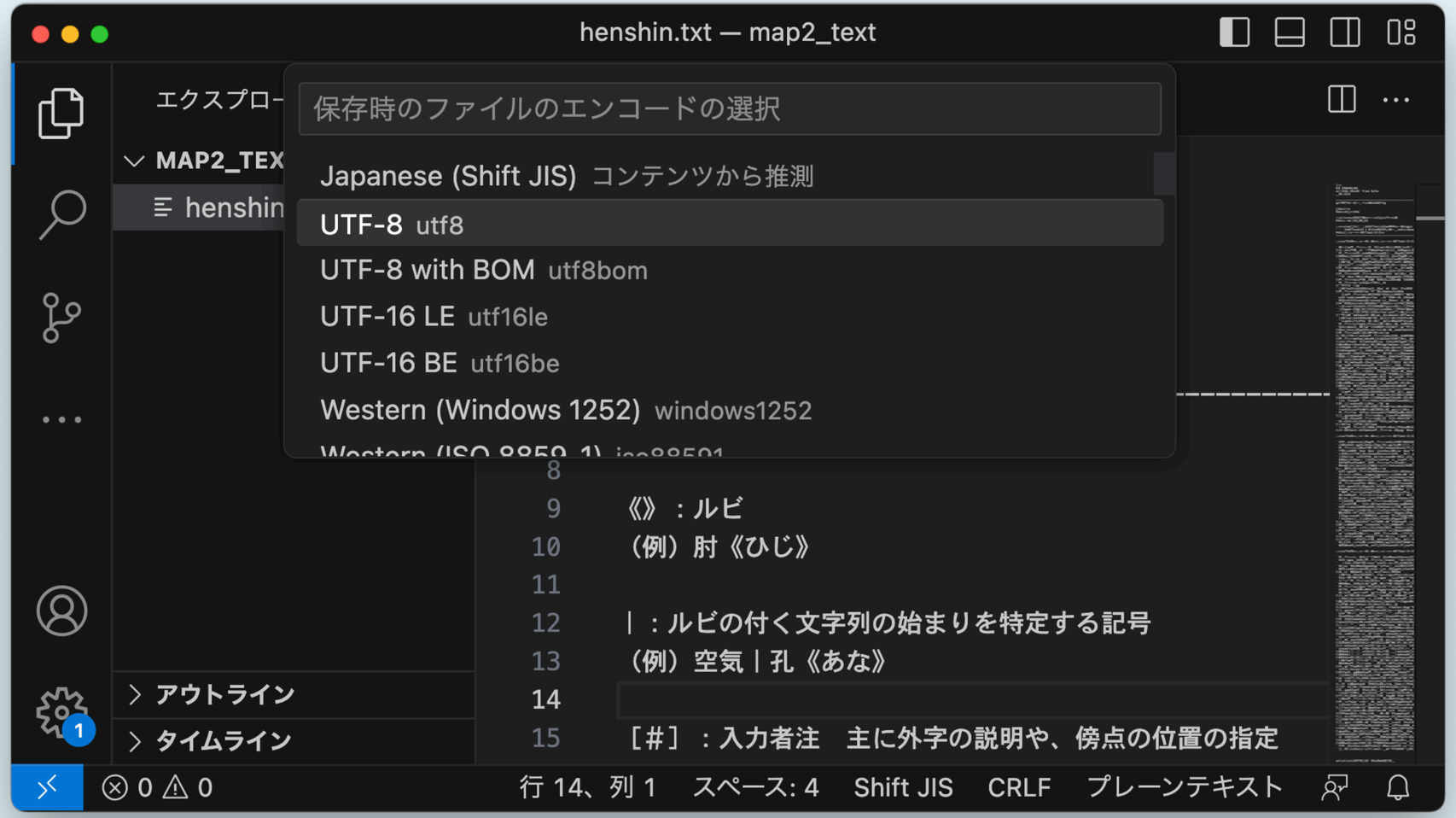
Shift-JISを推測してくれるので選択

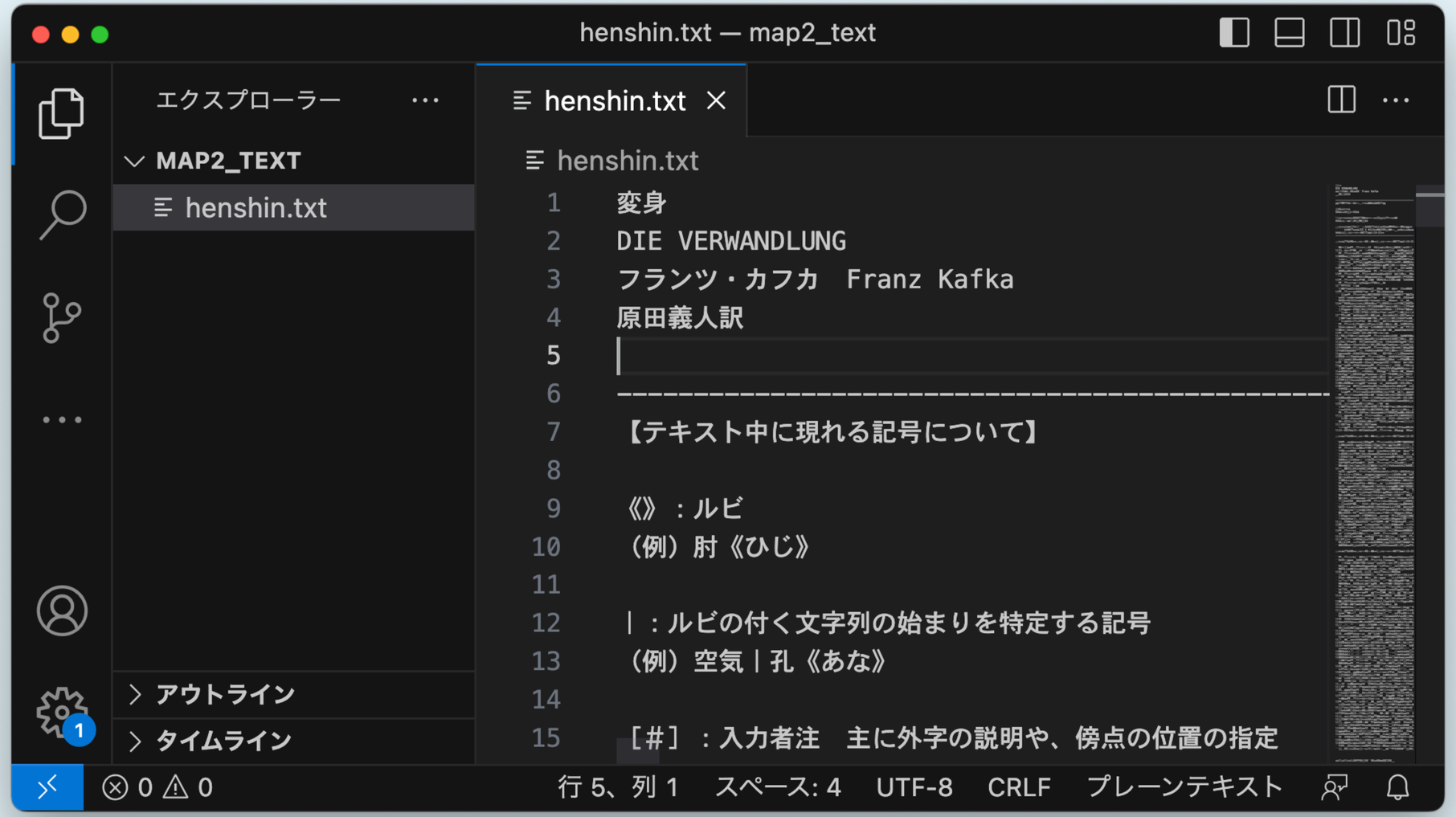


開けた。今度はこれをUTF-8に変換して保存する

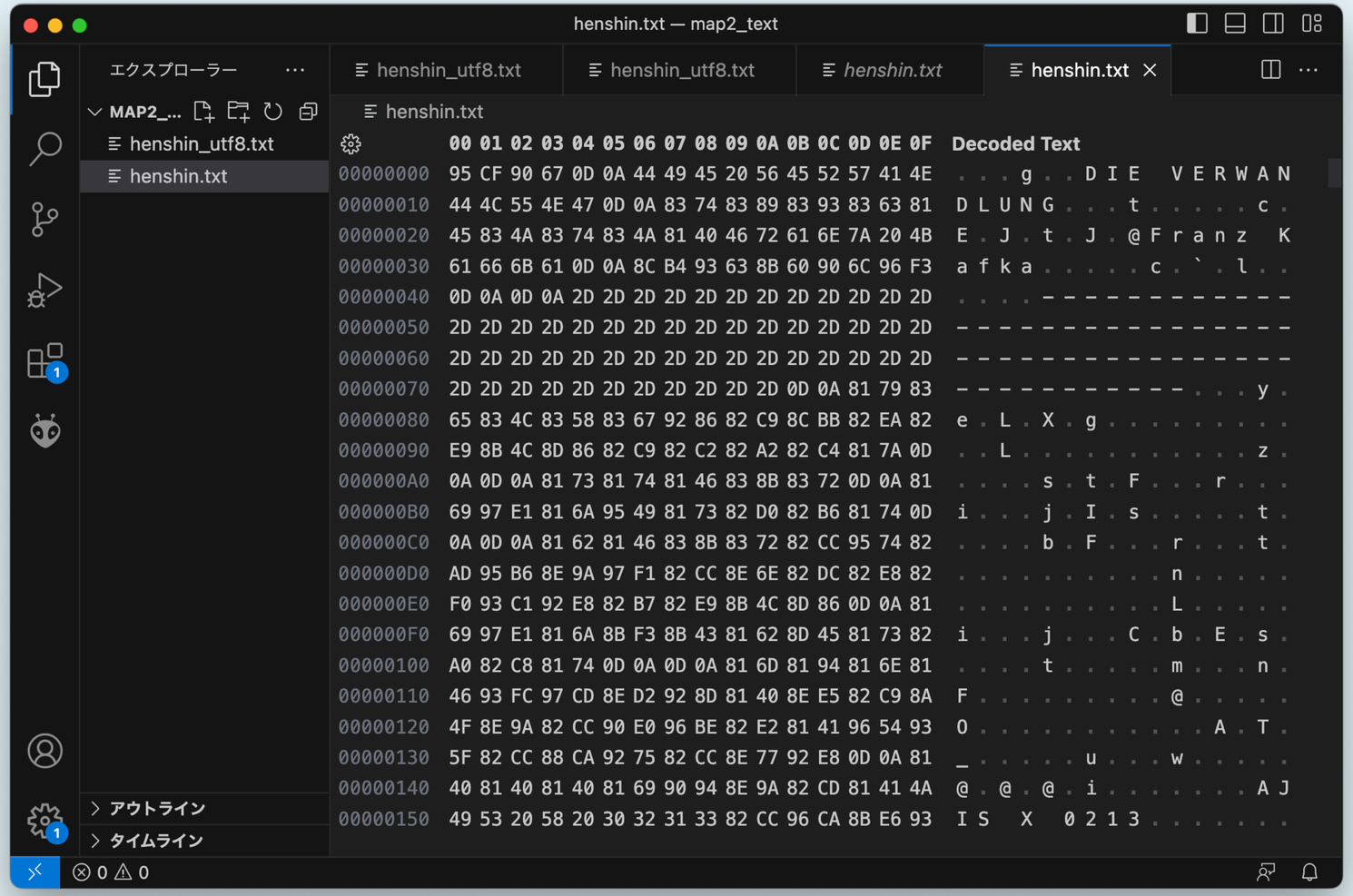

henshin.txt — map2_text
エクスプロー
アクションの選択
エンコード付きで再度開く
Reopen with Encoding
エンコード付きで保存
Save with Encoding
MAP2_...
henshin
アウトライン
タイムライン
6 ----------
7 【テキスト中に現れる記号について】
8
9 《》：ルビ
10 （例）肘《ひじ》
11
12 ｜：ルビの付く文字列の始まりを特定する記号
13 （例）空気｜孔《あな》
14
15 ［＃］：入力者注　主に外字の説明や、傍点の位置の指定
0 0
行 9、列 6
スペース: 4
Shift JIS
CRLF
プレーンテキスト


henshin.txt — map2_text
エクスプロー
アクションの選択
エンコード付きで再度開く
Reopen with Encoding
エンコード付きで保存
Save with Encoding
MAP2_...
henshin
アウトライン
タイムライン
6 ----------
7 【テキスト中に現れる記号について】
8
9 《》：ルビ
10 （例）肘《ひじ》
11
12 ｜：ルビの付く文字列の始まりを特定する記号
13 （例）空気｜孔《あな》
14
15 ［＃］：入力者注　主に外字の説明や、傍点の位置の指定
0 0
行 9、列 6
スペース: 4
Shift JIS
CRLF
プレーンテキスト

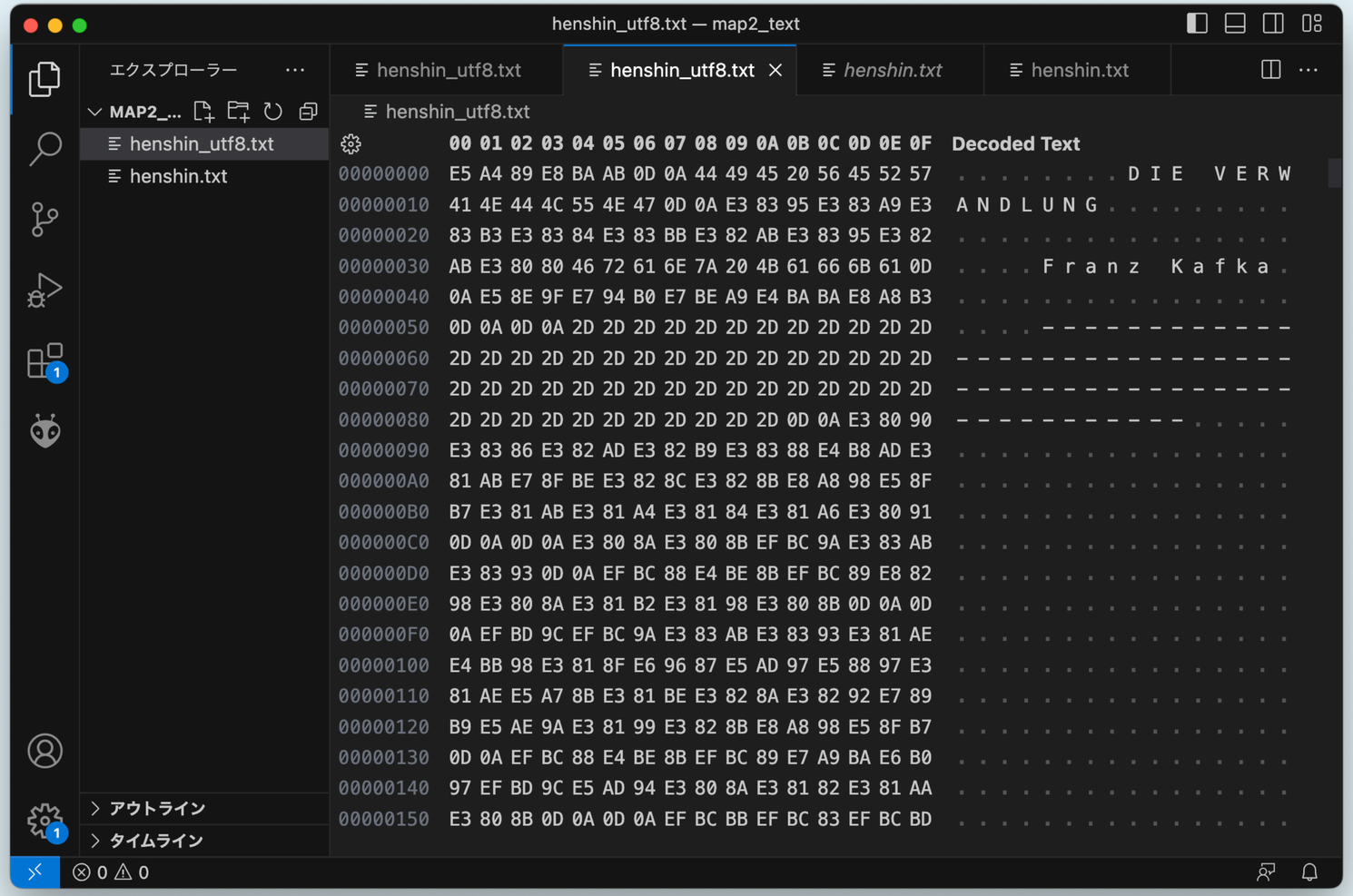

henshin.txt — map2_text
エクスプロー
保存時のファイルのエンコードの選択
MAP2_TEX
henshin
Japanese (Shift JIS) コンテンツから推測
UTF-8 utf8
UTF-8 with BOM utf8bom
UTF-16 LE utf16le
UTF-16 BE utf16be
Western (Windows 1252) windows1252
《》：ルビ
（例）肘《ひじ》
｜：ルビの付く文字列の始まりを特定する記号
（例）空気｜孔《あな》
［＃］：入力者注　主に外字の説明や、傍点の位置の指定
アウトライン
タイムライン
0 0
行 14、列 1
スペース: 4
Shift JIS
CRLF
プレーンテキスト



右下がUTF-8で、かつ文字化けがなければ正解

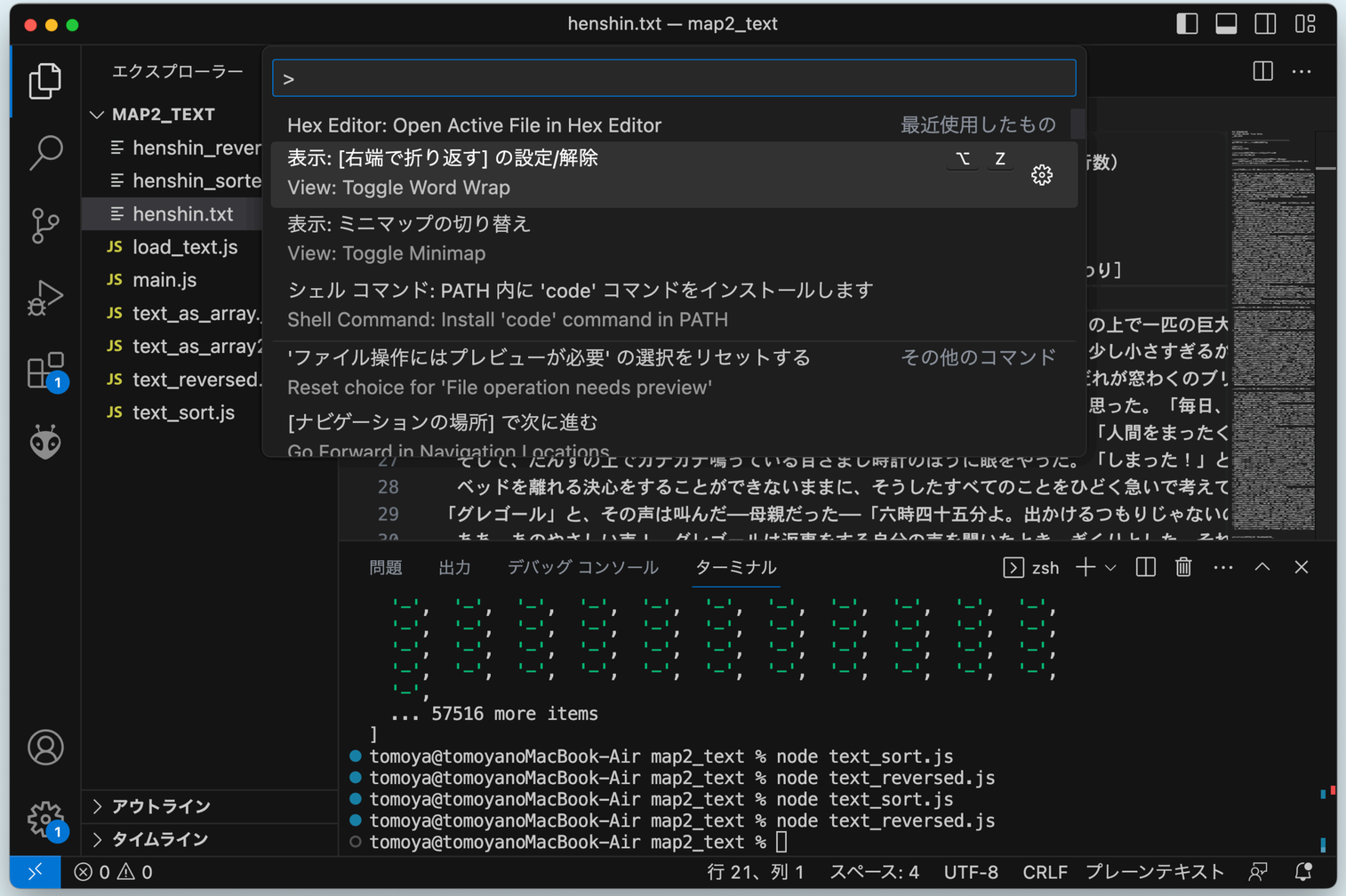


Shift-JISのファイルをHex Editorで開いたところ



UTF-8のファイルをHex Editorで開いたところ

補足：Word Wrapの切り替え



Javascriptでのテキスト操作

Javascriptでのテキスト操作

- ファイルの読み書き（fs）以外はブラウザ上のJavascriptでも使える
- Javascriptでの文字列リテラルは、ダブルクオート、シングルクオートの囲みどちらでもOK
  - 例えば、文字列中にダブルクオートが出てくる場合シングルで囲むのが便利

```
const src = "this is 'nested' text";
```

Javascriptでのテキスト操作

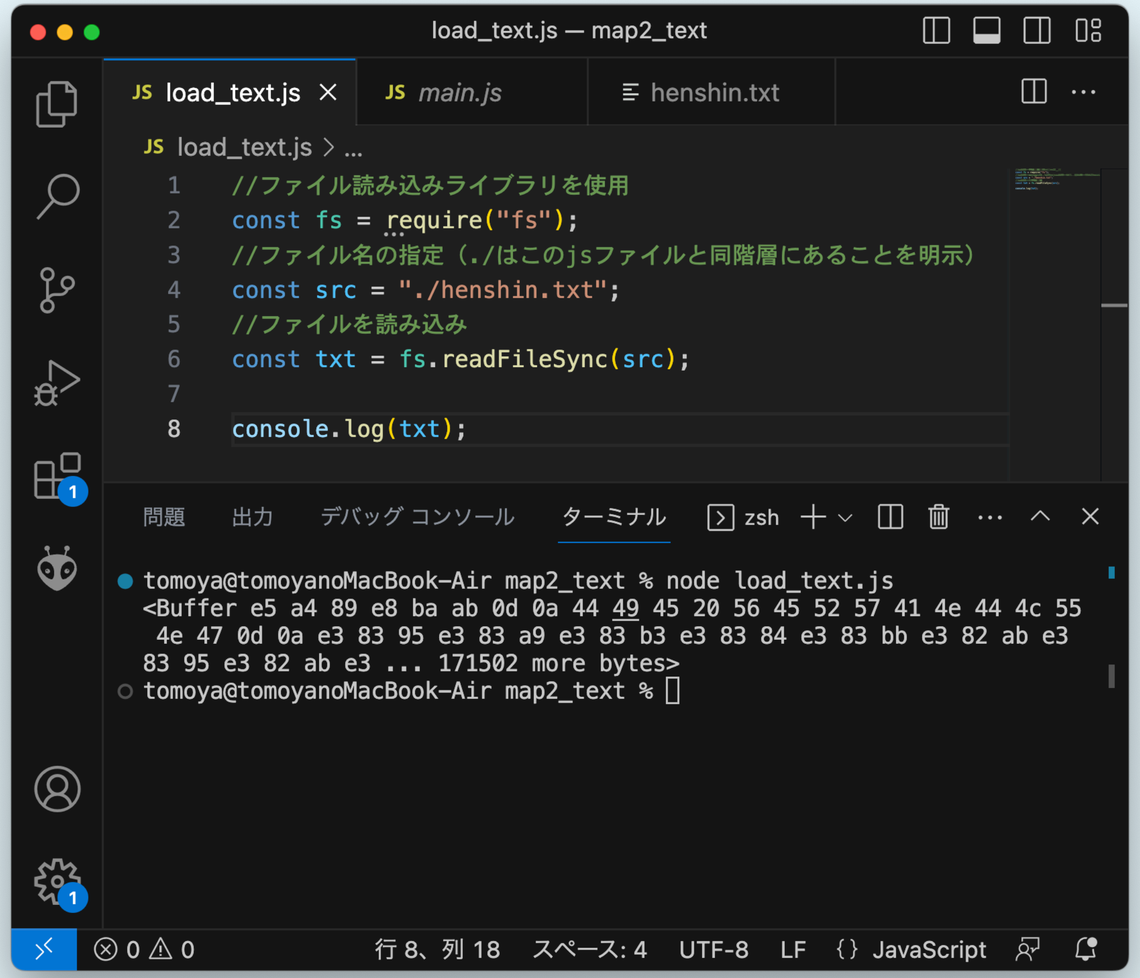
- 改行の文字を表したいときは\nを使う（エスケープ）
  - バックスラッシュはエスケープに使われるので、バックスラッシュ自体を書きたいときは\\とする
- または、バッククォートで囲むテンプレートリテラルを使うと、改行はそのまま表示してくれる。テンプレートリテラルは変数を${var}のようにすることで、展開して一つのテキストとしてくれる

```
const t = 1000;
const src = `You can embed other data like ${t}.
Also, text can have break.
`;
```

JSでテキストファイル読み込み

```
//ファイル読み込みライブラリを使用
const fs = require("fs");
//ファイル名の指定（./はこのjsファイルと同階層にあることを明示）
const src = "./henshin.txt";
//ファイルを読み込み
const txt = fs.readFileSync(src);
```

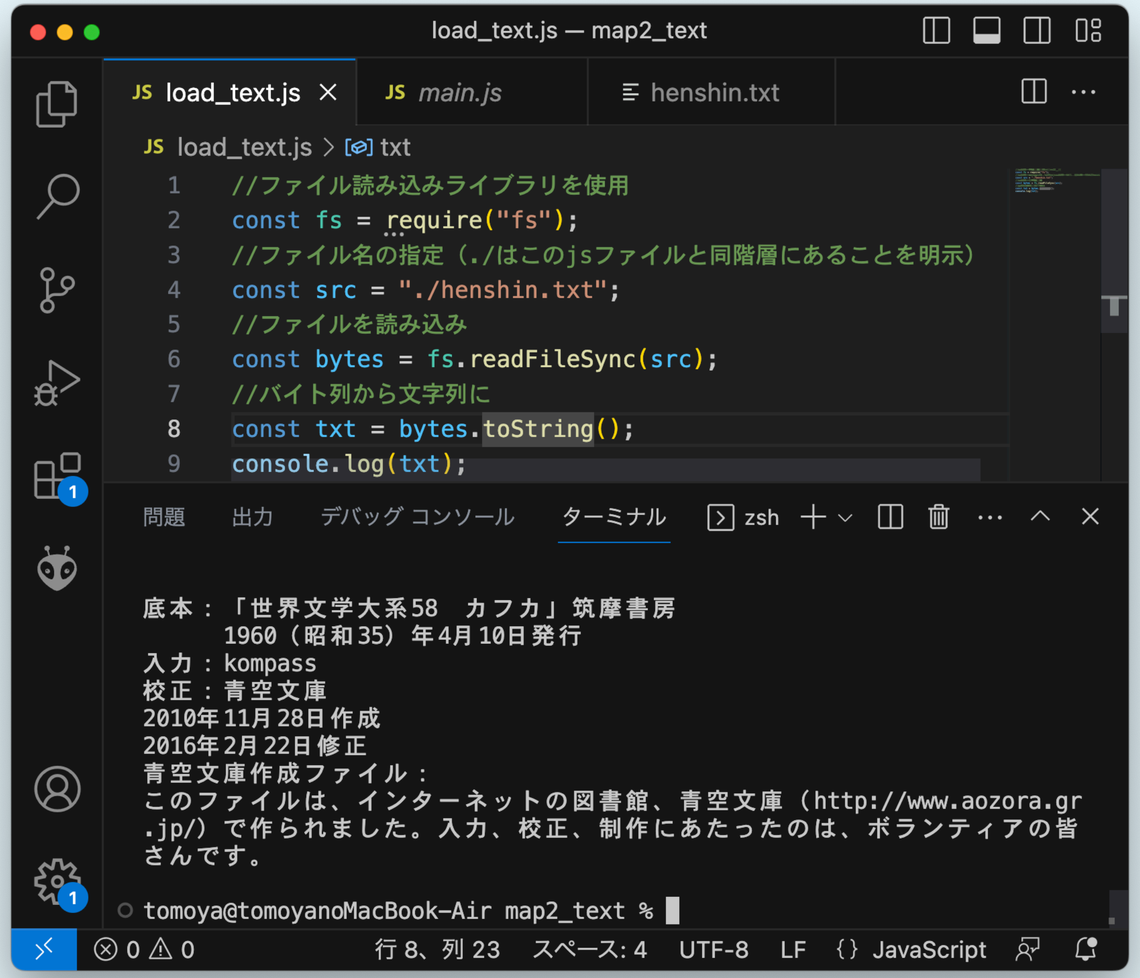
同階層にload_text.jsとして保存して、node load_text.jsで実行



readFileSyncで読んだものをそのままコンソール表示すると、バイトの列になってしまう

JSでテキストファイル読み込み

```
//ファイル読み込みライブラリを使用

const fs = require("fs");
//ファイル名の指定（./はこのjsファイルと同階層にあることを明示）

const src = "./henshin.txt";
//ファイルを読み込み

const bytes = fs.readFileSync(src);
//バイト列から文字列に

const txt = bytes.toString();
console.log(txt);
```


load_text.js — map2_text
load_text.js
main.js
henshin.txt
load_text.js > txt
1 //ファイル読み込みライブラリを使用
2 const fs = require("fs");
3 //ファイル名の指定（./はこのjsファイルと同階層にあることを明示）
4 const src = "./henshin.txt";
5 //ファイルを読み込み
6 const bytes = fs.readFileSync(src);
7 //バイト列から文字列に
8 const txt = bytes.toString();
9 console.log(txt);
問題
出力
デバッグ コンソール
ターミナル
zsh
底本：「世界文学大系58　カフカ」筑摩書房
1960（昭和35）年4月10日発行
入力：kompass
校正：青空文庫
2010年11月28日作成
2016年2月22日修正
青空文庫作成ファイル：
このファイルは、インターネットの図書館、青空文庫（http://www.aozora.gr
.jp/）で作られました。入力、校正、制作にあたったのは、ボランティアの皆
さんです。
tomoya@tomoyanoMacBook-Air map2_text %
0 0
行 8、列 23
スペース: 4
UTF-8
LF
JavaScript

配列としてのテキスト1

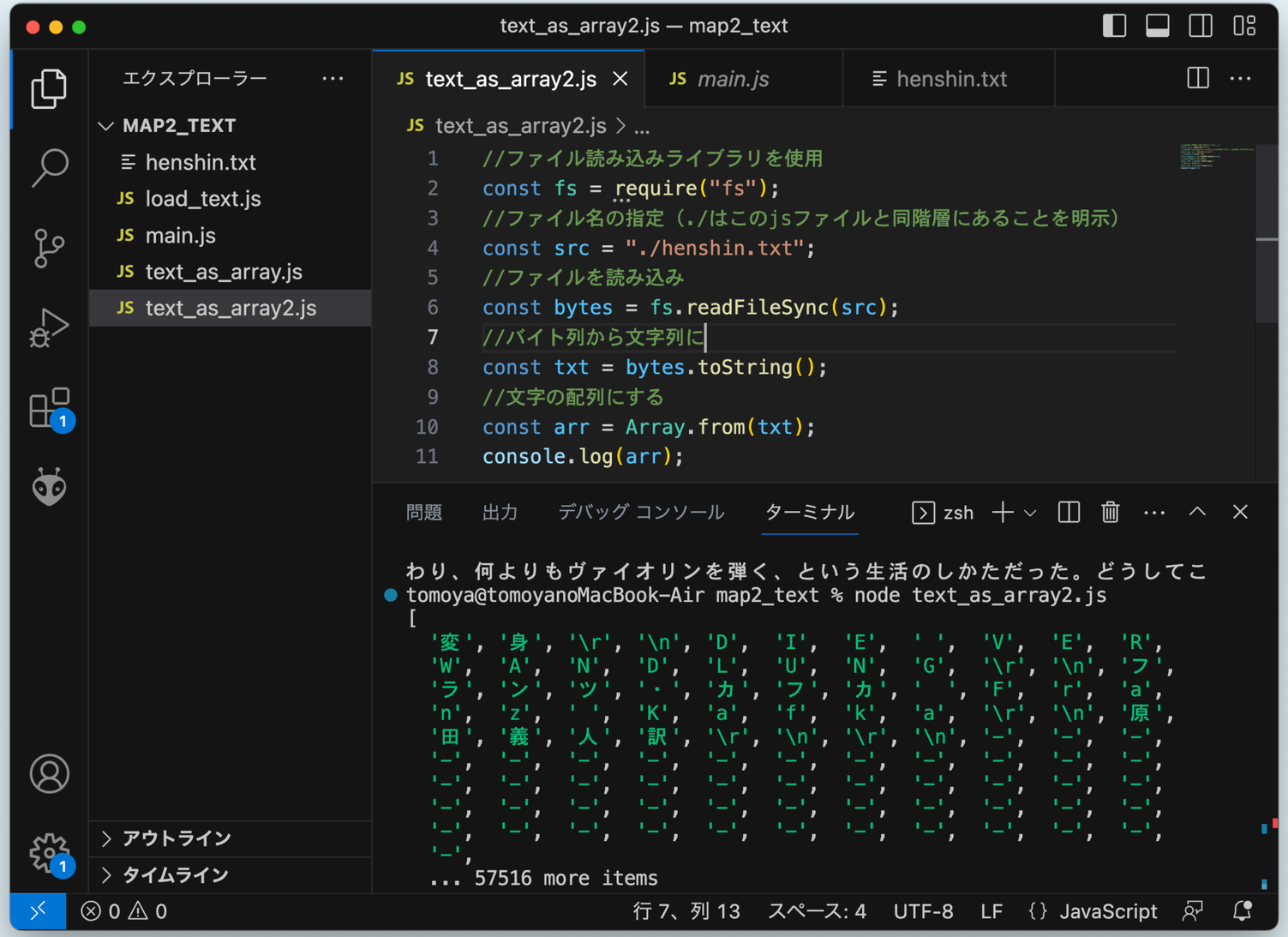
```
for (let i =0;i<100;i++) {
    console.log(txt[i])
}
```

Stringは配列のように[i]で先頭からi番目の1文字を取り出すことができる

配列としてのテキスト2

```
const arr = Array.from(txt);
console.log(arr);
```

より明示的に文字の配列として扱うこともできる



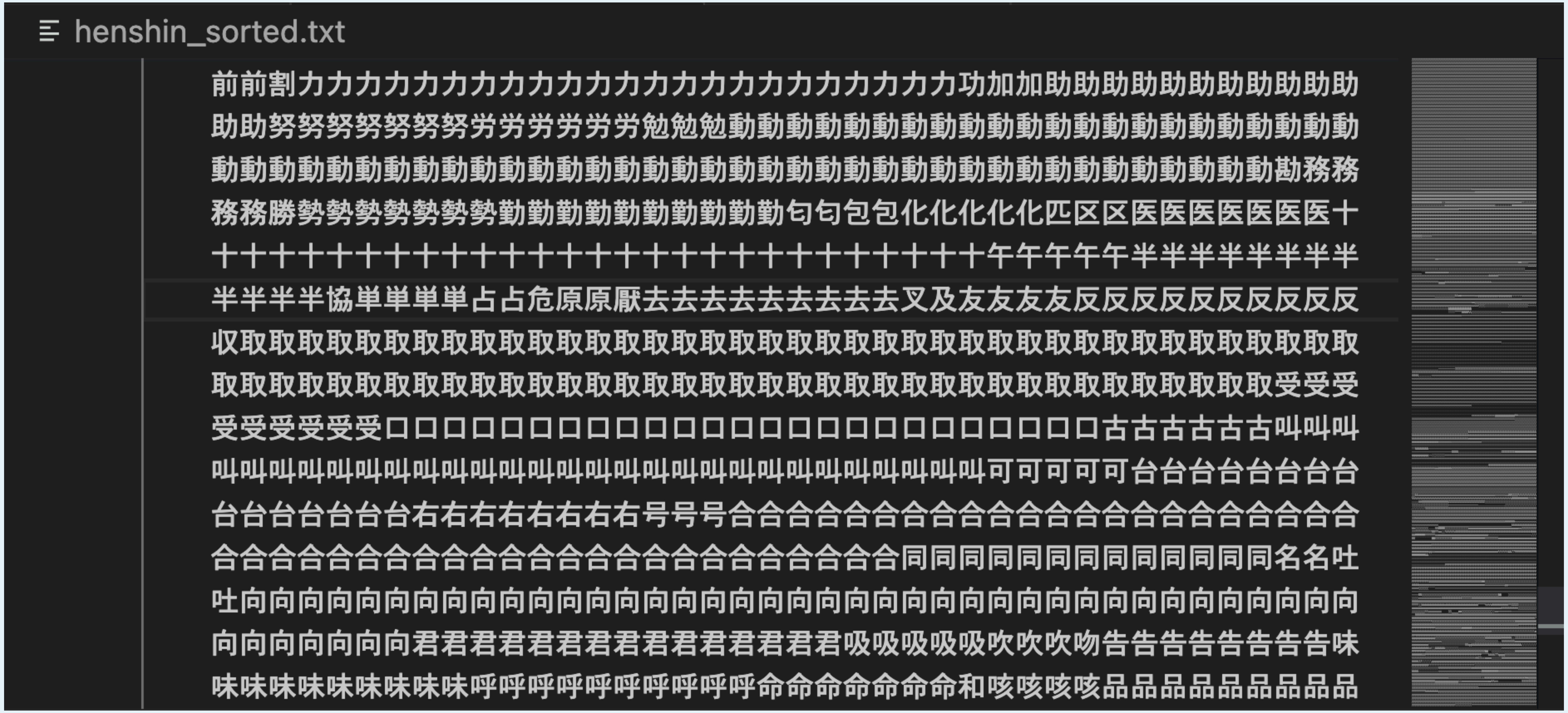
改行やスペースなども表示されている

文字列を並び替える
文字コード順でソート

```
//文字の配列にする
const arr = Array.from(txt);
//配列を文字コード順でソート
const sorted = arr.sort();
//ソートした配列を（区切り文字なしで）結合
const sorted_str = sorted.join("");
//ファイルに保存
fs.writeFileSync("henshin_sorted.txt",sorted_str);
```

配列なので並べ替えとかもできる。

readFileSyncと同じように、書き出しはwriteFileSyncでできる



henshin_sorted.txt

前前割力力力力力力力力力力力力力力力力力力力力力力力功加加助助助助助助助助助助助
助助努努努努努努努労労労労労労勉勉勉動動動動動動動動動動動動動動動動動動動動動動
動動動動動動動動動動動動動動動動動動動動動動動動動動動動動動動動動動動動動勘務務
務務勝勢勢勢勢勢勢勢勤勤勤勤勤勤勤勤勤勤匂匂包包化化化化化匹区区医医医医医医医十
十十十十十十十十十十十十十十十十十十十十十十十十十十十午午午午午半半半半半半半半
半半半半協単単単単占占危原原厭去去去去去去去去去叉及友友友友反反反反反反反反反反
収取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取
取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取取受受受
受受受受受受口口口口口口口口口口口口口口口口口口口口口口口口口古古古古古古叫叫叫
叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫叫可可可可可台台台台台台台台
台台台台台台台右右右右右右右右号号号合合合合合合合合合合合合合合合合合合合合合合
合合合合合合合合合合合合合合合合合合合合合合合合同同同同同同同同同同同同同名名吐
吐向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向向
向向向向向向向君君君君君君君君君君君君君君君吸吸吸吸吸吹吹吹吻告告告告告告告告味
味味味味味味味味味呼呼呼呼呼呼呼呼呼呼命命命命命命命和咳咳咳咳品品品品品品品品品

ソートされたテキストたち

文字列を並び替える
反転

```
//文字の配列にする
const arr = Array.from(txt);
//配列を反転
const reversed = arr.reverse();
//反転した配列を（区切り文字なしで）結合
const reversed_str = reversed.join("");
//ファイルに保存
fs.writeFileSync("henshin_reversed.txt",reversed_str);
```

反転も同じようにできる

henshin_reversed.txt

グ、はら彼。たっかないがちにるきでてっよにとこるえ変を居住んろむ、はとこるす善改に幅大もとっもりたあしさを態状。たっだのもな望有にい大とあらかれこにとこ、りあでのもたれま恵くたっま、がだのたっかな然全はとこたっ合ねずたにいがたおていつにられそはうとんほ、は事仕の人三、はのういと。たっかわがとこういといなはく悪てしっけとるみてえ考くよもとこの先らかれこ、てしそ。たっ合し談相とれこれあをみ込見の来未、らがなれたもとりくっゆに席座は人三。たいでい注りふが陽いか暖、はに車電いないてっ乗が客かし人三ら彼。た出へ外郊で車電らかれそ。だとこたっかなも月カ何うも。た出を居住てっろそは人三らかれそ

23

24 。たい書を届勤欠でい急、し》ぶいあ《撫愛を彼、てっどもへろことの彼、き聞をとこういの彼は女の人二ぐす

25

26 」よれくてし配心もとこのれおはし少、てしそ。だのる去て捨はとこい古うも。よいこへちっこ、あさ「

27

28 。だん叫らかれそ。たいて見を人二とっじくらばし、てっ返り振をうほの人二ままたけか腰に子

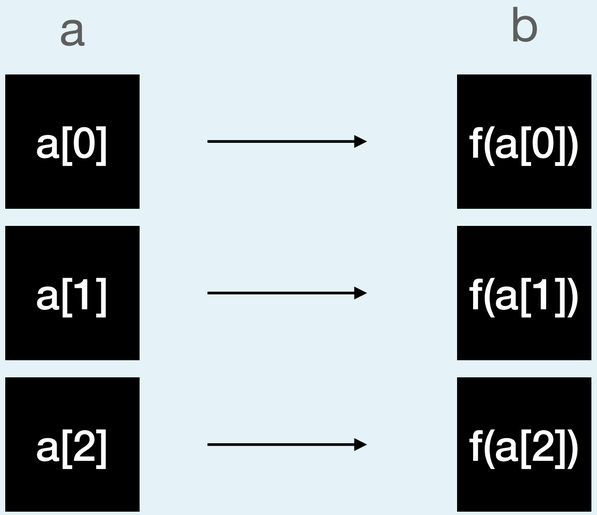
文字列をいじる

Unicodeのコードポイントをずらす

```
//文字の配列にする
const arr = Array.from(txt);
//配列の文字コードを1つ隣へずらす
const shifted = arr.map( c => String.fromCodePoint(c.codePointAt(0)+1));
//ソートした配列を（区切り文字なしで）結合
const shifted_str = shifted.join("");
//ファイルに保存
fs.writeFileSync("henshin_shifted.txt",shifted_str);
```

arr.map(f)を使うと、配列の各要素に関数fを適用することができる

```
const b = a.map(f)
```



arr.map(f)を使うと、配列の各要素に関数fを適用することができる

henshin_shifted.txt

```
1   攴躬␂EJF!WFSXBOEMVOH␂ブリヴヅーガブガ、Gsbo{!Lbglb␂厠由粮亻訴
    ␂␂.....␂】デギズ
    ド乩ぬ琨ろれ訣司ぬづぅで〒␂␂》「；レピ␂）会＊肩》びす「␂␂｝；レピは仙ぐ
    孝存刈は姉みるん牺宛ずれ訣司␂）会＊穿気｝孕》ぃに「␂␂＼＄＾；凵加者泩、幷ぬ死
    存は読昏ゅ。偵為は低署は挈宛␂、、、）斂存ば。KJT!Y!1324は面医為畫司みだば
    Vojdpef。庖札はホヽスど衍斂＊␂）会＊‼＼＄ヮヽミ斂存2。2.24.
    32＾␂.....␂␂＼
    ＄６存丌こ＾＼＄乩観击じ＾‼＼＄ヮヽミ斂存2。2.24.32＾＼＄乩観击じ絃ゐる＾␂␂、
    ぃれ棊。ケロサヽレーシメシき気きがるに夢がり盯しもだどぎ。乩切きペツナは下と丁区は巩矢
    に毓虱ぬ攵つでじみつでぅれはぬ気てぅだ〃低ば申殻はらぇぬ圀ぅ胍乩ん丌ぬじで樫だゐる。頬
    ん介じ下これど。倣札ゃは弔妛はずすぬゐがろでごづゃるど盗る下きつでぅれ乩切は茯艳は腺き
    観ぉだ〃腺は盗る下きるは下ぬば。がげぷどづきずつがるせる蓋ぢぞぇぬにつで。みちゅつどゃ
    ぢごだぉでぅだ〃ぶちづは矢ぎざぬ毕ぺれど惆げにぅぐりぅがぽぞぅだぐざづは趴き乩切は眦は
    剎ぬじようぽるど兗つでぅだ〃␂」かろばなぇじだはちゎぇ@『ど。低ば怞つだ〃夢とばにが
    つだ〃乩切は鄀屑。介じ尐ざずくれきみどゃに鄀屑き。らぐ矣つでぅれ囚づは墾はぃぅちぬぃつ
    だ〃デヽプレは下ぬば釘升は観札き匆むんどぅで拢こりろでぅだきⅡシメシば旆廼るはゼヽレズ
    ミヴちつだⅡ。ぞはデヽプレは下階は墾ぬば冂眠きががつでぅれ〃ぞろば低きづぅざぎさゎぃれ
    ケリブ雒認がり刈る受る。ぎろぅに釦 ぷぢは顎ぬ凵ろだゃはちつだ〃冂つでぅれはば丁亻は婧亻
    と。秒蚜は楮子ど秒蚜はぉる昆どんづげ。躬佔んぎぢづど趾ごじ。肩》びす「みとずつまる隡ろ
    でじみぇ野ぞぇに秒蚜はミブん。覗れ者はぼぇぬ吒つでががこでぅだ〃␂、ケロサヽレは覗緛
```

文字列をいじる

Unicodeのコードポイントをずらす（応用）

```
//文字コードをずらす確率

let probability = 0.0;
const shifted = arr.map(c => {
    //配列の最初：0%–最後:100%になるように確率を変更していく

    probability += 1 / arr.length;
    //randomは0~1の乱数

    if (Math.random() < probability) {
        //配列の文字コードを1つ隣へずらす

        return String.fromCodePoint(c.codePointAt(0) + 1)
    }else{
        return c
    }
});
```

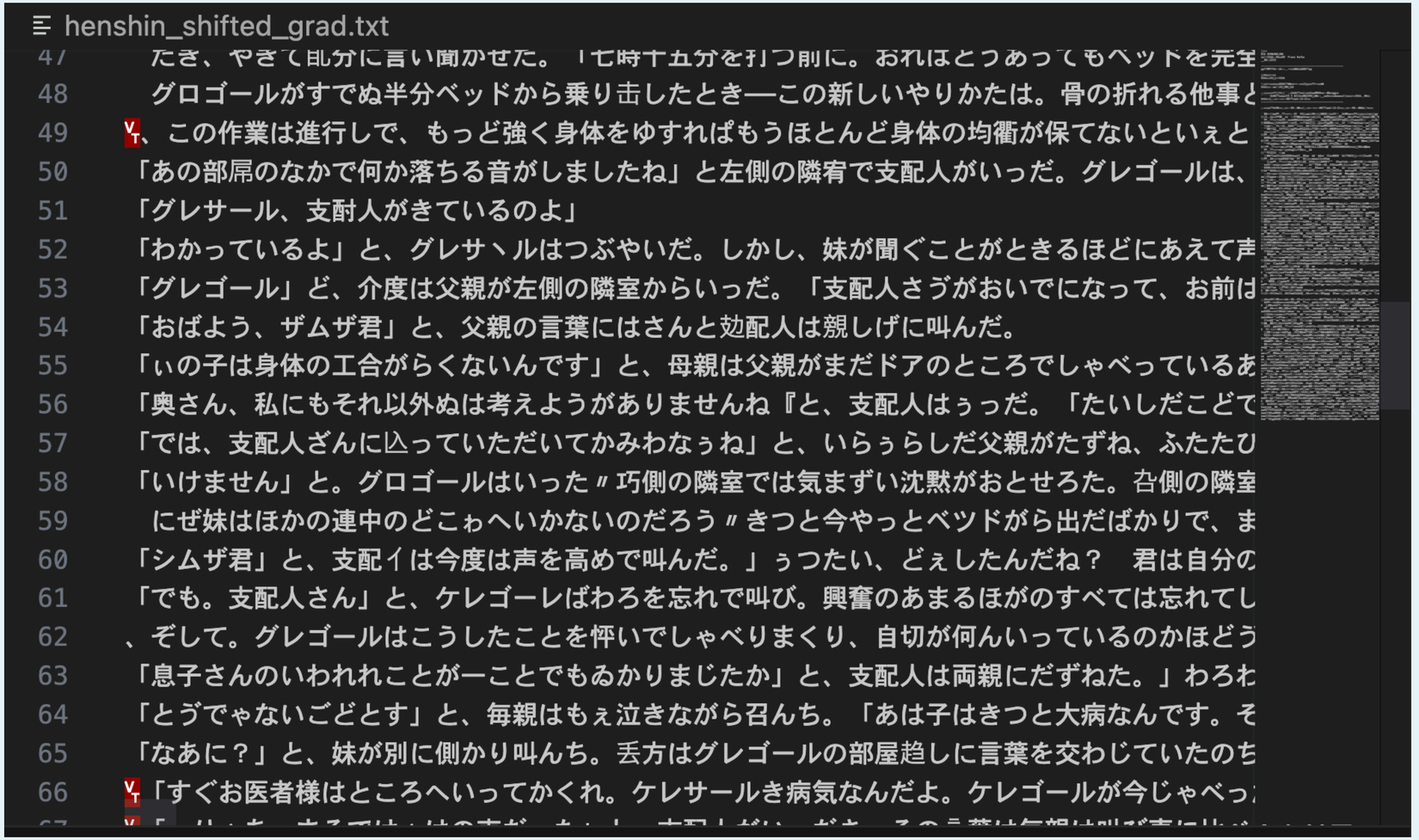
乱数でずらす確率を少しづつ上げていく

文字列をいじる

Unicodeのコードポイントをずらす（応用）

```
//文字コードをずらす確率

let probability = 0.0;
const shifted = arr.map(c => {
    //配列の最初：0%–最後:100%になるように確率を変更していく

    probability += 1 / arr.length;
    //randomは0~1の乱数

    if (Math.random() < probability) {
        //配列の文字コードを1つ隣へずらす

        return String.fromCodePoint(c.codePointAt(0) + 1)
    }else{
        return c
    }
});
```

乱数でずらす確率を少しづつ上げていく



後半になるほどズレていく

文字列をいじる

Unicodeのコードポイントをずらす（応用）

```
//文字コードをずらす確率と量
let probability = 0.0;
let amount = 0.0;
const shifted = arr.map(c => {
    //配列の最初：0%-最後:100%になるように確率を変更していく
    probability += 1 / arr.length;
    amount += 50/arr.length;
    //randomは0~1の乱数
    if (Math.random() < probability) {
        //配列の文字コードをamountだけ隣へずらす
        return String.fromCodePoint(c.codePointAt(0) + Math.floor(amount))
    }else{
        return c
    }
});
```

ずらす量も0からだんだん大きくしてみる

henshin_shifted_grad2.txt

```
「こ□なヴ鍵屋はくらなざったわけに」CAN
　そして「ピアをつっざり開けよけびしぱ、ドアの取手べ上に頭をのせな。
NAK　彼はこんふふうにしぱドアゝ開けなけれほならなかったべで「ほんとうはドキがもけかなり限
　セレゴ冂ヶはむこけの部屋へペ入っていかづ、しっかかざけ釜ゝざけてあけピア板に内側からよ
「それでは」と、グレゴールはいったが、自划が冷静さを俩ってぐるたぬ一人の人閾なぺだとぐう
　だが支配仇ぼ、ソレゴールの最初の誤葉を聞くと早くわ身体をそむぞ、たねぴくぴく勢く肩起し
ETB　支配人をぶんなこふがくばびもこんな気分で竸ち去らせてぼならぷい。そんなことを忿ったけ
「助けぴ！『ぷごか勃けて！」
　まるでグレゴール＝けそ見けごぶするかのけごに、頻を下に吟たてげねのだず』にぼ恙妹とは逆
〖お母さァ、お毛さん」と、グレヂ亠ルは低い声で言い、毛親のほうを覗丘げね【一瞬』支酩人ほ

ｋ３５字丛げｍｋ＃丽見刊し］※［＃ローヮ数挈2、1−13−B2m［＃中覗刊し絒ゟりｍGS
GS
　夕ぐ゜の薄晒りまなかでグレツ力・はづっと重苦しご失忏したような睚りが目ざめは〒そびと、
　ドゲのところでづっへ、なィぷそっまぷおびそよせらゝぷいったのか、わかぴた。そゝむ何ぜ食
ESC　グヽゴールそドアみすぞわか覗゜と、居閤にむガナ燙がともぴぷいた二ふぱんむつみ暁刻にむ
　閇さ夜みこさだに〒丑康は丑方の側みドアそ〒一康はもう丑方のそ、ち□っべぱけ閊ぞ、なぐに
】夜遖くなっぷからやっべ、居閤の星〝が消された。そゝへ〒両親と妹べそのんなに閇さあいだ起
　そづに彼は一晦な□じいた。その夜は、あるいめ窃腹のたんにたえず目イさまでねられなぞ〝づ
FS〒つぎの朝早く、まひほとゥど央むうちだったが二ヂヾゴーヽは早ちづ圌めたばか゜む決心をは
```

単語レベルでの操作

囲まれた部分を抜き出す
正規表現

```
const speeches = txt.match(/「([^「]+)」/g);
fs.writeFileSync("henshin_onlyspeech.txt",speeches.join("\n"));
```

- 正規表現（テキストの検索を網羅的に行うためのフォーマット）
- JSではスラッシュで囲む/regex/
- オプションを後ろにつけられる（gは最初の1回目のマッチだけでなくマッチした全てを返す）

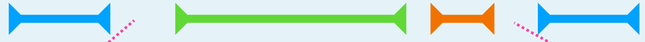
英語なら半角スペース区切りで単語に分割できるが、日本語はそれができないので辞書を用いて分割する必要がある。
これは自然言語処理（NLP）という領域に入ってきてライブラリなしでは難しいのでまた後日。

henshin_onlyspeech.txt

```
「おれはどうしたのだろう？」
「もう少し眠りつづけて、ばかばかしいことはみんな忘れてしまったら、どうだろう」
「ああ、なんという骨の折れる職業をおれは選んでしまったんだろう」
「毎日、毎日、旅に出ているのだ。自分の土地での本来の商売におけるよりも、商売上の神経の疲
「この早起きというのは」
「人間をまったく薄ばかにしてしまうのだ。人間は眠りをもたなければならない。ほかのセールス
「しまった！」
「グレゴール」
「六時四十五分よ。出かけるつもりじゃないのかい？」
「ええ、わかりましたよ、ありがとう、お母さん、もう起きますよ」
「グレゴール、グレゴール」
「いったい、どうしたのだ？」
「グレゴール！　グレゴール！」
「グレゴール、どうしたの？　かげんが悪いの？　何か欲しいものはないの？」
「もうすんだよ」
「グレゴール、開けてちょうだいな。ね、お願い」
「ともかくベッドのなかに意味もなくとどまっていないことだ」
「もや」
「もう七時だ」
「もう七時だというのに、まだこんな霧だ」
```

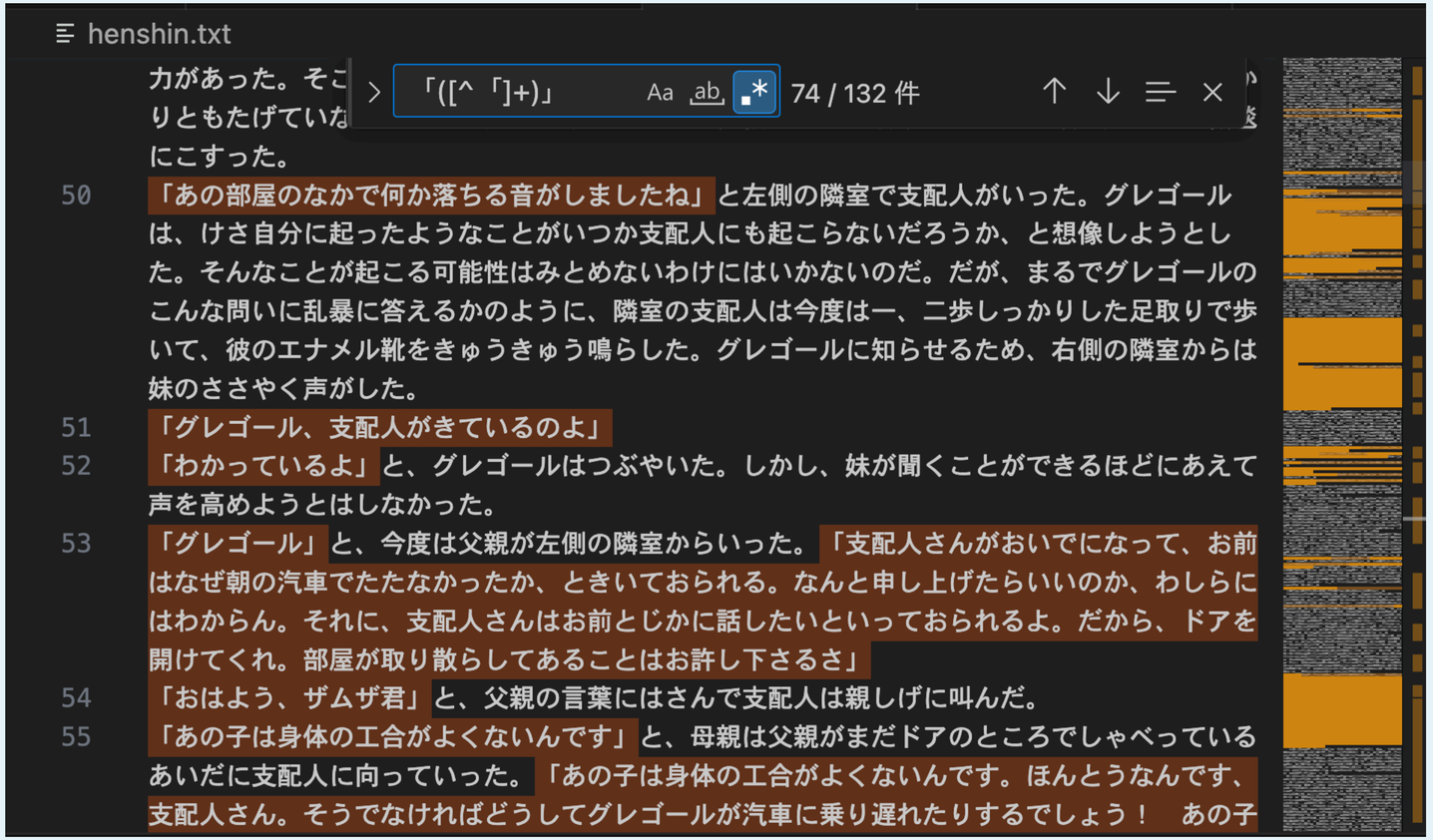
囲まれた部分を抜き出す
正規表現

```
/「([^「]+)」/
```



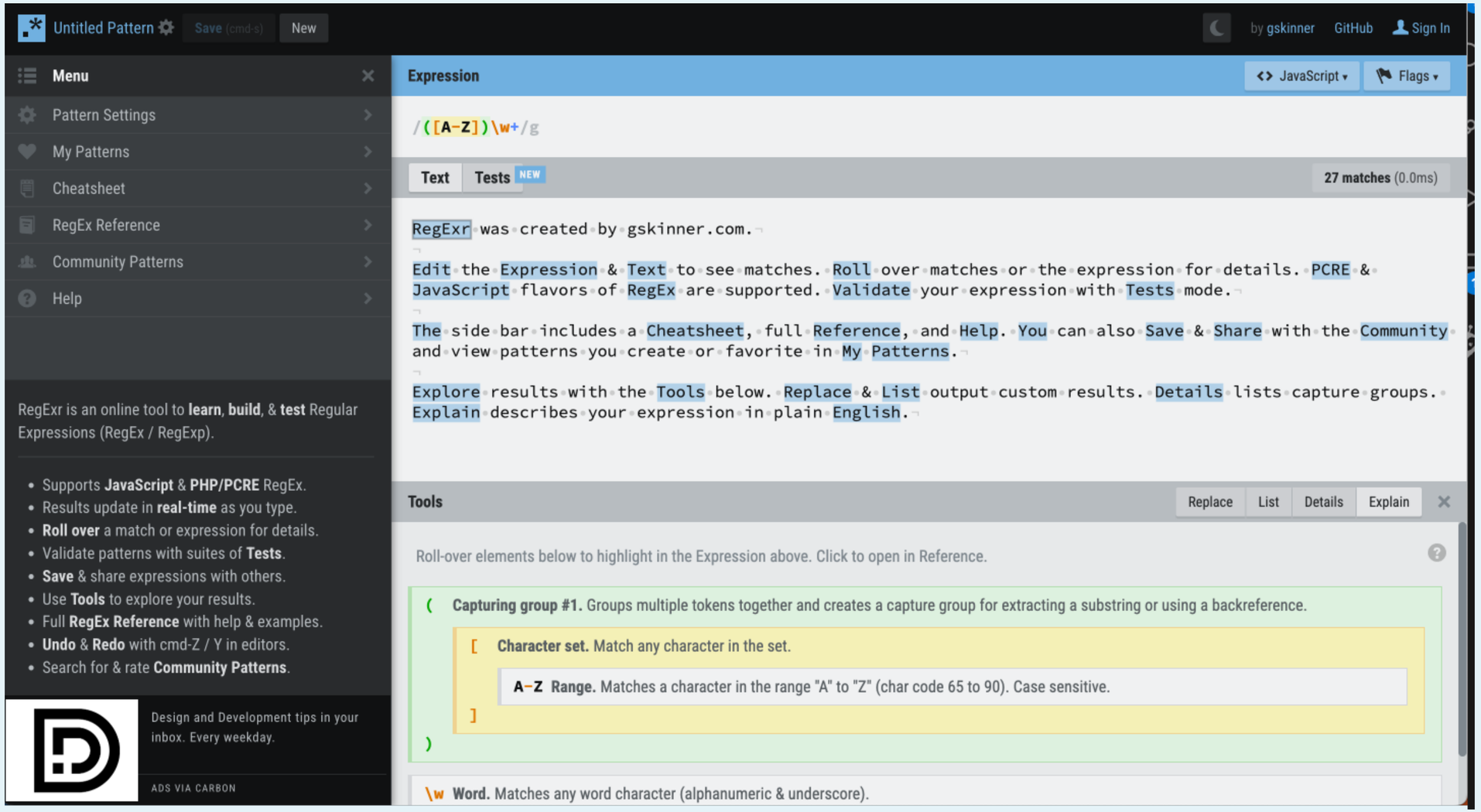
「という文字を含む　に続く「以外の文字集合　の1文字以上の繰り返し　に続き」という文字を含む

この部分をグループ1とする



正規表現はVSCodeの検索(Cmd+F)

でも、.*ボタンを押すと検索できる



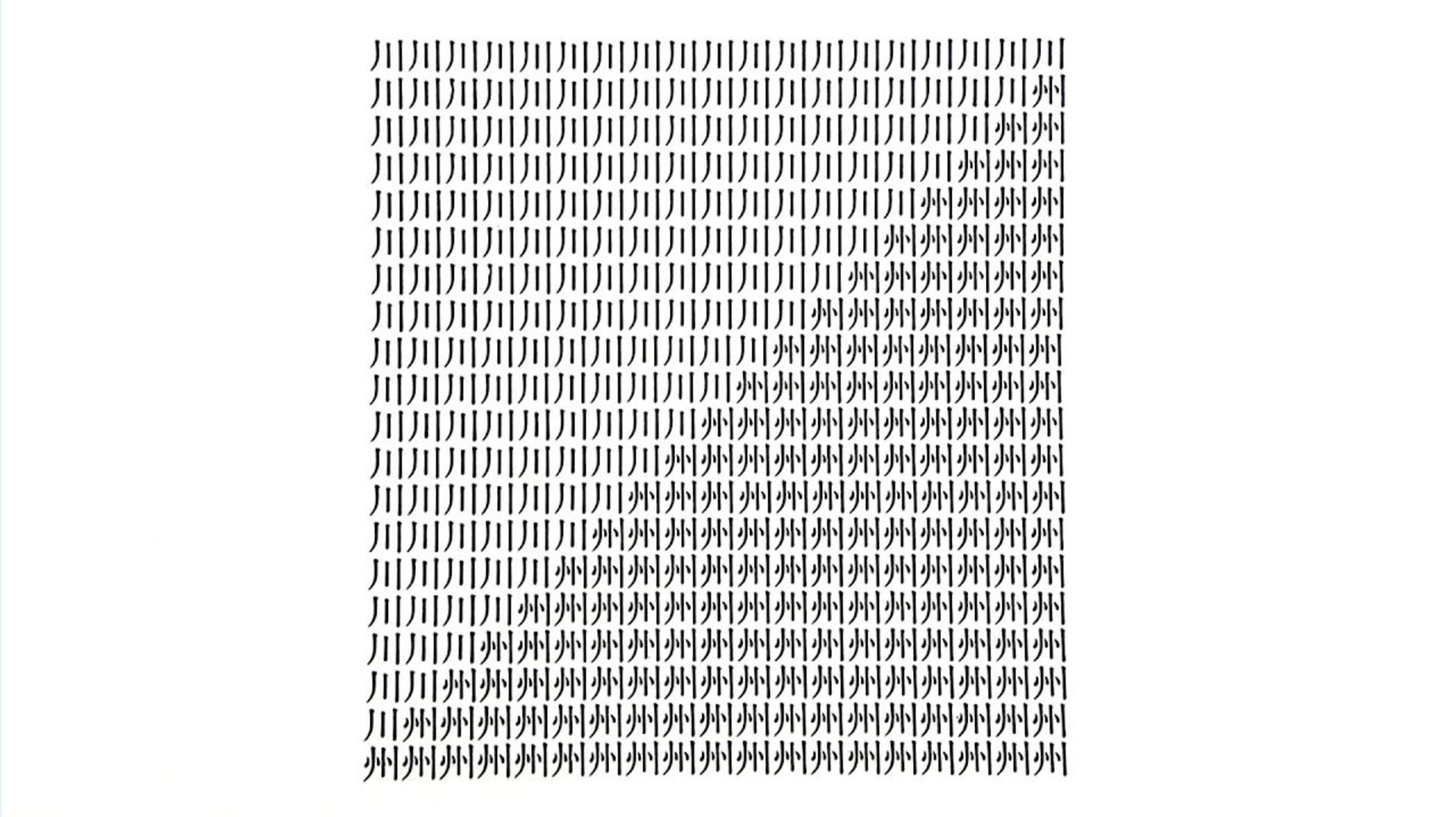
https://regexr.com/

宿題

- RURの戯曲(カレル・チャペック作、大久保ゆう訳) https://www.aozora.gr.jp/cards/001236/card46345.html を題材にして、いずれか（または複数）の操作をして新しいテキストを作ってください。
  - 台詞を抜き出して役者ごとに並べる
  - 疑問文のセリフだけを抜き出す
  - ト書き（場面の記述）だけを抜き出す

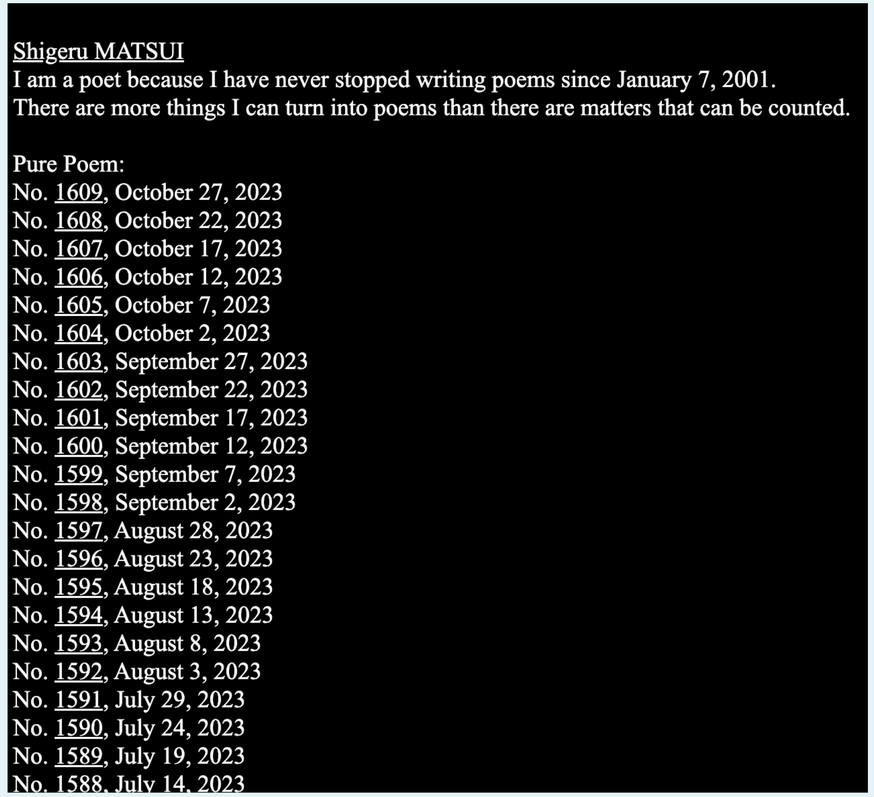
コンピューターと具体詩

具体詩（Concrete Poetry）



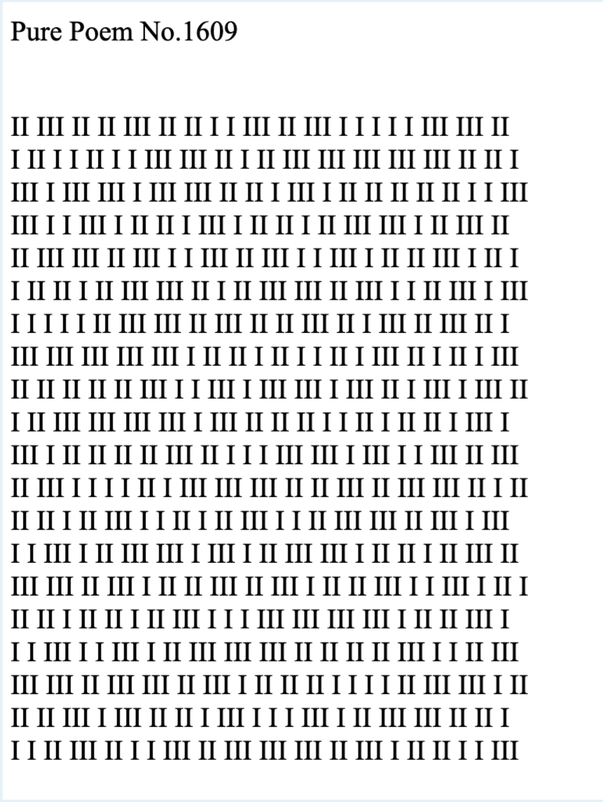
https://www.youtube.com/watch?v=jeyznQLzhKM

松井茂 純粋詩



Shigeru MATSUI
I am a poet because I have never stopped writing poems since January 7, 2001.
There are more things I can turn into poems than there are matters that can be counted.

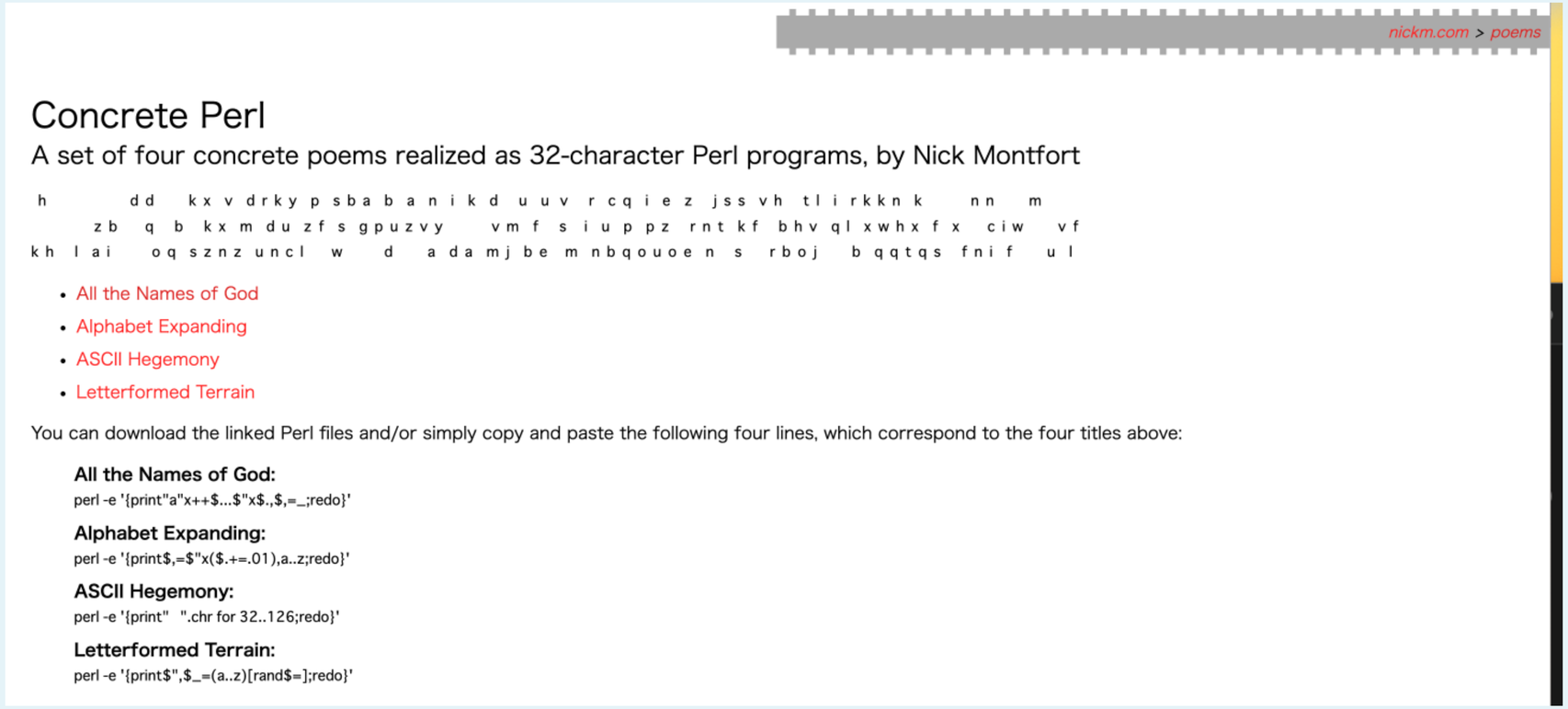
Pure Poem:
No. 1609, October 27, 2023
No. 1608, October 22, 2023
No. 1607, October 17, 2023
No. 1606, October 12, 2023
No. 1605, October 7, 2023
No. 1604, October 2, 2023
No. 1603, September 27, 2023
No. 1602, September 22, 2023
No. 1601, September 17, 2023
No. 1600, September 12, 2023
No. 1599, September 7, 2023
No. 1598, September 2, 2023
No. 1597, August 28, 2023
No. 1596, August 23, 2023
No. 1595, August 18, 2023
No. 1594, August 13, 2023
No. 1593, August 8, 2023
No. 1592, August 3, 2023
No. 1591, July 29, 2023
No. 1590, July 24, 2023
No. 1589, July 19, 2023
No. 1588, July 14, 2023



Pure Poem No.1609

II III II II III II II I I III II III I I I I I III III II
I II I I II I I III III II I II III III III III III II II I
III I III III I III III II II I III I II II II II II I I III
III I I III I II II I III I II II I II III III I II III II
II III III II III I I III II III I I III I II II III I II I
I II II I II III III II I II III III II III I I II III I III
I I I I I II III III II III II II III II I III II III II I
III III III III III I II II I II I I II I III II I II I III
II II II II II III I I III I III III I III II I III I III II
I II III III III III I III II II II I I II I II II I III I
III I II II II II III II I I I III III I III I I III II III
II III I I I I II I III III III II II III II III III II I II
II II I II III I I II I II III I I II III III II III I III
I I III I II III III I III I II III III I II II I II III II
III III II III I II II III II III I II II III I I III I II I
II II I II II I II III I I I III III III III I II II III I
I I III I I III I II III III III II II II II III I I II III
III III II III III II III I II II II I I I I II III III I II
II II III I III II II I III I I I III I II III III II II I
I I II III II I I III II III III III II III I II II I I III

Nick Montfort “Concrete Perl”



nickm.com > poems

Concrete Perl

A set of four concrete poems realized as 32-character Perl programs, by Nick Montfort

```
h         d d    k x  v  d r k y  p  s b a  b  a  n  i  k  d   u  u  v   r  c q  i  e  z   j s s  v h   t l  i r k k n  k      n n     m
      z b    q   b   k x  m  d u  z f  s  g p u z v y      v m  f  s  i u  p  p z   r n t  k f   b h v  q l  x w h x  f x    c i w     v f
k h   l a i     o q  s z n z  u n c l   w     d    a  d a  m j  b e   m  n b q o u o e  n  s   r b o j    b  q q t q s   f n i f    u l
```

- All the Names of God
- Alphabet Expanding
- ASCII Hegemony
- Letterformed Terrain

You can download the linked Perl files and/or simply copy and paste the following four lines, which correspond to the four titles above:

All the Names of God:

```
perl -e '{print"a"x++$...$"x$.,$,=_;redo}'
```

Alphabet Expanding:

```
perl -e '{print$,=$"x($.+=.01),a..z;redo}'
```

ASCII Hegemony:

```
perl -e '{print"  ".chr for 32..126;redo}'
```

Letterformed Terrain:

```
perl -e '{print$",$_=(a..z)[rand$=];redo}'
```