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.

|  |  |  |  |
| --- | --- | --- | --- |
| 123 | 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.

|  |  |  |  |
| --- | --- | --- | --- |
| 123456 | rolls = [1, 4, 3, 6]count = 0for **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.

|  |  |  |  |
| --- | --- | --- | --- |
| 123456 | 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 on Bourne to Code.

|  |  |
| --- | --- |
| 123 | from extra.data import dictionarynb\_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.

1. Display a **print** message to the user that says “Length of words to search for:”
2. Capture the users **input** as an **integer** and assign it to the **variable** length. Use one of the lines below:
	1. length = str(input())
	2. length = int(input())
	3. length = ord(input())
3. Create a **variable** named **count** and **assign** it the value of **0** (zero)
4. Create a **for** loop to go through each **word** in **dictionary.** Check the length of each word and if it is equal to the length you choose add one to the variable count.

Below is the code to add but the **order** is **mixed up** and needs **indenting**

count = count + 1

if len(word) == length:

for word in dictionary:

1. Display a **print** message to the user that says ("There are", **count**, "words with", **length**, "letters"). **Count** and **length** are the **variables** you created.

|  |  |
| --- | --- |
| **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 on Bourne to Code.

|  |  |
| --- | --- |
| 1 | from extra.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.

1. Display a **print** message to the user that says “Text to search for:”
2. Create a variable named **sub**, then capture the users **input** to it (You don’t need to use **int** this time to make it an integer)
3. Create an empty **list** named **collection**. One of the below has the correct brackets for a list.
	1. collection = [ ]
	2. collection = { }
	3. collection = ( )
4. You now need to create the **for** loop. This will go through each **word** in the **dictionary**. If the **sub** **variable** **value** is **in** the **word** then use the **append** command to add it to the end of the **collection** **list**.

**Rearrange** the **code** **below** to get this working:

collection.append(word)

if sub in word:

for word in dictionary:

1. Last two lines, **for** each **word** in the **collection** list, **print** out the word. You will need another **for** loop and then a line to **display** each word.

Create a **for** loop line and then a **print** line.

|  |  |
| --- | --- |
| **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 pythonissapythoness pythonistpythonic pythonizepythonical pythonoidpythonid pythonomorphpythonidae pythonomorphapythoniform pythonomorphicpythoninae pythonomorphouspythonine pythonspythonism |

 Task 3 . The longest word

**Open** the program below in your development environment ([ncce.io/py-words-3](https://replit.com/%40NCCE/prg5-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. If the **word** is **greater** **than** (>) the **longest** then longest becomes that word.

|  |  |
| --- | --- |
| 123456 | from extra.data import dictionarylongest = ""for word in dictionary: if : longest = wordprint(longest) |

 Explorer task .

**Read** the Python program below:

|  |  |
| --- | --- |
| 123456 | 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 the complete list containing thousands of English words.

This resource is licensed under the Open Government Licence, version 3. For more information on this licence, see [ncce.io/ogl](http://ncce.io/ogl).