10. Files and Exceptions

Reading from a File

In [4]:

```
with open('pi_digits.txt') as file_object:
        contents = file_object.read()
        print(contents.rstrip())
```

```
3.
1415926535
8979323846
2643383279
```

- **Reading an Entire File**
- the first line has an *open()* function
- to do something with a file first you have to open it with this function
- the open function needs one argument
  - the name of the file you want to open
- we get a return of this object
- we store this object in *file_object*
- the keyword *with* closes the file once acces to it is no longer needed
- you could simply open and close the file with *open()* and *close()* but this would give some errors (the file may never close)
- once we have a file object representing the text file we can use *read()*
  - this method reads the entire txt file and stores that in contents
  - when we print *contents* we get the txt file

In [5]:

```
with open('files/pi_digits.txt') as file_object:
    contents = file_object.read()
    print(contents.rstrip())
```

```
3.
1415926535
8979323846
2643383279
```

- **File paths**
- you can also reach the file from another place
- also absolute paths also possible

In [7]:

```
filename = 'pi_digits.txt'

with open(filename) as file_object:
    for line in file_object:
        print(line.rstrip())
```

```
3.
1415926535
8979323846
2643383279
```

- **Reading line by line**
- first we store the name of the file we are reading from in a variable
- **this is a commin convention when working with files**
- this is just represents where the file is
- after calling open we store that object in file_object
- we print now each line with the for loop

In [8]:

```
filename = 'pi_digits.txt'

with open(filename) as file_object:
    lines = file_object.readlines()

for line in lines:
    print(line.rstrip())
```

```
3.
1415926535
8979323846
2643383279
```

- **Making a List of Lines from a File**
- the *readlines()* method takes ech line from the file and stores it in alist
- we can work with that data later
- for now we print each line of our new list lines

In [15]:

```
filename = 'pi_digits.txt'

with open(filename) as file_object:
    lines = file_object.readlines()

pi_string = ''
for line in lines:
    pi_string += line.strip()

print(pi_string)
print(len(pi_string))
```

```
3.141592653589793238462643383279
32
```

- **Working with t a File's Contents**
- we create a variable pi_string to hold the pi_string
- then we create a loop that adds each line of digits to pi_string and removes the newline character from each line
- we print the string and also show how long the string is

In [92]:

```
filename = 'files/million_digits.txt'

with open(filename) as file_object:
    lines = file_object.readlines()

pi_string = ''
for line in lines:
    pi_string += line.rstrip()

birthday = input("Enter your birthday, in the form mmddyy: ")
if birthday in pi_string:
    print("Your birthday appears in the first million digits of pi!")
else:
    print("Your birthday does not appear in the first million digits of pi.")
```

```
Enter your birthday, in the form mmddyy: 041089
Your birthday appears in the first million digits of pi!
```

Tasks

- 10-1. Learning Python: Open a blank file in your text editor and write a few lines summarizing what you've learned about Python so far. Start each line with the phrase In Python you can.... Save the file as learning_python.txt in the same directory as your exercises from this chapter. Write a program that reads the file and prints what you wrote three times. Print the contents once by reading in the entire file, once by looping over the file object, and once by storing the lines in a list and then working with them outside the with block.
- 10-2. Learning C: You can use the replace() method to replace any word in a string with a different word. Here's a quick example showing how to replace 'dog' with 'cat' in a sentence: >>> message = "I really like dogs." >>> message.replace('dog', 'cat') 'I really like cats.' Read in each line from the file you just created, learning_python.txt, and replace the word Python with the name of another language, such as C. Print each modified line to the screen.

In [95]:

```
# 10-1
filename = 'learning_python.txt'

print("--- Reading in the entire file:")
with open(filename) as f:
    contents = f.read()
print(contents)

print("\n--- Looping over the lines:")
with open(filename) as f:
    for line in f:
        print(line.rstrip())

print("\n--- Storing the lines in a list:")
with open(filename) as f:
    lines = f.readlines()

for line in lines:
    print(line.rstrip())
```

```
--- Reading in the entire file:
In Python you can write files.
In Python you can load files.
In Python you can make tests.
In Python you can make random numbers.

--- Looping over the lines:
In Python you can write files.
In Python you can load files.
In Python you can make tests.
In Python you can make random numbers.

--- Storing the lines in a list:
In Python you can write files.
In Python you can load files.
In Python you can make tests.
In Python you can make random numbers.
```

In [102]:

```
# 10-2
# the replace.() method replaces any word in a string.
# an example:
# message = "I really like dogs."
# message.replace('dog', 'cat')
# I really like cats.

filename = 'learning_python.txt'

with open(filename) as f:
    lines = f.readlines()

for line in lines:
    # Get rid of newline, then replace Python with C.
    line = line.rstrip()
    print(line.replace('Python', 'C'))
```

```
In C you can write files.
In C you can load files.
In C you can make tests.
In C you can make random numbers.
```

Writing to a File

In [45]:

```
filename = 'programming.txt'

with open(filename, 'w') as file_object:
    file_object.write("I love programming.")
```

- **Writing to an empty file**
- we use the *open()* method with a second argument telling, that you want to write to a file
  - the first argument is the name of the file
  - the second argument 'w' tells, that we are in the *write mode*
  - you can open a file:
    - with the read mode '*r*'
    - with the write mode '*w*'
    - and with the append mode '*a*'
    - or with the read plus mode '*r+*' which allows you to read and write the file
- we use the write() method on the file object to write a string to the file
- you can just write strings to a txt file, if you want to write numerical data use the str() function

In [46]:

```
filename = 'programming.txt'

with open(filename, 'a') as file_object:
    file_object.write("I also love finding meaning in large datasets.\n")
    file_object.write("I love creating apps that can run in a browser.\n")
```

In [104]:

```
filename = 'programming.txt'

with open(filename, 'a') as file_object:
    file_object.write("I also love finding meaning in large datasets.\n")
    file_object.write("I love creating apps that can run in a browser.\n")
```

Tasks

- 10-3. Guest: Write a program that prompts the user for their name. When they respond, write their name to a file called guest.txt.
- 10-4. Guest Book: Write a while loop that prompts users for their name. When they enter their name, print a greeting to the screen and add a line recording their visit in a file called guest_book.txt . Make sure each entry appears on a new line in the file.
- 10-5. Programming Poll: Write a while loop that asks people why they like programming. Each time someone enters a reason, add their reason to a file that stores all the responses.

In [115]:

```
# 10-3
filename = 'username.txt'
username = input("Enter your username: ")

with open(filename, 'a') as f_obj:
    f_obj.write("The username is: " + username + "\n")
```

```
Enter your username: Ugur
```

In [125]:

```
# 10-4

filename = 'names.txt'

while True:
    usernames = input("Enter your username: ")
    if usernames == 'q':
        break
    else:
        with open(filename, 'a') as f_obj:
            f_obj.write(usernames + "\n")
        print("Hi, " + usernames + "!")
```

```
Enter your username: Ugur
Hi, Ugur!
Enter your username: Tigu
Hi, Tigu!
Enter your username: q
```

In [160]:

```
# 10-5

filename = "poll.txt"
reasons = []

while True:
    reason = input("Why do you like programming? ")
    reasons.append(reason)
    continue_poll = input("Do you like to answer more questions? (y/n)")
    if continue_poll != 'y':
        break

with open(filename, 'a') as f_obj:
    for reason in reasons:
        f_obj.write(reason + "\n")

with open(filename, 'r') as f_obj:
    print("Your list of reasons contains this answers: ")
    for reason in reasons:
        print("\t" + str(reason))
```

```
Why do you like programming? Job
Do you like to answer more questions? (y/n)y
Why do you like programming? hobby
Do you like to answer more questions? (y/n)n
Your list of reasons contains this answers:
        Job
        hobby
```

Exceptions

In [49]:

```
print(5/0)
```

```
---------------------------------------------------------------------
-------
ZeroDivisionError                         Traceback (most recent cal
l last)
<ipython-input-49-fad870a50e27> in <module>
----> 1 print(5/0)

ZeroDivisionError: division by zero
```

- **Handling the ZeroDevisionError Exception**

In [50]:

```
try:
    print(5/0)
except ZeroDivisionError:
    print("You can't divide by zero!")
```

```
You can't divide by zero!
```

- we put the line of code which is reponsible for the error inside the try block
- if the code in the block runs, python skips over the except block
- if the code in the try block causes an error python looks for an except block whose error matches the one that was raised and runs the code in that block
- we get an friendly message instead of an error which stops the program

In [52]:

```
print("Give me two numbers, and I'll divide them.")
print("Enter 'q' to quit.")

while True:
    first_number = input("\nFirst number: ")
    if first_number == 'q':
        break
    second_number = input("Second number: ")
    if second_number == 'q':
        break
    answer = int(first_number) / int(second_number)
    print(answer)
```

```
Give me two numbers, and I'll divide them.
Enter 'q' to quit.

First number: 3
Second number: 0

---------------------------------------------------------------------------
-------
ZeroDivisionError                         Traceback (most recent cal
l last)
<ipython-input-52-c518059aa6de> in <module>
      9     if second_number == 'q':
     10         break
---> 11     answer = int(first_number) / int(second_number)
     12     print(answer)

ZeroDivisionError: division by zero
```

- **Using Exceptions to Prevent Craches**
- this program prompts the user to input a first and second number and divides tham
- unfortunately you can't divide by zero, so you'll get an error

In [54]:

```
print("Give me two numbers, and I'll divide them.")
print("Enter 'q' to quit.")

while True:
    first_number = input("\nFirst number: ")
    if first_number == 'q':
        break
    second_number = input("Second number: ")
    try:
        answer = int(first_number) / int(second_number)
    except ZeroDivisionError:
        print("You can't divide by 0!")
    else:
        print(answer)
```

```
Give me two numbers, and I'll divide them.
Enter 'q' to quit.

First number: 12
Second number: 12
1.0

First number: 0
Second number: 1
0.0

First number: 2
Second number: 0
You can't divide b 0!

First number: q
```

- **The else Block**
- wrapping the line that might produce errors in a try-except block
- any code that depends on the try block executing successfully goes in the else block
- we ask to try to complete the division operation in a try block
- this code, the only code might be responsible for the error is in that block
- we then have a ZeroDivisionError which we handle by printing our error message
- if the try statement does not succeed we have no division by zero so our code is running normal

In [55]:

```
filename = 'alice.txt'

with open(filename) as f_obj:
    contents = f_obj.read()
```

```
---------------------------------------------------------------------------
-------
FileNotFoundError                         Traceback (most recent cal
l last)
<ipython-input-55-afc69c46d931> in <module>
      1 filename = 'alice.txt'
      2
----> 3 with open(filename) as f_obj:
      4     contents = f_obj.read()

FileNotFoundError: [Errno 2] No such file or directory: 'alice.txt'
```

- **Handling the FIleNotFoundError Exception**
- we can't read from a missing file so we need an exception

In [57]:

```
filename = 'alice.txt'

try:
    with open(filename) as f_obj:
        contents = f_obj.read()
except FileNotFoundError:
    msg = "Sorry, the file " + filename + " does not exist."
    print(msg)
```

```
Sorry, the file alice.txt does not exist.
```

- we can read from the error message python gave us that we have a FileNotFoundError
- we can see that the *open()* function is responsible for that
- we use a try block and put that open function into that block
- we use a except block, wich we know that was originally a FileNotFoundError and print a message instead of the error

In [61]:

```
filename = 'alice.txt'

try:
    with open(filename) as f_obj:
        contents = f_obj.read()
except FileNotFoundError:
    msg = "Sorry, the file " + " does not exist."
    print(msg)
else:
    # Count the approximate number of words in the file.
    words = contents.split()
    num_words = len(words)
    print("The file " + filename + " has about " + str(num_words) + " words.")
```

```
The file alice.txt has about 17842 words.
```

- **Analyzing text**
- we use the *split()* method
  - title = "Alice in Wonderland"
  - title.split()
  - ['Alice', 'in', 'Wonderland']
  - split() will split the string into elemnts and store it in a list of words
- this time the FileNotFoundError will not occur
- we go into the else block which does:
  - using a split method and storing the list into 'words'
  - measure the lenght of the list with the len() method
  - printing the output

In [63]:

```
def count_words(filename):
    """Coun the approximate number of word in a file."""
    try:
        with open(filename) as f_obj:
            contents = f_obj.read()
    except FileNotFoundError:
        msg = "Sorry, the file " + " does not exist."
        print(msg)
    else:
        # Count the approximate number of words in the file.
        words = contents.split()
        num_words = len(words)
        print("The file " + filename + " has about " + str(num_words) + " word
s.")

filename = 'alice.txt'
count_words(filename)
```

```
The file alice.txt has about 17842 words.
```

- **Working with Multiple Files**
- first we put the bulk of our counting program into a function so we can call it later again

In [65]:

```
filenames = ['missing_book.txt', 'alice.txt', 'siddhartha.txt']
for filename in filenames:
    count_words(filename)
```

```
Sorry, the file  does not exist.
The file alice.txt has about 17842 words.
The file siddhartha.txt has about 46296 words.
```

- We included two new txt files in our directory
- we have a list of files, the last one is intentionally not in the same directory
- we make a for loop which uses our function for each file in the list
- we can see the exception error, when the file is missing
- the advantage of the exception is:
  - we don't get an error which stops the program from running
  - means the other two files would not be scanned, if the error happens
  - and the user is getting informed about an error
  - no one can see the report (safety reasons)

In [66]:

```
def count_words(filename):
    """Coun the approximate number of word in a file."""
    try:
        with open(filename) as f_obj:
            contents = f_obj.read()
    except FileNotFoundError:
        pass
    else:
        # Count the approximate number of words in the file.
        words = contents.split()
        num_words = len(words)
        print("The file " + filename + " has about " + str(num_words) + " word
s.")

filenames = ['missing_book.txt', 'alice.txt', 'siddhartha.txt']
for filename in filenames:
    count_words(filename)
```

```
The file alice.txt has about 17842 words.
The file siddhartha.txt has about 46296 words.
```

- **Failing Silently**
- we use a pass statement instead of a print, that some error is happening
- the user will not see any error
- the pass statement acts like a placeholder
- it is a reminder that you're choosing to do nothing a a specific place
- for example we want to write any missing file in a *missing_files.txt* the user has to see nothing

Tasks

- 10-6. Addition: One common problem when prompting for numerical input occurs when people provide text instead of numbers. When you try to convert the input to an int, you'll get a TypeError. Write a program that prompts for two numbers. Add them together and print the result. Catch the TypeError if either input value is not a number, and print a friendly error message. Test your program by entering two numbers and then by entering some text instead of a number.
- 10-7. Addition Calculator: Wrap your code from Exercise 10-6 in a while loop so the user can continue entering numbers even if they make a mistake and enter text instead of a number.
- 10-8. Cats and Dogs: Make two files, cats.txt and dogs.txt. Store at least three names of cats in the first file and three names of dogs in the second file. Write a program that tries to read these files and print the contents of the file to the screen. Wrap your code in a try-except block to catch the FileNotFound error, and print a friendly message if a file is missing. Move one of the files to a different location on your system, and make sure the code in the except block executes properly.
- 10-9. Silent Cats and Dogs: Modify your except block in Exercise * 10-8 to fail silently if either file is missing.
- 10-10. Common Words: Visit Project Gutenberg (http://gutenberg.org/ (http://gutenberg.org/)) and find a few texts you'd like to analyze. Download the text files for these works, or copy the raw text from your browser into a text file on your computer. You can use the count() method to find out how many times a word or phrase appears in a string. For example, the following code counts the number of times 'row' appears in a string:
  - line = "Row, row, row your boat"
  - line.count('row')
  - 2
  - line.lower().count('row')
  - 3
  - Notice that converting the string to lowercase using lower() catches all appearances of the word you're looking for, regardless of how it's formatted.
  - Write a program that reads the files you found at Project Gutenberg and determines how many times the word 'the' appears in each text.

In [171]:

```
# 10-6 + 10-7

print("Give me two numbers, and I'll add them together.")
print("Enter 'q' to quit.")

while True:
    first_number = input("\nFirst number: ")
    if first_number == 'q':
        break
    second_number = input("Second number: ")
    try:
        answer = int(first_number) + int(second_number)
    except ValueError:
        print("You have to give some number!")
    else:
        print(answer)
```

```
Give me two numbers, and I'll add them together.
Enter 'q' to quit.

First number: 12
Second number: 12
24

First number: 12
Second number: we
You have to give some number!

First number: 12
Second number: 34
46

First number: q
```

In [200]:

```
# 10-8

filenames = ['cats.txt', 'dogs.txt']
cat_names = ['einstein', 'diva', 'garfield']
dog_names = ['wullf', 'kurdi', 'köpek']

# Please select the filename!
filename = 'cats.txt'
# Please select the filename!

# writing the cat_names into the cats.txt
with open(filenames[0], 'w') as f_obj:
    for cat_name in cat_names:
        f_obj.write(cat_name + "\n")

# writing the dog_names into the dogs.txt
with open(filenames[1], 'w') as f_obj:
    for dog_name in dog_names:
        f_obj.write(dog_name + "\n")

# a function that reads the files of a certain file.
# if the file is missing the user gets an alert.
def read_files(filenames):
    """Read the names in the files and prints the content."""
    try:
        with open(filenames) as f_obj:
            contents = f_obj.read()
    except FileNotFoundError:
        msg = "Sorry, the file " + " does not exist."
        print(msg)
    else:
        # Print the content of the file.
        words = contents.split()
        # checking what kind of file is used, editing the message!
        if filename == 'cats.txt':
            for word in words:
                print("The name of the cat is: " + " \n- " + word + "!")
        elif filename == 'dogs.txt':
            for word in words:
                print("The name of the dog is: " + " \n- " + word + "!")

# call the function to read the content of the file!
read_files(filename)
```

```
The name of the cat is:
- einstein!
The name of the cat is:
- diva!
The name of the cat is:
- garfield!
```

In [227]:

```
# 10-9

filenames = ['cats.txt', 'dogs.txt']
dog_names = ['wullf', 'kurdi', 'köpek']

# writing the dog_names into the dogs.txt
with open(filenames[1], 'w') as f_obj:
    for dog_name in dog_names:
        f_obj.write(dog_name + "\n")

# the cats.txt is missing this time!

# a function that reads the files of a certain file.
# if the file is missing the user gets an alert.
def read_files(filenames):
    """Read the names in the files and prints the content."""
    try:
        with open(filenames) as f_obj:
            contents = f_obj.read()
    except FileNotFoundError:
        pass
    # cats.txt is missing, we just go on with the dogs.txt instead of giving an
 alert!!!
    else:
        # Print the content of the file.
        words = contents.split()
        # checking what kind of file is used, editing the message!
        if filename == 'cats.txt':
            for word in words:
                print("The name of the cat is: " + " \n- " + word + "!")
        elif filename == 'dogs.txt':
            for word in words:
                print("The name of the dog is: " + " \n- " + word + "!")

# call the function to read the content of the file!
for filename in filenames:
    read_files(filename)
```

```
The name of the dog is:
- wullf!
The name of the dog is:
- kurdi!
The name of the dog is:
- köpek!
```

In [238]:

```
filenames = ['girls_paper.txt', 'alice.txt', 'siddhartha.txt']

def count_the(filename):
    """Count the number of the word 'the' in a file."""
    try:
        with open(filename) as f_obj:
            contents = f_obj.read()
    except FileNotFoundError:
        msg = "Sorry, the file " + " does not exist."
        print(msg)
    else:
        # Count how many times the word 'the' is in the text.
        words = contents.lower().split()
        the_words = words.count('the')
        print("The file " + filename + " has about " + str(the_words) + " 'th
e's'.")

for filename in filenames:
    count_the(filename)
```

```
The file girls_paper.txt has about 1417 'the's'.
The file alice.txt has about 833 'the's'.
The file siddhartha.txt has about 2395 'the's'.
```

Storing Data

In [67]:

```
import json

numbers = [2, 3, 5, 7, 11, 13]

filename = 'numbers.json'
with open(filename, 'w') as f_obj:
    json.dump(numbers, f_obj)
```

- **Using json.dump() and json.load()**
- we import the json module
- we build a list of numbers and store them
- we chose a filename 'numbers.json' and store it into filename
- we open a file, the second argument is 'w', we write that line
- the json.dump() takes two arguments:
  - our list which contains the numbers
  - the file, which we created as f_obj
- this program has no output but we can check the file, which is correct

In [68]:

```
import json

filename = 'numbers.json'
with open(filename) as f_obj:
    numbers = json.load(f_obj)

print(numbers)
```

```
[2, 3, 5, 7, 11, 13]
```

- this time we open the file in read mode, we just need to read the file
- we use the json.load() function to load the json file into a variable 'numbers'
- we simply print this variable, so we get our numbers

In [69]:

```
import json

username = input("What is your name? ")

filename = 'username.json'
with open(filename, 'w') as f_obj:
    json.dump(username, f_obj)
    print("We'll remember you when you come back, " + username + "!")
```

```
What is your name? Ugur
We'll remember you when you come back, Ugur!
```

- **Saving an Reading User-Generated Data**
- we prompt an username and store it into username
- we use the dump and pass the username (which is an input stored in that variable 'username') and the file (which is f_obj)

In [71]:

```
import json

filename = 'username.json'

with open(filename) as f_obj:
    username = json.load(f_obj)
    print("Welcome back, " + username + "!")
```

```
Welcome back, Ugur!
```

- we load the data with the json.load() function from the file whih is f_obj
- we print the loaded json with our message

In [76]:

```
import json

# Load the username, if it has been stored previously.
# Otherwise, prompt for the username and store it.

filename = 'username.json'
try:
    with open(filename) as f_obj:
        username = json.load(f_obj)
except FileNotFoundError:
    username = input("What is your name? ")
    with open(filename, 'w') as f_obj:
        json.dump(username, f_obj)
        print("We'll remember you when you come bnack, " +  username + "!")
else:
    print("Welcome back, " + username + "!")
```

```
Welcome back, Ugur!
```

- we put both into one
- first we try to open the file username.json with the json.load() function
- if the file exists we do the else block
- if we get an FileNotFoundError we ask the user for input and we use therefore the json.dump() method

In [81]:

```
import json

def get_stored_username():
    """Get stored username if aviable."""
    filename = 'username.json'
    try:
        with open(filename) as f_obj:
            username = json.load(f_obj)
    except FileNotFoundError:
        return None
    else:
        return username

def greet_user():
    """Greet the user by name."""
    username = get_stored_username()
    if username:
        print("Welcome back, " + username + "!")
    else:
        username = input("What is your name? ")
        filename = 'username.json'
        with open(filename, 'w') as f_obj:
            json.dump(username, f_obj)
            print("We'll remember you when you come back, " +  username + "!")

greet_user()
```

```
Welcome back, Ugur!
```

- **Refactoring**
- breaking the code into a series of functions

- the new function *get_stored_username* has a clear purpose
- this function retreives the username
- if the file is missing it returns nothing
- a function should either return a value or None
- the *greet_user()* function takes the username from the function above and stores it into 'username'
- we check if the username is stored or not
- in case we don't have an username we ask the user for input
- we store that
- we write it into the file via json.dump() method and print a message

Tasks

- 10-11. Favorite Number: Write a program that prompts for the user's favorite number. Use json.dump() to store this number in a file. Write a separate pro- gram that reads in this value and prints the message, "I know your favorite number! It's _ ."
- 10-12. Favorite Number Remembered: Combine the two programs from Exercise 10-11 into one file. If the number is already stored, report the favorite number to the user. If not, prompt for the user's favorite number and store it in a file. Run the program twice to see that it works.
- 10-13. Verify User: The final listing for remember_me.py assumes either that the user has already entered their username or that the program is running for the first time. We should modify it in case the current user is not the person who last used the program. Before printing a welcome back message in greet_user(), ask the user if this is the correct username. If it's not, call get_new_username() to get the correct username.

In [288]:

```
# 10-11 & 10-12

import json

# initialization
filename = 'favorite_number_user.json'

# function to get the number and store it
def get_number():
    """Gets the number of the user and stores it in a variable."""
    number = input("What is your favorite number? ")
    with open(filename, 'w') as f_obj:
        json.dump(number, f_obj)
    return int(number)

# function to prompt the number from the .json file
def prompt_number():
    """Reads and prints the number."""
    number = get_number()
    if number == get_number():
        print("The number is the same!")
    else:
        with open(filename) as f_obj:
            json.load(f_obj)
            print("The number before was: " + str(number) + "!")

prompt_number()
```

```
What is your favorite number? 12
What is your favorite number? 12
The number is the same!
```

In [296]:

```
# 10-13

import json

def get_stored_username():
    """Get stored username if aviable."""
    filename = 'username.json'
    try:
        with open(filename) as f_obj:
            username = json.load(f_obj)
    except FileNotFoundError:
        return None
    else:
        return username

def get_new_username():
    """Prompt for a username."""
    username = input("What is your name? ")
    filename = 'username.json'
    with open(filename, 'w') as f_obj:
        json.dump(username, f_obj)
    return username

def greet_user():
    """Greet the user by name."""
    username = get_stored_username()
    if username:
        correct = input("Are you " + username + "? (y/n)")
        if correct == 'y':
            print("Welcome back " + username + "!")
        else:
            username = get_new_username()
            print("We will remember you when you come back " + username + "!")
    else:
        username = get_new_username()
        print("We will remember you when you come back " + username + "!")

greet_user()
```

```
Are you Ugur? (y/n)n
What is your name? Heino
We will remember you when you come back Heino!
```