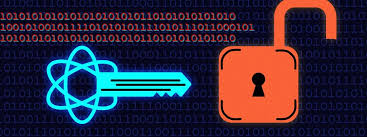
****

**Criptografia**

**TP1**

João Miguel da Silva Alves (83624)

Paulo Jorge Alves (84480)

**MESTRADO INTEGRADO EM ENGENHARIA BIOMÉDICA**

**INFORMÁTICA MÉDICA 2020/2021**

**Cifra de Substituição**

É o tipo de cifra na qual cada letra da mensagem é substituída por outra completamente aleatória. O código Python para fazer um ataque a esta cifra encontra-se nas figuras abaixo.

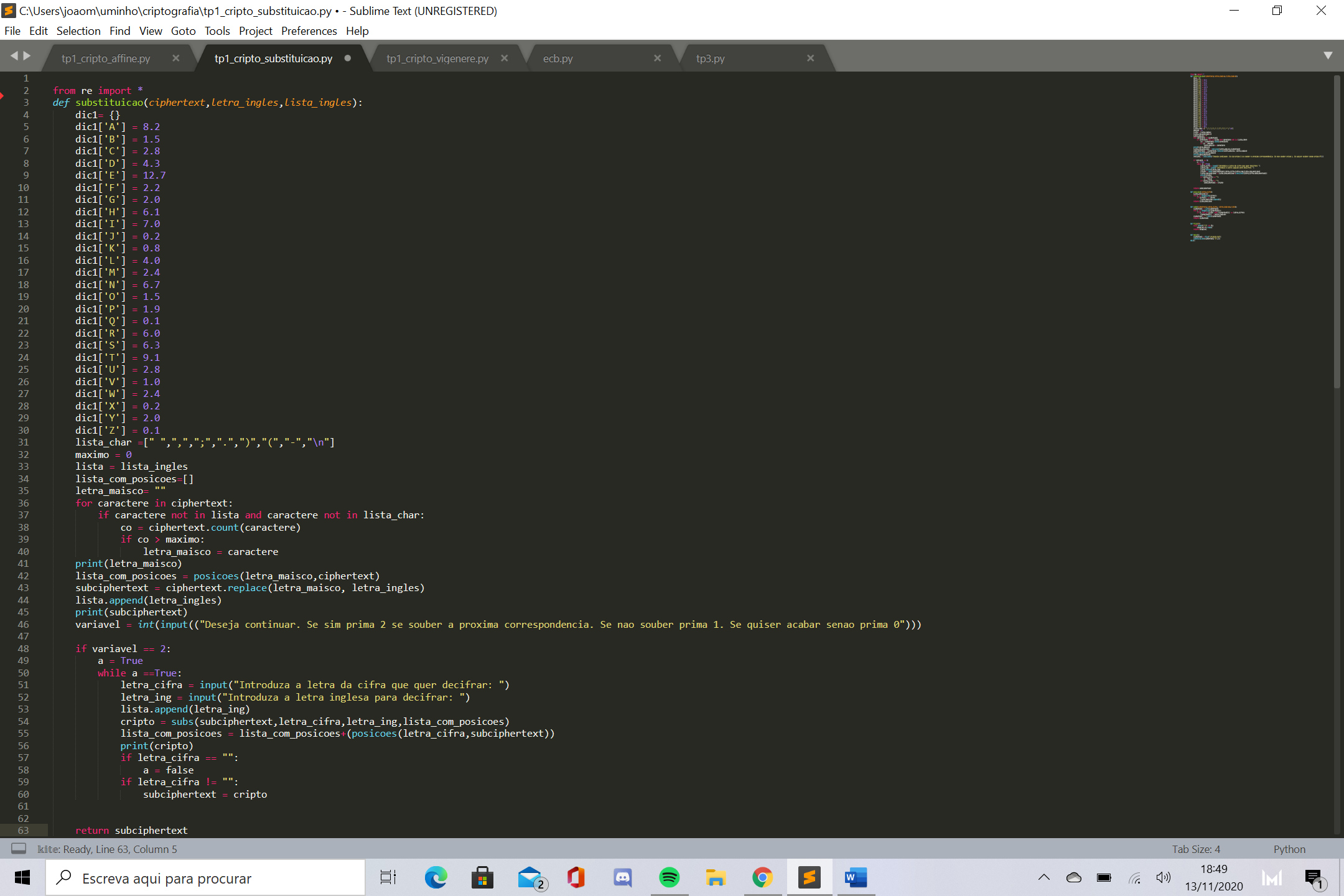


Figura 1 – Código para a cifra de substituição

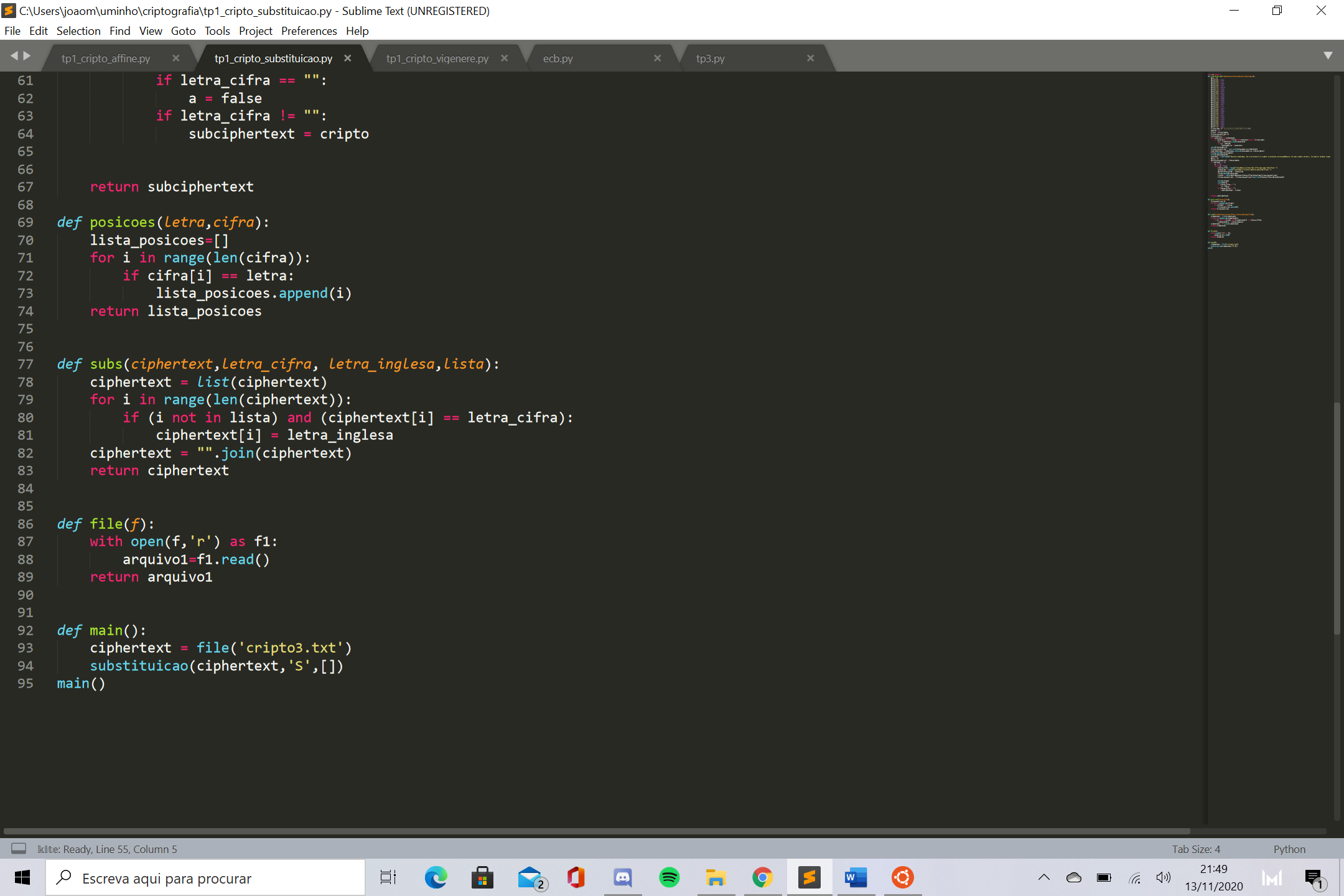


Figura 2 - Código para a cifra de substituição (continuação)

Na cifra de substituição, é feito um *shift* de uma letra na cifra para a correspondente letra no texto limpo. Por exemplo, se a letra V na cifra corresponder à letra S, então qualquer letra V na cifra corresponderá à letra S no texto limpo e, assim, sucessivamente.

Para decifrar esta cifra necessitamos de construir um dicionário, sendo as chaves as letras do alfabeto e os valores correspondem às suas frequências relativas na língua inglesa. De seguida, construímos uma lista com os caracteres da cifra que não iram ser cifrados. Depois fomos ao nosso criptograma, identificar a letra que mais aparecia. A partir daí, fomos testar se, de facto, a letra que mais aparecia no criptograma correspondia à letra “E”, que é a letra mais comum em textos de língua inglesa. Contudo, ao realizar essa correspondência e assumindo a partir daí outras correspondências, não obtínhamos palavras em inglês. Por esse motivo, tentamos que a letra com mais ocorrências correspondesse à letra “T”, mas usando o mesmo raciocínio anterior, não obtivemos resultados. De seguida, tentamos para a letra “A” e letra “I”, mas sem sucesso. No momento que fizemos a correspondência da letra que mais aparecia no criptograma com a letra “S”, algumas palavras, principalmente de duas letras começavam a fazer sentido no texto.

Depois, entramos num ciclo *while* em que só terminava quando o “*user*” não soubesse mais nenhuma correspondência, o que correspondia ao momento em que terminava e já tinha obtido o texto limpo. Em cada iteração fomos fazendo correspondências, uma vez que palavras começavam a formar-se. Em cada iteração é pedido ao utilizador que assuma uma correspondência de uma letra na cifra para uma letra do alfabeto inglês e essa substituição é feita com recurso à função subs(), que retorna um criptograma com as alterações feitas. A função subs() irá para cada caractere dado como input presente na cifra e que não esteja no conjunto dos caracteres especiais que não são cifrados, substituir pela letra do alfabeto inglês também dado como input. Com recurso, também, à função “posições”, é possível obter uma lista das posições em que foram feitas a substituição, de forma a não ocorrer mais nenhuma substituição nas próximas iterações. No fim de encontrar correspondência para todas as letras na cifra, o utilizador não dá nenhum caractere como input e o ciclo *while* acaba e é retornado o texto limpo. É também dado como output na tela um dicionário que contém como chaves as várias letras na cifra e como valores as letras do alfabeto inglês que cada uma corresponde.

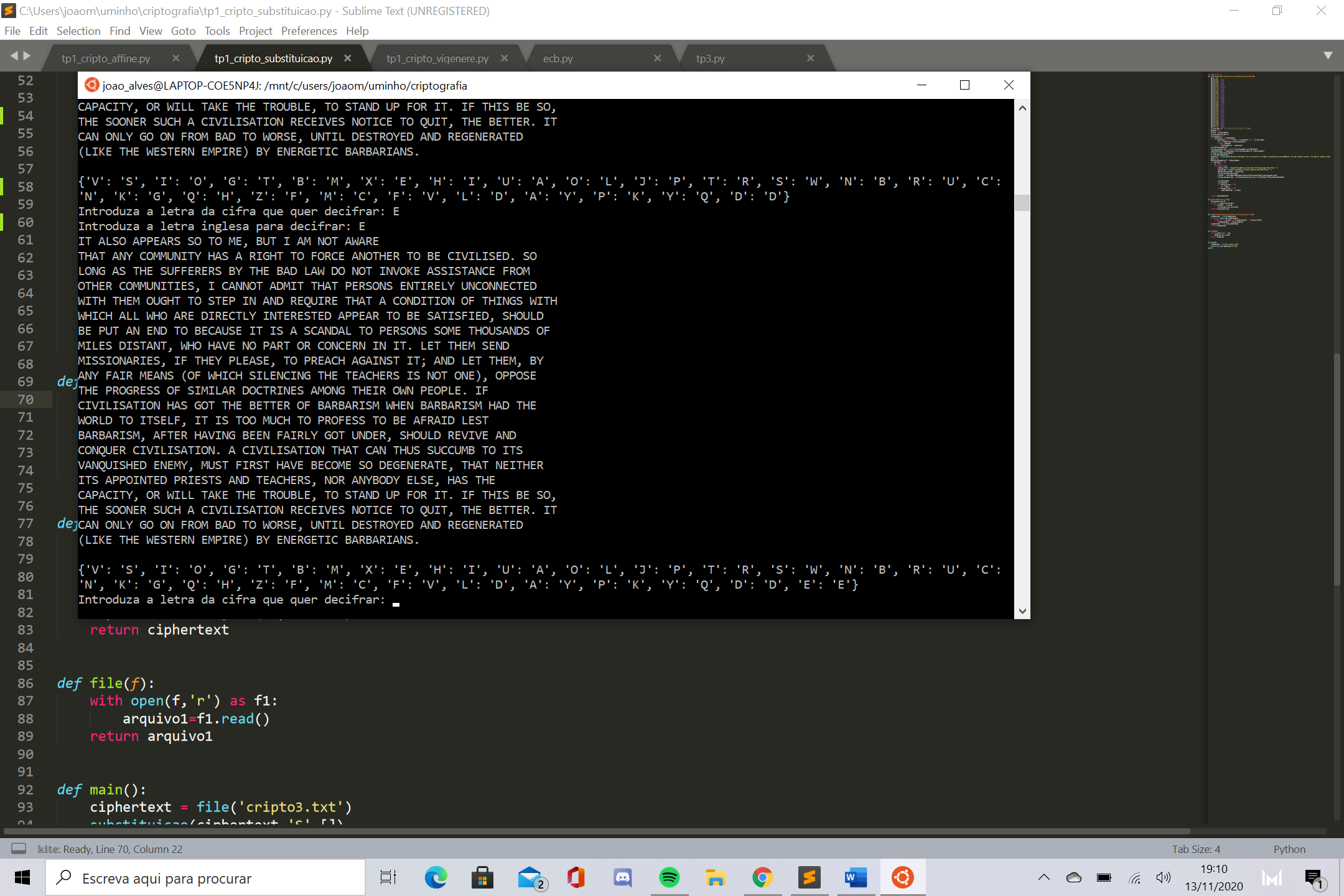


Figura 3 -Texto limpo obtido para a cifra de substituição, e dicionário com todas as correspondências.

**Cifra de Affine**

A cifra de afinne é um tipo de cifra de substituição, em que cada letra de um alfabeto é mapeada para seu equivalente numérico, criptografada usando uma função matemática simples (y=ax+b)mod26 e convertida de volta para uma letra.

O código Python para o ataque a esta cifra está representado nas fotos seguintes.

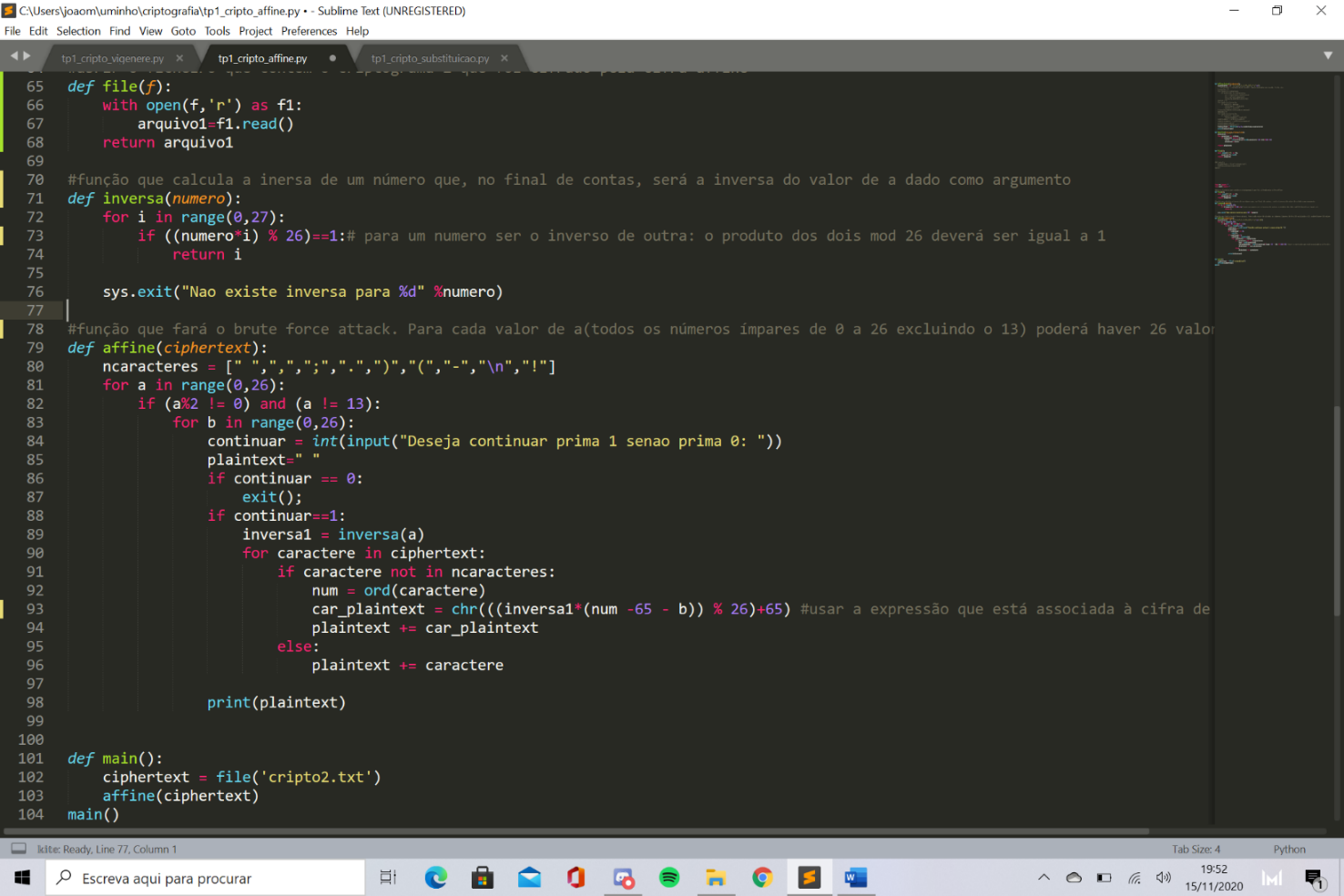


Figura 4 – Código para a cifra de affine

Para a cifra de affine, realizamos um “*brute force attack*”. Na expressão matemática usada na cifra de affine (y=(x-b) mod26) existem 12 valores possíveis para a variável “a”, uma vez que existem 12 números possíveis que sejam co-primos com o número 26 e existem 26 números possíveis para a variável “b”, desde o número 0 ao 25. Portanto, há 312 (12\*26) chaves possíveis para decifrar um criptograma que foi cifrado pela cifra de affine.

Para efetuar um ataque deste tipo, implementamos a função “affine” que recebe como argumento o criptograma cifrado pela cifra de affine. De seguida, através de dois ciclos “for”, iteramos para cada valor de “a”, de forma a determinar a usa inversa. Por fim, com o “b” e a inversa de “a” obtivemos o texto limpo através da expressão escrita acima.

O “*brute force attack*” realizado, numa primeira perspetiva, poderá parecer que demorará imenso tempo, no entanto, assumindo que uma pessoa carrega no *enter* de dois em dois segundos, isso significa que durante um minuto terá carregado 30 vezes e, por isso, terá obtido 30 textos limpos. Em pouco mais de dez minutos, qualquer pessoa é capaz de ter carregado as 312 vezes no *enter* e obtido os 312 textos limpos possíveis. Após 138 iterações, a qual corresponde a um valor 11 para a variável “a” e um valor “8” para a variável b, verificou-se que o texto mostrado era o pretendido (figura 5).

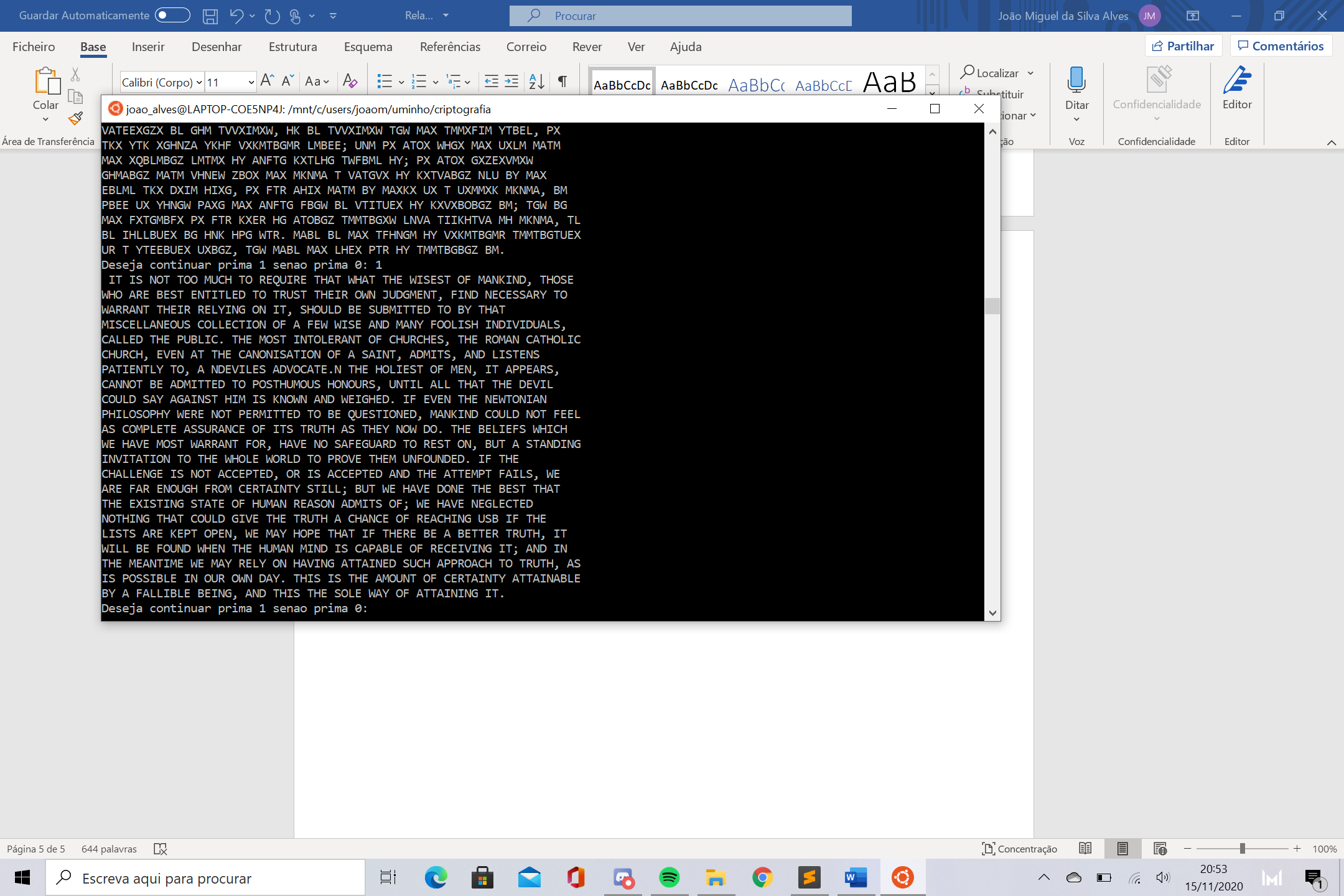


Figura 5 – Texto limpo para a cifra de affine

**Cifra de Vigenère**

A cifra de Vigenère é um método de criptografia que utiliza uma série de diferentes "cifras de César" com base nas letras de uma palavra-chave.

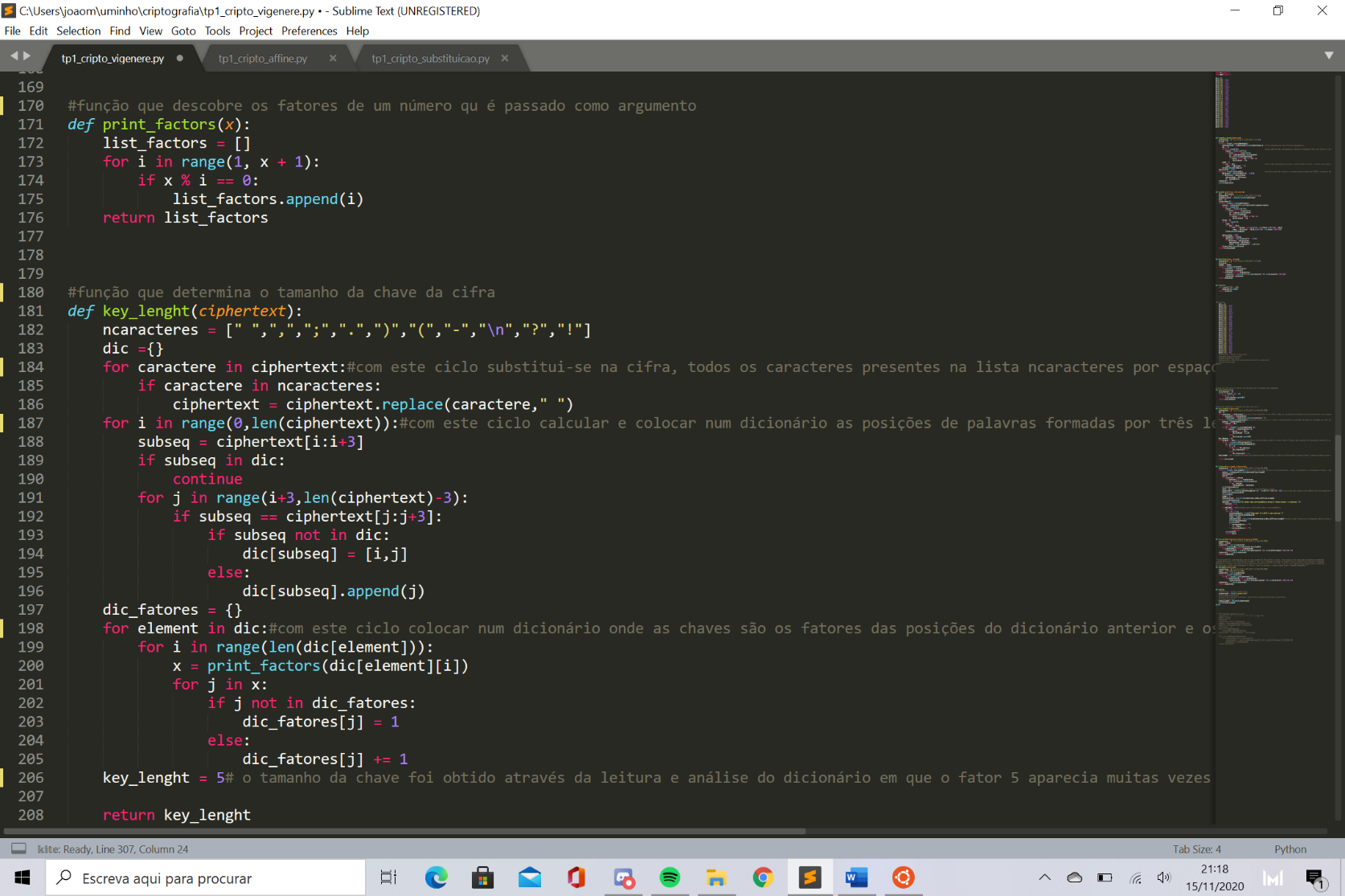
****O código Python para o ataque a esta cifra está representado nas fotos abaixo.

Figura 6 – Código para a cifra de vigenère

Para o criptograma cifrado com a cifra de vigenère, implementamos a função designada por “key\_lenght”. Primeiramente, substitui no criptograma todos os caracteres que se encontrão na lista “ncaracteres” por espaços, uma vez que no ciclo “for” a seguir tentar-se-á encontrar palavras formadas por três letras que se repetem ao longo do criptograma e essas mesmas palavras não podem conter letras dessa lista. De seguida, é colocado num dicionário onde as chaves são as palavras de três letras e os valores correspondem a uma lista que contém todas posições onde essa palavra se encontra no criptograma.

De seguida é implementado um dicionário onde as chaves serão cada fator das posições encontradas antes e o valor será o número de vez que esse fator ocorre. Para calcular os fatores recorre-se à função “*print\_factors*”. Após uma análise ao dicionário obtido, verificou-se que o tamanho da chave seria 5, visto que o fator 5 aparecia várias vezes sendo um possível valor para o tamanho da chave. Mais tarde, após tentativas e erro e por análise do output das funções seguintes que retornavam um texto limpo, verificou-se que o tamanho da chave teria de ser 5, pois com esse tamanho conseguíamos encontrar várias palavras em inglês a serem formadas.

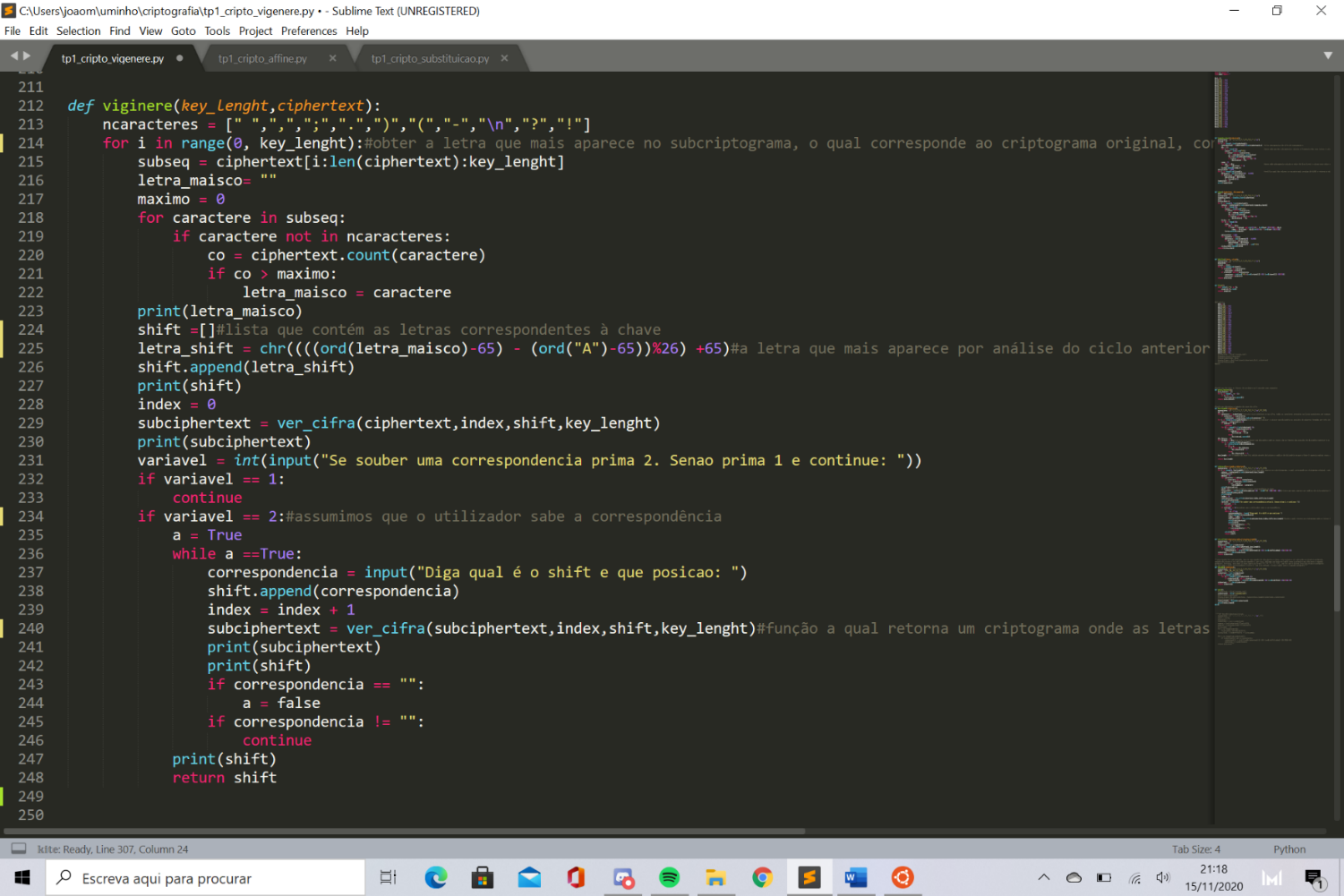


Figura 7 – Código para a cifra de vigenère

Com a implementação da função “viginère”, obtínhamos um subcriptograma que correspondia ao criptograma original, contudo com as letras nas posições 5 em 5.

De seguida, fomos descobrir qual a letra que mais ocorria nesse subcriptograma e fizemo-la corresponder à letra “A” do alfabeto inglês. Esta correspondência resulta no facto, de suspeitarmos que a primeira palavra formada por “WMP” do criptograma original correspondia à palavra “THE”. Como para obter a letra T a partir da letra W, é necessário que a primeira letra da chave seja um “D”. Como a letra que mais aparecia no subcriptograma era um “D”, logo para que a primeira letra da chave fosse um “D”, teríamos de associá-la à letra “A” no texto limpo.

A seguir, é pedido ao utilizador que ao olhar para o criptograma agora alterado possa saber uma correspondência possível. Essa correspondência irá constituir a letra seguinte da chave da cifra e com ajuda da função “ver\_cifra”, todas as letras nessas posições irão sofre esse Shift, obtendo, assim, um novo criptograma com essas alterações. Após algumas tentativas, verificou-se que a chave da cifra era [D, F, L, V, Y].



Figura 8 – Código para a cifra de vigenère

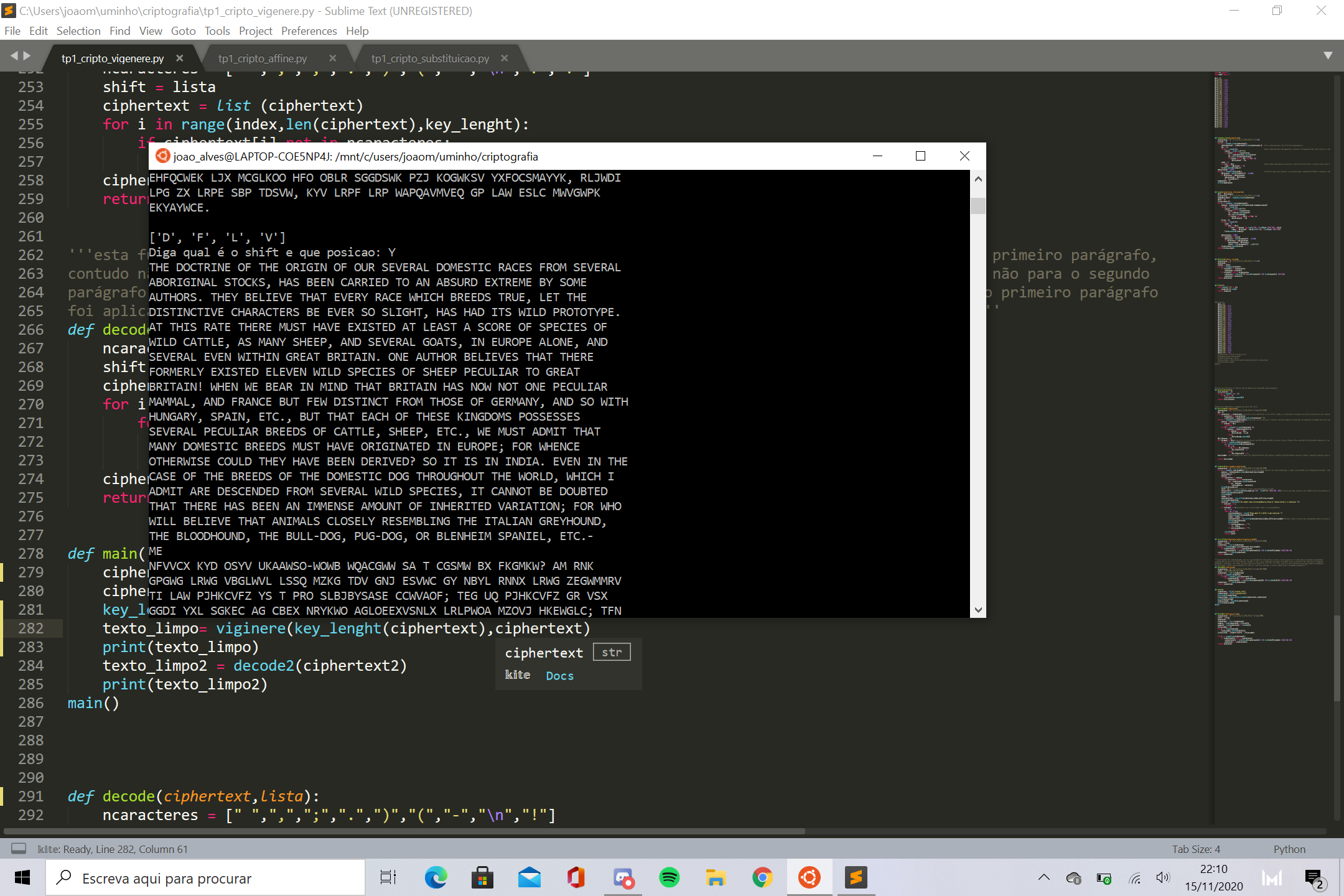
A função “decode2” foi implementada, uma vez que quando foi descoberta a chave, esta estava a ser aplicada ao primeiro parágrafo, contudo não estava a ser aplicada ao segundo e, por isso, obtíamos um texto em inglês para o primeiro, mas não para o segundo parágrafo. Provavelmente, não estava a reconhecer a passagem de linha através do caractere “\n”. Portanto, aplicamos a chave agora na forma ['Y','D','F','L','V'], uma vez que à última letra do primeiro parágrafo foi aplicado o *shift* correspondente à letra V. Com isto obtemos o texto inglês para o segundo parágrafo.

Figura 9 – Primeiro parágrafo do texto limpo

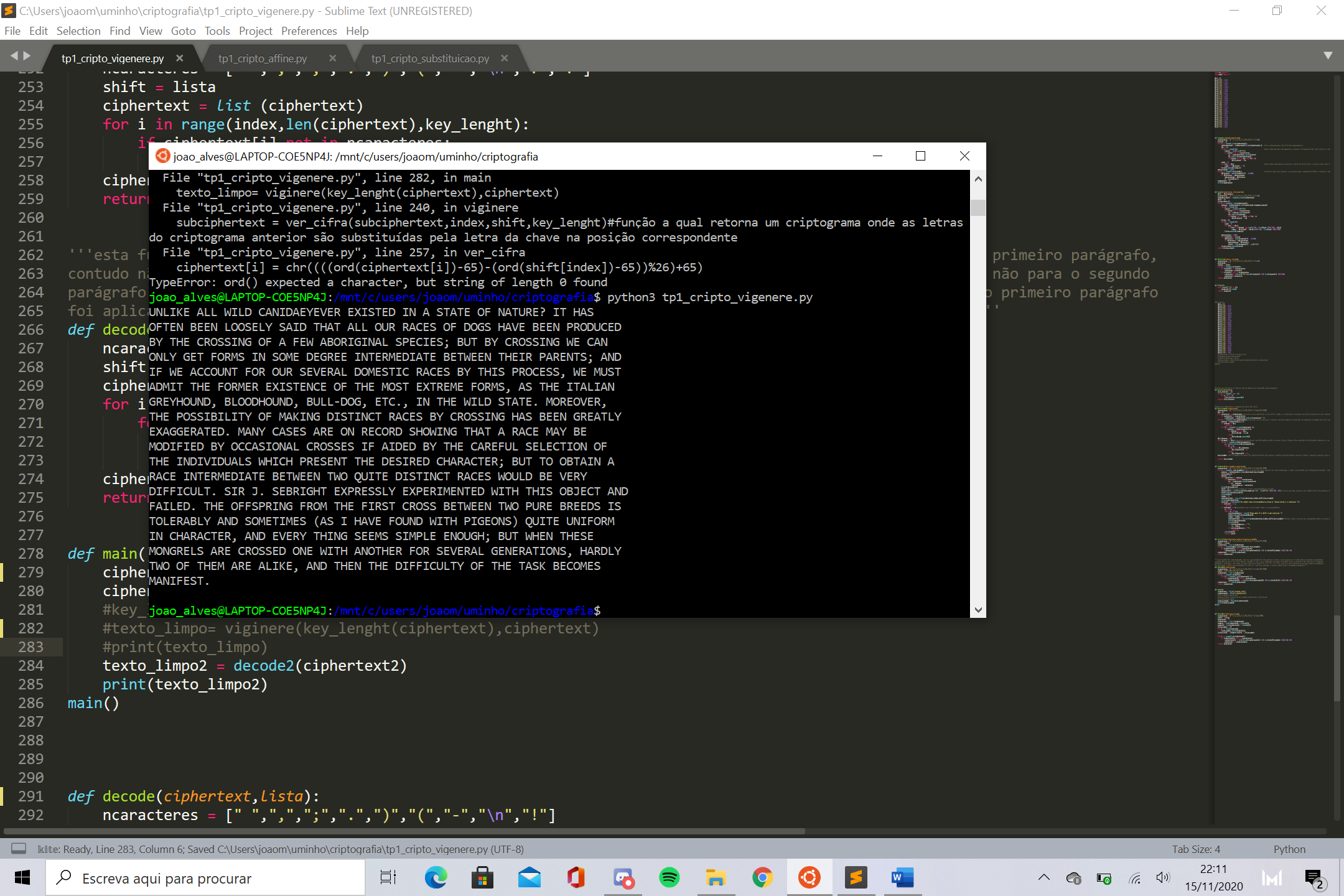


Figura 10 - Segundo parágrafo do texto limpo