🧬

Lab 122: Cálculo de la carga neta de la insulina mediante listas y bucles de Python

🔗 Link: https://awsrestart.instructure.com/courses/1632/modules/items/886836

📁 Repo: https://github.com/francopig/aws-python/tree/main/12. Calculo de la carga neta de la insulina mediante listas y bucles de Python

Calculating the Net Charge of Insulin by Using Python Lists and Loops

Lab overview

In the Flow Control module, you learned about `if-else` statements, `while` loops, lists, and `for` loops. Now you will apply what you have learned to the real-world application of human insulin.

Here, you will use `lists`, `for` and `while` loops, and basic math to calculate the net charge of insulin from pH 0 to pH 14.

In this lab, you will:

- Create a dictionary of pKa values (which indicate the strength of an acid) that will be used in the net charge calculations
- Use the `count()` method to get a count of amino acids
- Use a `while` loop to calculate the net charge of insulin from pH 0 to pH 14

Exercise 1: Assigning variables, lists, and dictionaries

1. From the navigation pane of the IDE, choose the file that you created in the previous *Creating your Python exercise file* section.
2. Copy the following code, paste it into the file, and save the file:

```
# Python3.6
# Coding: utf-8
# Store the human preproinsulin sequence in a variable called preproinsulin:
preproInsulin = "malwmrllpllallalwgpdpaaafvnqhlcgshlvealylvcgergffytpktrreaedlqvgqvelgggpgagslqplalegslqkrgiveqcctsicslyq
lenycn"
# Store the remaining sequence elements of human insulin in variables:
lsInsulin = "malwmrllpllallalwgpdpaaa"
bInsulin = "fvnqhlcgshlvealylvcgergffytpkt"
aInsulin = "giveqcctsicslyqlenycn"
cInsulin = "rreaedlqvgqvelgggpgagslqplalegslqkr"
insulin = bInsulin + aInsulin
```

3. On the next line, create a new dictionary by entering: `pKR = {}`
4. To fill the dictionary with key-value pairs, insert the first key of *y* with a value of *10.07*. Place the cursor inside the braces, and enter: `'y': 10.07,`

> Note: You included a comma after the value so that you can add the remaining key-value pairs.

5. To match the code segment, add the following key-value pairs into the dictionary.
   - `'c': 8.18`
   - `'k': 10.53`
   - `'h': 6.00`

- `'r': 12.48`
- `'d': 3.65`
- `'e': 4.25`

The dictionary should look like the following code:

```
pKR = {'y':10.07,'c': 8.18,'k':10.53,'h':6.00,'r':12.48,'d':3.65,'e':4.25}
```

> Note: Y, C, K, H, R, D, and E are the only amino acids that contribute to the net-charge calculation.

lab122.py

```
# Python3.6
# Coding: utf-8

# Store the human preproinsulin sequence in a variable called preproinsulin:
preproInsulin = "malwmrllpllallalwgpdpaaafvnqhlcgshlvealylvcgergffytpktrreaedlqvgqvelgggpgagslqplalegslqkrgiveqcctsicslyqlenycn"

# Store the remaining sequence elements of human insulin in variables:
lsInsulin = "malwmrllpllallalwgpdpaaa"
bInsulin = "fvnqhlcgshlvealylvcgergffytpkt"
aInsulin = "giveqcctsicslyqlenycn"
cInsulin = "rreaedlqvgqvelgggpgagslqplalegslqkr"
insulin = bInsulin + aInsulin

pKR = {'y':10.07,'c': 8.18,'k':10.53,'h':6.00,'r':12.48,'d':3.65,'e':4.25}
```

Exercise 2: Using count() to count the numbers of each amino acid

In this exercise, you will use the `count()` method and list comprehension to count the number of Y, C, K, H, R, D, and E amino acids. These amino acids contribute to the net charge.

1. To identify a count of an item within a list, you can use the `count()` method. To see how many amino acids in insulin are *Y*, use the `count()` method by entering: `insulin.count("Y")`
2. Next, update the `insulin.count()` line by casting the variable returned by the `count()` method as a float: `float(insulin.count("Y"))`
3. Now that you have the basis for identifying a single entity, you can use this method to find all entities from a list. This process can be done by using list comprehension. For the entire line, enter: `seqCount = ({x: float(insulin.count(x)) for x in ['y','c','k','h','r','d','e']})`

> Note: The first two steps in this exercise are predecessors to the third step.

lab122.py

```
# Coding: utf-8

# Store the human preproinsulin sequence in a variable called preproinsulin:
preproInsulin = "malwmrllpllallalwgpdpaaafvnqhlcgshlvealylvcgergffytpktrreaedlqvgqvelgggpgagslqplalegslqkrgiveqcc

# Store the remaining sequence elements of human insulin in variables:
lsInsulin = "malwmrllpllallalwgpdpaaa"
bInsulin = "fvnqhlcgshlvealylvcgergffytpkt"
aInsulin = "giveqcctsicslyqlenycn"
cInsulin = "rreaedlqvgqvelgggpgagslqplalegslqkr"
insulin = bInsulin + aInsulin

pKR = {'y':10.07,'c': 8.18,'k':10.53,'h':6.00,'r':12.48,'d':3.65,'e':4.25}
#🧐👆 Y, C, K, H, R, D, and E are the only amino acids that contribute to the net-charge calculation.

seqCount = ({x: float(insulin.count(x)) for x in ['y','c','k','h','r','d','e']})
print(seqCount)
```

13:1 Python Spaces: 4

programacion/lab122/lab1

Run Command: programacion/lab122/lab122.py Runner: Python 3 CWD ENV

```
0.0
{'y': 4.0, 'c': 6.0, 'k': 1.0, 'h': 2.0, 'r': 1.0, 'd': 0.0, 'e': 4.0}
```

Exercise 3: Writing the net charge formula

In this exercise, you will construct the net charge formula. You will use the provided netCharge variable in a Python-based net charge formula. The function for the formula includes a `while` loop that will print the net charge while the pH variable is equal to or below 14.

1. Create a variable called *pH* and initialize it to zero by entering `pH = 0` and pressing ENTER.
2. Create the `while` loop by entering `while (pH <= 14):` and pressing ENTER.
3. Copy the following *netCharge* variable and paste it at the beginning of the `while` loop.

```
netCharge = (
    +(sum({x: ((seqCount[x]*(10**pKR[x]))/((10**pH)+(10**pKR[x]))) \
    for x in ['k','h','r']}.values()))
    -(sum({x: ((seqCount[x]*(10**pH))/((10**pH)+(10**pKR[x]))) \
    for x in ['y','c','d','e']}.values())))
```

4. To print the *netCharge* variable with the *pH*, use a format string for better readability. Enter `print('{0:.2f}'.format(pH), netCharge)` and press ENTER.
5. Finally, increment the *pH* variable by entering `pH +=1` and pressing ENTER.
6. Save and run the file.

```
seqCount = ({x: float(insulin.count(x)) for x in ['y','c','k','h','r','d','e']})
print(seqCount)

pH = 0
while (pH <= 14):
    netCharge = (
    +(sum({x: ((seqCount[x]*(10**pKR[x]))/((10**pH)+(10**pKR[x]))) \
    for x in ['k','h','r']}.values()))
    -(sum({x: ((seqCount[x]*(10**pH))/((10**pH)+(10**pKR[x]))) \
    for x in ['y','c','d','e']}.values())))
    print('{0:.2f}'.format(pH), netCharge)
    pH +=1
```

25:65 Python Spaces: 4

programacion/lab122/lab1

Run Command: programacion/lab122/lab122.py Runner: Python 3 CWD ENV

```
{'y': 4.0, 'c': 6.0, 'k': 1.0, 'h': 2.0, 'r': 1.0, 'd': 0.0, 'e': 4.0}
0.00 3.999773036108418
1.00 3.997731498980362
2.00 3.9774281495747488
3.00 3.7850011255505267
4.00 2.540057926152534
5.00 0.4181014741593989
6.00 -0.9698633190556047
7.00 -2.186629624776233
8.00 -4.403272105402016
9.00 -7.551924685030059
10.00 -9.980672054414311
11.00 -12.349327503864025
12.00 -13.168625084165932
13.00 -13.75988990979344
14.00 -13.969868023471733
```

Be careful about indentation and spacing in Python

Subsets of Python code are organized by indentation and spaces. In Python, even one misplaced indentation or space can throw an exception or other error. For example, be sure that every item within your `while` loop is properly indented so the code will work.

Congratulations! You have worked with lists and loops in a Python function.

End Lab

Congratulations! You have completed the lab.

1. 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.*

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