1. Set up your environment (skip this if you've completed the first activity). There are several options for getting up and running:

Done!

2. Lists, tuples, sets and dictionaries. Use the starter notebook to help with the remaining problems. Four built in data structures in Python that we will make use of are lists, tuples, sets and dictionaries.

a. A list is exactly that: a list. Python lists are ordered and changeable. Write code to convert the following paragraph to a list of strings:

Answer: list_of_strings = in_string.split()

b. Create a new list that consists of every third word from the original list; print the new list, which should start with "of".

Answer: list_of_strings[2::3]

c. What is the 16th from last item in the original list?

Answer: "Uncovered" (list_of_strings[-16])

d. Convert the new list you created in part b to a tuple; print the tuple. When should you use a tuple instead of a list?

Answer: Use tuple() as a function. A tuple should be preferable when the items in the object is fixed (immutable) since tuples are immutable but faster than lists.

In [11]:
```
# Use triple quotes if you want to enter a string on more than one line

in_string = """this deluge of data calls for automated methods of data analysis
which is what machine learning provides in particular we define machine learning
as a set of methods that can automatically detect patterns in data and then use
the uncovered patterns to predict future data or to perform other kinds of decision
making under uncertainty"""

list_of_strings = in_string.split()       # https://www.w3schools.com/python/ref_string_split.asp
print('List of strings is:')
print(list_of_strings[2::3])              # fix me to answer the question

print('\n16th word from end is:')         # \n generates a newline
print(list_of_strings[-16])               # fix me to answer the question

tuple_of_strings = tuple(list_of_strings)
print('\nTuple of strings is:')
print(tuple(tuple_of_strings[2::3]))            # fix me to answer the question
```

```
List of strings is:
['of', 'for', 'of', 'which', 'machine', 'in', 'define', 'as', 'of', 'can', 'patterns', 'and', 'the', 'to', 'data', 'perform', 'of', 'under']

16th word from end is:
uncovered

Tuple of strings is:
('of', 'for', 'of', 'which', 'machine', 'in', 'define', 'as', 'of', 'can', 'patterns', 'and', 'the', 'to', 'data', 'perform', 'of', 'under')
```

e . Convert the list to a set and print the set. How many items are in the set? What is the difference between a list and a set?

Answer: There are 43 items in the set. A set is hashed (thus could only contain hashable items) and never contains repetitive items.

In [22]:
```
set_of_strings = set(list_of_strings)
print('Set of strings is:\n',set_of_strings,'\n',sep='')          # What happens if you remove ,sep='' ? Answer: Nothing (sep is defaulted to empty string)
print(f'Number of strings in the set = {len(set_of_strings)}\n') # https://www.geeksforgeeks.org/formatted-string-literals-f-strings-python/
print(f'Number of strings in the original list = {len(list_of_strings)}')
```

```
Set of strings is:
 {'deluge', 'particular', 'detect', 'machine', 'for', 'automated', 'analysis', 'what', 'learning', 'provides', 'in', 'or', 'decision', 'uncertainty', 'kinds', 'then', 'which', 'perform', 'patterns', 'of', 'to', 'under',
'calls', 'other', 'the', 'this', 'use', 'a', 'define', 'is', 'methods', 'can', 'and', 'automatically', 'we', 'uncovered', 'predict', 'as', 'set', 'making', 'future', 'data', 'that'}

Number of strings in the set = 43

Number of strings in the original list = 55
```

f. A Python dictionary consists of key, value pairs. Create a Python dictionary where each distinct word in the paragraph above is the key, and the corresponding value is the count of times that word appears.

Answer: the code below does that

g. Which word/words appears most frequently, and how many times does it appear?

Answer: "of" and "data" (4 times each)

In [19]:
```
d = {}  #create an empty dictionary
for i in list_of_strings:
    if i not in d:
        d[i]=0
    d[i]=d[i]+1                     # <====== complete this line

print(f'Dictionary is:\n{d}\n') # Print dictionary and extra newline

v = d.values() # extract word counts
print(f'Wordcounts = {v}\n') # print word counts
maxv = max(v) # find largest word count
print(f'max word count is {maxv}')
for i in d:
    if d[i]==maxv:
        print(f'"{i}" occurs {maxv} times')
```

```
Dictionary is:
{'this': 1, 'deluge': 1, 'of': 4, 'data': 4, 'calls': 1, 'for': 1, 'automated': 1, 'methods': 2, 'analysis': 1, 'which': 1, 'is': 1, 'what': 1, 'machine': 2, 'learning': 2, 'provides': 1, 'in': 2, 'particular': 1, 'we':
1, 'define': 1, 'as': 1, 'a': 1, 'set': 1, 'that': 1, 'can': 1, 'automatically': 1, 'detect': 1, 'patterns': 2, 'and': 1, 'then': 1, 'use': 1, 'the': 1, 'uncovered': 1, 'to': 2, 'predict': 1, 'future': 1, 'or': 1, 'perf
orm': 1, 'other': 1, 'kinds': 1, 'decision': 1, 'making': 1, 'under': 1, 'uncertainty': 1}

Wordcounts = dict_values([1, 1, 4, 4, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1])

max word count is 4
"of" occurs 4 times
"data" occurs 4 times
```

h. Use a list comprehension to convert the dictionary to a list of tuples, where the first element of the tuple contains the key, and the second element contains the value. Note that your code should be a single line.

Answer: the code below does that

In [20]:
```
list_of_tuples = [(k,v) for k,v in d.items()]  #this line uses a list comprehension
print(f'The list of tuples is:\n{list_of_tuples}')
```

```
The list of tuples is:
[('this', 1), ('deluge', 1), ('of', 4), ('data', 4), ('calls', 1), ('for', 1), ('automated', 1), ('methods', 2), ('analysis', 1), ('which', 1), ('is', 1), ('what', 1), ('machine', 2), ('learning', 2), ('provides', 1),
('in', 2), ('particular', 1), ('we', 1), ('define', 1), ('as', 1), ('a', 1), ('set', 1), ('that', 1), ('can', 1), ('automatically', 1), ('detect', 1), ('patterns', 2), ('and', 1), ('then', 1), ('use', 1), ('the', 1),
('uncovered', 1), ('to', 2), ('predict', 1), ('future', 1), ('or', 1), ('perform', 1), ('other', 1), ('kinds', 1), ('decision', 1), ('making', 1), ('under', 1), ('uncertainty', 1)]
```

3. NumPy is a numerical computing package for Python. You'll often see the library imported at the beginning of a script as import numpy as np.

a. Python lists can be converted to NumPy arrays. Manually create a list of lists that represents the matrix

Answer: A = [[1,1,3],[4,4,4],[5,6,9]]

b. Convert the list of lists to a numpy array using np.array().

Answer: A = np.array(A)

c. What is A^{−1}?

Answer: A_inv = np.linalg.inv(A)

d. Solve Ax = b, where b = [1, 2, 3].T

Answer: x = np.linalg.solve(A,b) or A_inv@b (the former is more efficient)

e. Create a length n array random numbers distributed uniformly between 0 and 1

Answer: rng = np.random.default_rng(); y = rng.random(n)

f. Repeat for n = 10^1, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8 For each value of n, record the minimum value of the array. Create a plot with n on the horizontal axis and the logarithm of the minimum value on the vertical axis. Are the results what you expect? Why?

Answer: the code below does that. The results are as expected since the expectation of the minimum of several uniform random variables diminishes the more random variable there are.

In [30]:
```
import numpy as np
import matplotlib.pyplot as plt

A = [[1,1,3],[4,4,4],[5,6,9]]
A = np.array(A)
A_inv = np.linalg.inv(A)         # complete this line.


print(f'The inverse of A is\n{A_inv}\n')



b = np.array([[1,2,3]]).T        # What happens if you don't use double square brackets? Answer: It
print(f'b =\n{b}\n')
x = A_inv@b                      # @ is the matrix multiplication operator in Numpy, NOT * !!!
print(f'x = A_inv*b =\n{x}\n')

x = np.linalg.solve(A,b)
print(f'x = np.linalg.solve(A,b) =\n{x}')



### Write code below to answer question
# https://numpy.org/doc/stable/reference/random/index.html
rng = np.random.default_rng()

logmin_y = np.log(np.array([np.min(rng.random(10**n)) for n in range(1,9)]))
n = np.array([10**n for n in range(1,9)])

plt.plot(n, logmin_y)

# https://danieltakeshi.github.io/2016/09/25/the-expectation-of-the-minimum-of-iid-uniform-random-variables/
```

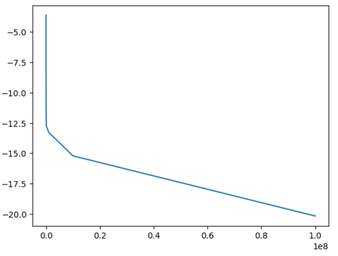
```
The inverse of A is
[[ 1.5    1.125 -1.   ]
 [-2.    -0.75   1.   ]
 [ 0.5   -0.125  0.   ]]

b =
[[1]
 [2]
 [3]]

x = A_inv*b =
[[ 0.75]
 [-0.5 ]
 [ 0.25]]

x = np.linalg.solve(A,b) =
[[ 0.75]
 [-0.5 ]
 [ 0.25]]
```

Out[30]:
```
[<matplotlib.lines.Line2D at 0x7fbcc8bd4c10>]
```



In [ ]: