EN-US

Creating File Handlers and Modules for Retrieving Information about Insulin

Lab overview

In this lab, you will:

- Create a module
- Open a file and load the JSON data it contains using the built-in JSON module of Python
- Parse the JSON structure to access insulin data
- Calculate the rough molecular weight of human insulin using given code (similar to the lab **Working with the String Sequence and Numeric Weight of Insulin in Python**)

Estimated completion time

25 minutes

Accessing the AWS Cloud9 IDE

1. Start your lab environment by going to the top of these instructions and choosing **Start Lab**.

   A **Start Lab** panel opens, displaying the lab status.

2. Wait until you see the message *Lab status: ready*, and then close the **Start Lab** panel by choosing the **X**.

3. At the top of these instructions, choose **AWS**.

3. At the top of these instructions, choose **AWS**.

   The AWS Management Console opens in a new browser tab. The system automatically logs you in.

   > **Note:** If a new browser tab does not open, a banner or icon at the top of your browser typically indicates that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop ups**.

4. In the AWS Management Console, choose **Services** > **Cloud9**. In the **Your environments** panel, locate the **reStart-python-cloud9** card, and choose **Open IDE**.

   The AWS Cloud9 environment opens.

   > **Note:** If a pop-up window opens with the message *.c9/project.settings have been changed on disk*, choose **Discard** to ignore it. Likewise, if a dialog window prompts you to *Show third-party content*, choose **No** to decline.

Creating your Python exercise File

5. From the menu bar, choose **File -> New from template -> Python File**.

   This action creates an untitled file.

6. Delete the sample code from the template file.

7. Choose **File -> Save As...**, provide a suitable name for the exercise file (for example, calc_weight_json.py), and save it under the **/home/ec2-user/environment** directory.

8. Create a second file and name it jsonFileHandler.py.

   > **Note**: The **.py** is the extension for Python files.

EN-US

9. Create a directory called **files**.

Accessing the terminal session

10. In your AWS Cloud9 IDE, choose the **+** icon and select **New Terminal**.

    A terminal session opens.

11. To display the present working directory, enter `pwd`. This command points to **/home/ec2-user/environment**.

12. In this directory, locate the file that you created in the previous section.

Exercise 1: Creating the JSON molecules data file

This JSON document stores all the information of previous lab, such as the insulin molecules, the numeric weights of the amino acids and the actual weight of the insulin molecule

13. From the menu bar, choose **File -> New File**.

14. Copy and paste the following code into this newly created file:

```
{
   "molecules":{
      "lsInsulin":"malwmrllpllallalwgpdpaaa",
      "bInsulin":"fvnqhlcgshlvealylvcgergffytpkt",
      "aInsulin":"giveqcctsicslyqlenycn",
      "cInsulin":"rreaedlqvgqvelgggpgagslqplalegslqkr"
   },
   "weights":{
      "A":89.09,
      "C":121.16,
      "D":133.10,
```

EN-US

This JSON document stores all the information of previous lab, such as the insulin molecules, the numeric weights of the amino acids and the actual weight of the insulin molecule

13. From the menu bar, choose **File -> New File**.
14. Copy and paste the following code into this newly created file:

```
{
   "molecules":{
      "lsInsulin":"malwmrllpllallalwgpdpaaa",
      "bInsulin":"fvnqhlcgshlvealylvcgergffytpkt",
      "aInsulin":"giveqcctsicslyqlenycn",
      "cInsulin":"rreaedlqvgqvelgggpgagslqplalegslqkr"
   },
   "weights":{
      "A":89.09,
      "C":121.16,
      "D":133.10,
      "E":147.13,
      "F":165.19,
      "G":75.07,
      "H":155.16,
      "I":131.17,
      "K":146.19,
      "L":131.17,
      "M":149.21,
      "N":132.12,
      "P":115.13,
      "Q":146.15,
      "R":174.20,
      "S":105.09,
      "T":119.12,
      "V":117.15,
      "W":204.23,
      "Y":181.19
   },
   "molecularWeightInsulinActual":5807.63
}
```

EN-US

15. To save the file as **insulin.json** in the **files** folder, select **File -> Save As...**.
16. In the **Save As** pop-up window for **Filename:**, enter `insulin.json`
17. For **Folder:** enter `files` or choose the **files** folder.

Exercise 2: Creating the JSON file handler module

In this task, you create a module that reads the JSON file and returns the JSON document.

18. Choose the **jsonFileHandler.py** file.
19. Import JSON to begin your work:

```
import json
```

20. Define the function that will read the file:

```
def readJsonFile(fileName):
```

21. Below the file definition, add a data variable as an empty string:

```
data=""
```

22. For the body of the function, open the json file using the **open** function, and parse the file using **json.load**.

```
def readJsonFile(fileName):
    data = ""
    with open('files/insulin.json') as json_file:
        data = json.load(json_file)
    return data
```

open returns a file handler to the **files/insulin.json** file.

json.load reads the JSON file and returns the content as a Python dictionary.

23. Add a **try/except** block to make this function more reliable:

```
import json

def readJsonFile(fileName):
    data = ""
    try:
        with open(fileName) as json_file:
            data = json.load(json_file)
    except IOError:
        print("Could not read file")
    return data
```

In case the file cannot be opened, the program will display the error *Could not read file*.

The returned **data** string is empty in case the open file method fails.

You created a **jsonFileHandle** module that you can import in other Python files to access the **readJsonFile** function.

Exercise 3: Creating the main program

You create the main program that parses the JSON data and calculates the molecular weight as you did in a previous lab.

24. First, import the **jsonFileHandle** module. Open the **calc_weight_json.py** file and add the following:

```
import jsonFileHandler
```


25. Retrieve the the JSON data and store it in a **data** variable.

EN-US

25. Retrieve the the JSON data and store it in a **data** variable.

```
data = jsonFileHandler.readJsonFile('files/insulin.json')
```

26. Test if the returned data is not empty and obtain the insulin data.

```
if data != "" :
    bInsulin = data['molecules']['bInsulin']
    aInsulin = data['molecules']['aInsulin']
    insulin = bInsulin + aInsulin
    molecularWeightInsulinActual = data['molecularWeightInsulinActual']
    print('bInsulin: ' + bInsulin)
    print('aInsulin: ' + aInsulin)
    print('molecularWeightInsulinActual: ' + str(molecularWeightInsulinActual))
else:
    print("Error. Exiting program")
```

27. You can run the program to see if the data is well retrieved. The results should be as follows:

```
bInsulin: fvnqhlcgshlvealylvcgergffytpkt
aInsulin: giveqcctsicslyqlenycn
molecularWeightInsulinActual: 5807.63
```

28. You can also test what happens if the file is not found. For example, change the name of the file to **'files/insuline.json'**, and run the program. You will get the following message:

```
Could not read file
Error. Exiting program
```

29. Undo the last change so that the file is named **files/insulin.json** again.

30. In the **if** section of the code below the last **print**, add the following code:

```
# Calculating the molecular weight of insulin
# Getting a list of the amino acid (AA) weights
```

EN-US

30. In the **if** section of the code below the last **print**, add the following code:

```
# Calculating the molecular weight of insulin
# Getting a list of the amino acid (AA) weights
aaWeights = data['weights']
# Count the number of each amino acids
aaCountInsulin = ({x: float(insulin.upper().count(x)) for x in ['A','C','D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R','S', 'T','V', 'W', 'Y']})
# Multiply the count by the weights
molecularWeightInsulin = sum({x: (aaCountInsulin[x]*aaWeights[x]) for x in
['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R','S', 'T', 'V', 'W', 'Y']}.values())
print("The rough molecular weight of insulin: " +
str(molecularWeightInsulin))
print("Percent error: " + str(((molecularWeightInsulin - molecularWeightInsulinActual)/molecularWeightInsulinActual)*100))
```

Welcome × {} insulin.json × jsonFileHandler.py × calcWeight_json.py ×

```
import jsonFileHandler

data = jsonFileHandler.readJsonFile('files/insulin.json')

if data != "" :
    bInsulin = data['molecules']['bInsulin']
    aInsulin = data['molecules']['aInsulin']
    insulin = bInsulin + aInsulin
    molecularWeightInsulinActual = data['molecularWeightInsulinActual']
    print('bInsulin: ' + bInsulin)
    print('aInsulin: ' + aInsulin)
    print('molecularWeightInsulinActual: ' + str(molecularWeightInsulinActual))

    # Calculating the molecular weight of insulin
    # Getting a list of the amino acid (AA) weights
    aaWeights = data['weights']
    # Count the number of each amino acids
    aaCountInsulin = ({x: float(insulin.upper().count(x)) for x in ['A', 'C',
    'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T',
    'V', 'W', 'Y']})
    # Multiply the count by the weights
    molecularWeightInsulin = sum({x: (aaCountInsulin[x]*aaWeights[x]) for x in
    ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R',
    'S', 'T', 'V', 'W', 'Y']}.values())
    print("The rough molecular weight of insulin: " +
    str(molecularWeightInsulin))
    print("Percent error: " + str(((molecularWeightInsulin - molecularWeightInsulinActual)/molecularWeightInsulinActual)*100))
```

EN-US

30. In the **if** section of the code below the last **print**, add the following code:

```
# Calculating the molecular weight of insulin
# Getting a list of the amino acid (AA) weights
aaWeights = data['weights']
# Count the number of each amino acids
aaCountInsulin = ({x: float(insulin.upper().count(x)) for x in ['A','C','D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R','S', 'T','V', 'W', 'Y']})
# Multiply the count by the weights
molecularWeightInsulin = sum({x: (aaCountInsulin[x]*aaWeights[x]) for x in
['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R','S', 'T', 'V', 'W', 'Y']}.values())
print("The rough molecular weight of insulin: " +
str(molecularWeightInsulin))
print("Percent error: " + str(((molecularWeightInsulin - molecularWeightInsulinActual)/molecularWeightInsulinActual)*100))
```

Welcome × {} insulin.json × jsonFileHandler.py × calcWeight_json.py ×

```
import jsonFileHandler

data = jsonFileHandler.readJsonFile('files/insulin.json')

if data != "" :
    bInsulin = data['molecules']['bInsulin']
    aInsulin = data['molecules']['aInsulin']
    insulin = bInsulin + aInsulin
    molecularWeightInsulinActual = data['molecularWeightInsulinActual']
    print('bInsulin: ' + bInsulin)
    print('aInsulin: ' + aInsulin)
    print('molecularWeightInsulinActual: ' + str(molecularWeightInsulinActual))

    # Calculating the molecular weight of insulin
    # Getting a list of the amino acid (AA) weights
    aaWeights = data['weights']
    # Count the number of each amino acids
    aaCountInsulin = ({x: float(insulin.upper().count(x)) for x in ['A', 'C',
    'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T',
    'V', 'W', 'Y']})
    # Multiply the count by the weights
    molecularWeightInsulin = sum({x: (aaCountInsulin[x]*aaWeights[x]) for x in
    ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R',
    'S', 'T', 'V', 'W', 'Y']}.values())
    print("The rough molecular weight of insulin: " +
    str(molecularWeightInsulin))
    print("Percent error: " + str(((molecularWeightInsulin - molecularWeightInsulinActual)/molecularWeightInsulinActual)*100))
```

EN-US

Welcome | {} insulin.json | jsonFileHandler.py | calcWeight_json.py

```
import jsonFileHandler

data = jsonFileHandler.readJsonFile('files/insulin.json')

if data != "" :
    bInsulin = data['molecules']['bInsulin']
    aInsulin = data['molecules']['aInsulin']
    insulin = bInsulin + aInsulin
    molecularWeightInsulinActual = data['molecularWeightInsulinActual']
    print('bInsulin: ' + bInsulin)
    print('aInsulin: ' + aInsulin)
    print('molecularWeightInsulinActual: ' + str(molecularWeightInsulinActual))

    # Calculating the molecular weight of insulin
    # Getting a list of the amino acid (AA) weights
    aaWeights = data['weights']
    # Count the number of each amino acids
    aaCountInsulin = ({x: float(insulin.upper().count(x)) for x in ['A', 'C',
    'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T',
    'V', 'W', 'Y']})
    # Multiply the count by the weights
    molecularWeightInsulin = sum({x: (aaCountInsulin[x]*aaWeights[x]) for x in
    ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R',
    'S', 'T', 'V', 'W', 'Y']}.values())
    print("The rough molecular weight of insulin: " +
    str(molecularWeightInsulin))
    print("Percent error: " + str(((molecularWeightInsulin - molecularWeightInsulinActual)/molecularWeightInsulinActual)*100))
else :
    print("Error. Exiting program")
```

31. Run the program. You will get the following:

```
bInsulin: fvnqhlcgshlvealylvcgergffytpkt
aInsulin: giveqcctsicslyqlenycn
molecularWeightInsulinActual: 5807.63
The rough molecular weight of insulin: 6696.420000000001
Percent error: 15.30383306099047
```

EN-US

```
        print("The rough molecular weight of insulin: " +
        str(molecularWeightInsulin))
        print("Percent error: " + str(((molecularWeightInsulin - molecularWeightInsulinActual)/molecularWeightInsulinActual)*100))
    else :
        print("Error. Exiting program")
```

31. Run the program. You will get the following:

```
bInsulin: fvnqhlcgshlvealylvcgergffytpkt
aInsulin: giveqcctsicslyqlenycn
molecularWeightInsulinActual: 5807.63
The rough molecular weight of insulin: 6696.420000000001
Percent error: 15.30383306099047
```

End Lab

🏁 Congratulations! You have completed the lab.

32. Choose ■ **End Lab** at the top of this page, and then select Yes to confirm that you want to end the lab.

    A panel indicates that *DELETE has been initiated... You may close this message box now.*

33. A message *Ended AWS Lab Successfully* is briefly displayed, indicating that the lab has ended.

Additional Resources

For more information about AWS Training and Certification, see https://aws.amazon.com/training/.

Your feedback is welcome and appreciated. If you would like to share any suggestions or corrections, please provide the details in our AWS Training and Certification Contact Form.