Worked example 1 . Iterating over items.

This program uses for to iterate over a list of dice rolls and print the value of each item in the list.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3 | rolls = [1, 4, 3, 6]  for **dice** in rolls:   |  |  | | --- | --- | | print(**dice**) | . | |

Worked example 2 . Counting selected items

This program uses for to iterate over a list of dice rolls and **count** the number of items with a value greater than 3.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6 | rolls = [1, 4, 3, 6]  count = 0  for **dice** in rolls:   |  |  | | --- | --- | | if **dice** > 3:  count = count + 1 | . |   print(count) |

Worked example 3 . Collecting selected items into a list

This program uses for to iterate over a list of dice rolls and **collect** the items with a value greater than 3 into a new list named selection.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6 | rolls = [1, 4, 3, 6]  selection = []  for **dice** in rolls:   |  |  | | --- | --- | | if **dice** > 3:  selection.append(dice) | . |   print(selection) |

**English words**

Task 1 .

**Step 1**

**Open** this [program](https://ncce.io/py-words-1) (ncce.io/py-words-1) in your development environment.

|  |  |
| --- | --- |
| 1  2  3 | from data import dictionary  nb\_words = len(dictionary)  print(nb\_words, "english words in the list") |

Line 1 imports the dictionary, i.e. the list of words that the program will use. This is **not a standard Python component**. The list has been created specifically to allow you to perform these tasks.

**Step 2**

**Extend** the program so that it first prompts the user to enter a word length (number of characters), and then iterates over the dictionary, i.e. the list of words, and **counts** the number of words of this length.

**Tip**: Refer to Worked example 2 about counting the number of selected items in a list.

**Tip**: Use the len function to retrieve the length of each word in the dictionary.

|  |  |  |
| --- | --- | --- |
|  | len(string)  e.g. len("deoxyribonucleic")  e.g. len(name) | Returns the length (number of characters) of a string. |

|  |  |
| --- | --- |
| **Example** |  |
| Note: Use this example to check your program. This is the output your program should produce when searching for 12-letter words. | |
| The program displays a prompt and waits for keyboard input. | Length of words to search for: |
| The user types a reply. | 12 |
| The program displays the number of words of the given length. | There are 29126 words with 12 letters |

Task 2 .

**Open** this [program](https://ncce.io/py-words-2) (ncce.io/py-words-2) in your development environment.

|  |  |
| --- | --- |
| 1 | from data import dictionary |

**Extend** the program so that it first prompts the user for a string (a piece of text) to search for, and then iterates over the list of words in the dictionary and **collects** the ones that contain this piece of text into a new list.

In the end, the program should display the collected words, one word per line.

**Tip**: Refer to Worked example 3 about collecting selected items into a new list. Worked example 1 should help with displaying the contents of the new list.

**Tip**: Use the in operator to check if a word contains a piece of text.

|  |  |  |
| --- | --- | --- |
|  | string in string  e.g. "syn" in term  e.g. letter in "aeiou" | Evaluates to True if a string can be found within another, or to False otherwise. |

|  |  |
| --- | --- |
| **Example** |  |
| Note: Use this example to check your program. This is the output your program should produce when searching for the words that contain the text "python". | |
| The program displays a prompt and waits for keyboard input. | Text to search for: |
| The user types a reply. | python |
| The program displays the words that contain the particular substring. | python pythonissa  pythoness pythonist  pythonic pythonize  pythonical pythonoid  pythonid pythonomorph  pythonidae pythonomorpha  pythoniform pythonomorphic  pythoninae pythonomorphous  pythonine pythons  pythonism |

Explorer task . The longest word

**Open** the program below in your development environment (ncce.io/py-words-3) and **complete** the missing condition in line 4, so that the longest variable holds the longest word contained in the list.

**Tip**: Use the len function and compare the length of the current word to the length of the longest word encountered so far.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | from data import dictionary  longest = ""  for word in dictionary:  if :  longest = word  print(longest) |

Explorer task .

**Read** the Python program below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | words = ["pig", "hen", "ox", "cow", "duck"]  counts = [0, 0, 0, 0, 0]  for word in words:  length = len(word)  counts[length] = counts[length] + 1  print(counts) |

Verbally, this program could be described as:

For each word in the list of words:

* Compute the length of the current word
* Increase the counter that corresponds to that length by 1

When this program is executed, what do you expect its output to be?

|  |
| --- |
|  |

In your development environment, **open** and **run** [a version of this program](http://ncce.io/py-words-4) (ncce.io/py-words-4) that performs the same task on a list containing one thousand English words.