```python
# classwork 3/29/2020 + additional (optional) information at the bottom

import os

print(os.getcwd()) # current working directory

try:
    os.mkdir('C:\\python')
except:
    print("Folder already exists")

os.chdir('C:\\python')

print(os.getcwd())

# establish a connection, create IO (input/output) stream
h = open('hamlet.txt', 'r', encoding = 'utf8')

print(h)
print(type(h))

htext = h.read() # read the whole file into memory
#print(htext)
print(type(htext))

print(f"Number of characters: {len(htext)}")
print(f"Number of words: {len(htext.split())}") # default - split by space
print(f"Number of sentences: {len(htext.split('.'))}")

print(f"Average number of words in a sentence:")
print(f"{len(htext.split())/len(htext.split('.'))}")

h.close()
# you have to close files that you open

# BETTER Alternative:
try:
    # assumes that hamlet.txt is in current working directory
    #with open('hamlet.txt', 'r', encoding = 'utf8') as h:
    # if working directory is different
    with open('C:\\python\\hamlet.txt', 'r', encoding = 'utf8') as h:
        htext = h.read(400) # read first 400 characters
except:
    print("No such file")
    htext = "none"

print("_" * 80)
print(htext)
print(f"Number of characters: {len(htext)}")
print(f"Number of words: {len(htext.split())}") # default - split by space
print(f"Number of sentences: {len(htext.split('.'))}")
```

```
# we do not need to close the IO stream in this case, no need for close()
# ------------------------------------------------------------------------------------------


# ------- ADDITIONAL (OPTIONAL) INFORMATION --------------------------------------------------

# encoding intro: https://www.w3.org/International/questions/qa-what-is-encoding
# encoding in more depth: https://docs.python.org/3/howto/unicode.html
# different encodings:
# https://docs.python.org/3/library/codecs.html#standard-encodings

# ------------------------------------------------------------------------------------------

# IO classes and subclasses:
# https://docs.python.org/3/library/io.html

# ------------------------------------------------------------------------------------------

# FROM: https://docs.python.org/3/tutorial/inputoutput.html

#It is good practice to use the with keyword when dealing with file objects.
#The advantage is that the file is properly closed after its suite finishes,
#even if an exception is raised at some point. Using with is also much shorter
#than writing equivalent try-finally blocks:
#
#>>>
#>>> with open('workfile') as f:
#...     read_data = f.read()
#
#>>> # We can check that the file has been automatically closed.
#>>> f.closed
#True
#If you’re not using the with keyword, then you should call f.close() to close
#the file and immediately free up any system resources used by it. If you don’t
#explicitly close a file, Python’s garbage collector will eventually destroy the
#object and close the open file for you, but the file may stay open for a while.
#Another risk is that different Python implementations will do this clean-up at
#different times.
#
#After a file object is closed, either by a with statement or by calling f.close(),
#attempts to use the file object will automatically fail.

# ------------------------------------------------------------------------------------------
```