Working with the String Sequence and Numeric Weight of Insulin in Python

Lab overview

In the Python Basics module, you learned about variables, comments, math, concatenations, and exceptions. Now, you will apply what you have learned to the real-world application of human insulin.

You will store the protein sequence of human preproinsulin in a string variable and the weight of preproinsulin in int and float variables. Next, you will print these variables to the console, with comments that explain the code. You will do basic math and string concatenations.

In this lab, you will:

- Add comments that explain the intention and flow of your code
- Use `print()` to print elements of your Python code to the console
- Use string manipulations to get the sequence of insulin from preproinsulin
- Do basic math on the molecular weight and sequence of insulin
- Assign string, int, and float variables to numbers that represent the weight of insulin
- Explore Python exceptions

Estimated completion time

30 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**.

   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...**, and provide a suitable name for the exercise file (for example, *string-insulin.py*) and save it under the **/home/ec2-user/environment** directory.

Accessing the terminal session

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

   A terminal session opens.

9. To display the present working directory, enter `pwd`. This command points to **/home/ec2-user/environment**. In this directory, you should also be able to locate the file you created in the previous section.

Exercise 1: Assigning variables to the sequence elements of human insulin

In this exercise, you will create variables and assign a string value to them.

10. From the navigation pane of the IDE, choose the file that you created in the previous *Creating your Python exercise file* section.

> **How to start your .py file**
>
> You should always start your Python file with comments. Recall that Python comments start with a pound sign (#).
>
> Your first comments should provide:
>
> - The Python version (*python3.6*) with a path to the executable, if possible
> - The encoding of the file (typically, *coding: utf-8*)

11. Write the following note on the next line:

```
# Store the human preproinsulin sequence in a variable called preproinsulin:
```

12. Create the first variable in the Python file by entering `preproInsulin =` as the name of the variable, and with the equals sign (=) as the assignment operator.

13. After the equal sign (=), enter the following input:

```
"malwmrllpllallalwgpdpaaafvnqhlcgshlvealylvcgergffytpktr" \
```

13. After the equal sign (=), enter the following input:

```
"malwmrllpllallalwgpdpaaafvnqhlcgshlvealylvcgergffytpktr" \
"reaedlqvgqvelgggpgagslqplalegslqkrgiveqcctsicslyqlenycn"
```

14. To finalize the first variable on that line, press ENTER.

    Maximum length of lines in Python files and other PEP standards

    The trailing backslash (\) in variable value from the previous step is used to maintain compliance with the Python Enhancement Proposals (PEP) 8 style guide. The PEP 8 style guide recommends a maximum of 79 characters per line. PEPs are standards for Python best practices. Though the file still runs with longer line lengths, sticking to the suggested limit increases simplicity and readability. By using a backslash (\), you can split variables and code into smaller blocks, thereby maintaining the 79-character limit.

15. Write a note in the file:

```
# Store the remaining sequence elements of human insulin in variables:
```

16. Repeat the steps to define a variable and assign a value to it by using the information from the following chart. Use an equal sign (=) between the variable name and string.

Variable Name	String to Save to Variable
lslnsulin	"malwmrllpllallalwgpdpaaa"
blnsulin	"fvnqhlcgshlvealylvcgergffytpkt"
alnsulin	"giveqcctsicslyqlenycn"
clnsulin	"rreaedlqvgqvelgggpgagslqplalegslqkr"

> **Note**: Variable names in Python usually begin with a lowercase first word, and then uppercase for each following word, without underscores or spaces. Be consistent when you name your variables.

17. Finally, you will merge the results of the smaller insulin groupings into a single variable called *insulin*. To do this, on a new line, enter: `insulin = bInsulin + aInsulin`

Exercise 3: Using print() to display sequences of human insulin to the console

In this exercise, you will use the `print()` built-in method to display sequence elements of human insulin in the console.

18. Write a note on the next line:

```
# Printing "the sequence of human insulin" to console using successive print() commands:
```

19. On a new line of the Python file, enter: `print("The sequence of human preproinsulin:")`
20. Press ENTER.

    This `print()` statement prints the direct representation of the provided string, with no formatting.

21. To print a string that is contained in a variable from your script, enter: `print(preproInsulin)`
22. Press ENTER.
23. Enter the following comment:

```
# Printing to console using concatenated strings inside the print function (one-liner):
```

24. To concatenate strings, use the plus sign (+) in the `print()` statement:

```
print("The sequence of human insulin, chain a: " + aInsulin)
```

25. Press ENTER.

> **Note:** The built-in `print()` function accepts multiple arguments that can accomplish the same task in step 5. For example:
>
> `print("The sequence of human insulin, chain a:", aInsulin)`

25. Press ENTER.

> **Note:** The built-in `print()` function accepts multiple arguments that can accomplish the same task in step 5. For example:
>
> `print("The sequence of human insulin, chain a:", aInsulin)`

26. Save and run the file.

Exercise 4: Calculating the rough molecular weight of human insulin using the given code

In this lab, you will calculate the molecular weight of insulin, which you will work with in later labs.

27. Ensure that your **.py** file is open

28. Copy the following code, and at the end of the **.py** file, paste it.

```
# Calculating the molecular weight of insulin
# Creating a list of the amino acid (AA) weights
aaWeights = {'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}
# 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))
```


29. Save and run the file.

29. Save and run the file.
30. Notice the resulting output. You will use elements of this code to work with loops and functions in other labs, so observe how the code is written and try to follow the expected output.

> **Note:** The actual molecular weight of human insulin is 5807.63, but the program delivers 6696.42 because it ignores certain bonds and post-translational processing. To calculate the error percentage:*error percentage = (| measured – accepted | / accepted)*100%*

31. Enter or copy the example into your script.

```
molecularWeightInsulinActual = 5807.63
print("Error percentage: " + str(((molecularWeightInsulin - molecularWeightInsulinActual)/molecularWeightInsulinActual)*100))
```

32. To see the error percentage, run and save the file.

> **Note:** When you use string concatenation with floating point calculations, the `print()` function returns an error. This error is handled by a method called *casting*, which tells Python to use a certain data type. The previous use of the `str()` function is an example of casting.

Congratulations! You have worked with variables and different data types in a Python function.

End Lab

🏁 Congratulations! You have completed the lab.

33. 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.*

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