November 2022

Computational Thinking and ABE

By Aynur Elif Kekec Bulut, ABE Türkiye

ABE Master Teacher Fellowship Program

AMGEN® Biotech Experience

Scientific Discovery for the Classroom

The curriculum projects designed by the 2021–22 ABE Master Teacher Fellows are a compilation of curricula and materials that are aligned with Amgen Biotech Experience (ABE) and prepare students further in their biotechnology education. These projects were created over the course of a 1-year Fellowship in an area of each Fellow's own interest. Each is unique and can be adapted to fit the needs of your individual classroom. Objectives and goals are provided, along with expected outcomes. Projects can be used in conjunction with your current ABE curriculum or as an extension.

As a condition of the Fellowship, these classroom resources may be downloaded and used by other teachers for free. The projects are not edited or revised by the ABE Program Office (for content, clarity, or language) except to ensure safety protocols have been clearly included where appropriate. We are grateful to the ABE Master Teacher Fellows for sharing their work with the ABE community.

If you have questions about any of the curriculum pieces, please reach out to us at ABEInfo@edc.org, and we will be happy to connect you with the author and provide any assistance needed.

AMGEN® Biotech Experience
Scientific Discovery for the Classroom
Türkiye

ABE Master Teacher Fellowship

Computation Thinking and ABE

NAME: Aynur Elif Kekec Balut
PROGRAM SITE: ABE Türkiye

LESSON OVERVİEW

For this activity, no instructions are provided. Instead, students will use examples of what imaginary players have done to figure out how to play the game. This lesson gives students the opportunity to practice the four parts of computational thinking (decomposition, pattern matching, abstraction, and algorithms) in one cohesive activity.

TEACHİNG SUMMARY

Getting Started - 15 minutes

1) Vocabulary
2) Figuring it Out

Activity: Computational Thinking - 25 minutes

3) Computational Thinking

Wrap-up - 10 minutes

4) Flash Chat - What did we learn?
5) Vocab Shmocab

Assessment - 5 minutes

6) Computational Thinking Assessment

LESSON OBJECTİVES

Students will:

- Analyze information to draw conclusions
- Match identical portions of similar phrases to match patterns
- Identify differences in similar phrases and abstract them out

AMGEN® Biotech Experience
Scientific Discovery for the Classroom
Türkiye

TEACHİNG GUİDE

MATERİALS, RESOURCES, AND PREP

For the Student

- One die per group
- One Computational Thinking Kit per group
- Pens, Pencils, & Scissors
- Computational Thinking Assessment for each student

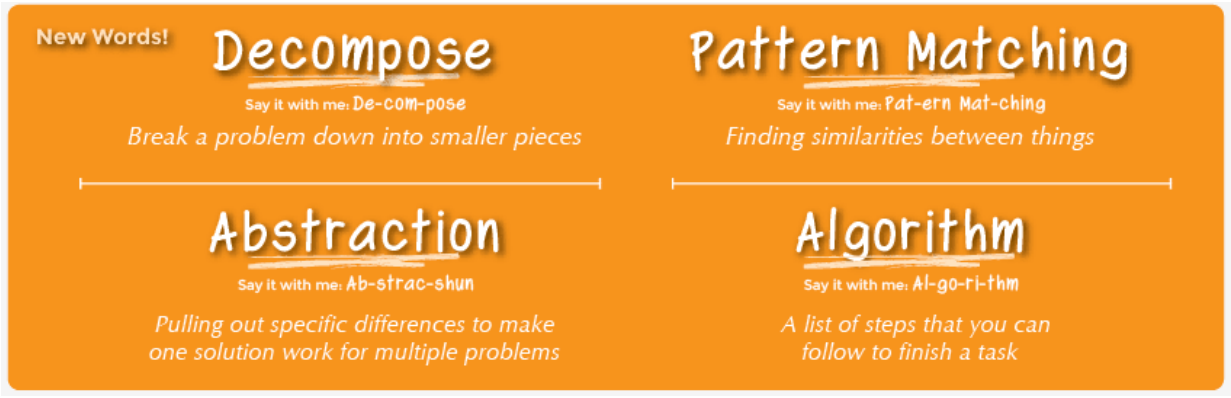
For the Teacher

- Lesson Video
- This Teacher Lesson Guide
- Print one Computational Thinking Kit per group
- Print one Computational Thinking Assessment for each student

GETTİNG STARTED (15 MİN)

1) Vocabulary

This lesson has four new and important words:



Algorithm - Say it with me: Al-go-ri-thm
A list of steps that you can follow to finish a task

Decompose - Say it with me: De-com-pose
Break a problem down into smaller pieces

Abstraction - Say it with me: Ab-strac-shun
Pulling out specific differences to make one solution work for multiple problems

Pattern Matching - Say it with me: Pat-ern Matching
Finding similarities between things

2) Figuring It Out

- Tell your students that you want them to sum up all of the numbers between 1 & 200.
  - Use your body language to indicate that this is not a "serious" or graded exercise.
    - Now, let them know that they must do it all in their heads.
    - Add the time constraint of thirty seconds.
    - They may feel overwhelmed. This is intentional. You can indicate with your tone and demeanor that you might be crazy asking this of them, but begin timing with a resounding: "Starting NOW".
- Watch the class as you keep time. How many are lost in thought?
- When time is up, ask if anyone was able to get the total.
- Ask if there is anyone who thought the problem was so hard that they didn't even attempt it.
- Did anyone attempt it and just not finish?
  - What did they try?
- Guide students toward thinking a little smaller.
  - If we break the problem up into smaller pieces, it becomes easier to manage.
  - Let's start at the two ends. What is 200 + 1?
  - What is 199 + 2?
  - What is 198 + 3?
  - See a pattern?
  - How many of these pairs will we have?
    - What is the last pair we will find? 100 + 101
    - That means that we have 100 total pairs.
    - If we have 100 total pairs of sums of 201, how do we find the final total?
    - What is 100 * 201?
      - Now, what if we wanted to find the trick to do this with other numbers?
  - Can we do it easily with 2,000?
  - How about 20,000?
  - What stays the same? What is different?
  - If we use abstractions to make our end goal something that can change (say we name it "blank") then we can make an algorithm that will work for any number
- Work through the problem until you ultimately get ? = ("blank"/2) * ("blank"+1)
- Do a few simple examples to show that the algorithm is correct for blanks= 2, 3, 4, & 5.

"This is all to show that if you use the tools of Computational Thinking (decomposition, pattern matching, abstraction, and algorithms), then you can figure out how to solve problems that no one has already taught you how to solve...just like we did here! This will be an extremely powerful skill for the rest of your life!"

ACTİVİTİES (25 MİN)

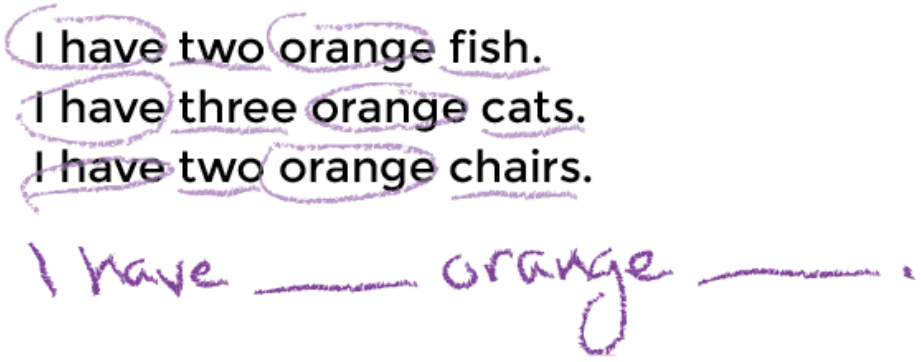
3) Computational Thinking

This lesson is all about a "Game with No Instructions." Students will be charged with figuring out how to play the game as a small group. The small details of their final algorithm are unimportant. What *is* important is that they were able to take a huge task like "figuring out how to play a game on their own" and take small steps toward achieving the goal.

Students will be guided toward discovering the rules using the steps of computational thinking. Resist the temptation to point the students toward "doing it right" and allow them just to do it on their own. If they feel stumped or confused, encourage the students to look at the information that has been given to them, or if they must, ask a classmate.

Directions:

1) Divide students into groups of 2–4.
2) Have the groups read over user experiences to get an idea of how other students have played the “Game with No Instructions.”
3) Encourage them to pattern match between each experience by circling the sections of words that are identical from player to player.
4) Next, have them abstract away differences from each experience by underlining words that change from player to player.
5) Using pattern matching and abstraction, have them make a script template for game play by writing up the circled parts of the other students’ experiences, and leaving the underlined sections as blanks. For example:



6) Give students a blank sheet of paper to write a list of instructions for how they think this game should be played based on the user experiences that they just read. This will be their algorithm.
7) Have students play the game using the algorithm that they just made. Each player should get at least two turns.

WRAP-UP (5 MİN)

4) Flash Chat: What did we learn?

- What should you try to do when you're asked to do something and you don't know how?
- If a problem is too hard, what should you try to do?
- If you find similarities in lots of solutions to different problems, what does that probably tell you?
- If you have a problem that is just a little different from a problem that you have a solution for, what would you do?

LESSON TİP

Flash Chat questions are intended to spark big-picture thinking about how the lesson relates to the greater world and the students' greater future. Use your knowledge of your classroom to decide if you want to discuss these as a class, in groups, or with an elbow partner.

5) Vocab Shmocab

- Which one of these definitions did we learn a word for today?

  "Bringing two pieces together"
  "Breaking a problem down into smaller pieces"
  "An educated guess"

  ...and what is the word that we learned?

ASSESSMENT (5 MİN)

6) Computational Thinking Assessment

- Hand out the assessment worksheet and allow students to complete the activity independently after the instructions have been well explained.
- This should feel familiar, thanks to the previous activities.

CONNECTİONS AND BACKGROUND INFORMATİON

ISTE Standards (formerly NETS)

- 1.a - Apply existing knowledge to generate new ideas, products, or processes.
- 1.c - Use models and simulation to explore complex systems and issues.
- 2.d - Contribute to project teams to solve problems.
- 4.b - Plan and manage activities to develop a solution or complete a project.
- 4.d - Use multiple processes and diverse perspectives to explore alternative solutions.

CSTA K-12 Computer Science Standards

- CPP.L1:6-05. Construct a program as a set of step-by-step instructions to be acted out.
- CT.L1:6-02. Develop a simple understanding of an algorithm using computer-free exercises.
- CT.L2-01. Use the basic steps in algorithmic problem solving to design solutions.
- CT.L2-06. Describe and analyze a sequence of instructions being followed.
- CT.L2-08. Use visual representations of problem states, structures, and data.
- CT.L2-12. Use abstraction to decompose a problem into sub problems.
- CT.L2-14. Examine connections between elements of mathematics and computer science including binary numbers, logic, sets, and functions.

NGSS Science and Engineering Practices

- 3-5-ETS1-2. Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem.

Common Core Mathematical Practices

- 1. Make sense of problems and persevere in solving them.
- 2. Reason abstractly and quantitatively.
- 3. Construct viable arguments and critique the reasoning of others.
- 6. Attend to precision.
- 7. Look for and make use of structure.
- 8. Look for and express regularity in repeated reasoning.

Common Core Math Standards

- 3.OA.3 - Use multiplication and division within 100 to solve word problems in situations involving equal groups, arrays, and measurement quantities.
- 4.NBT.B.4 - Fluently add and subtract multi-digit whole numbers using the standard algorithm.
- 5.NBT.B.5 - Fluently multiply multi-digit whole numbers using the standard algorithm.

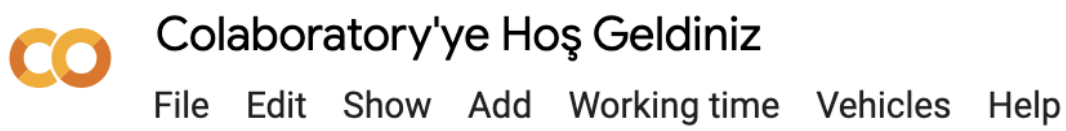
Common Core Language Arts Standards

- SL.3.1 - Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 3 topics and texts, building on others' ideas and expressing their own clearly.
- SL.3.3 - Ask and answer questions about information from a speaker, offering appropriate elaboration and detail.
- L.3.6 - Acquire and use accurately grade-appropriate conversational, general academic, and domain-specific words and phrases, including those that signal spatial and temporal relationships.
- SL.4.1 - Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 4 topics and texts, building on others' ideas and expressing their own clearly.
- L.4.6 - Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases, including those that signal precise actions, emotions, or states of being and that are basic to a particular topic.
- SL.5.1 - Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 5 topics and texts, building on others' ideas and expressing their own clearly.
- L.5.6 - Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases, including those that signal contrast, addition, and other logical relationships

Computational Thinking Co Lab Instructions

1) Co Lab

Sign in to this website: https://colab.research.google.com/notebooks/welcome.ipynb?hl=tr

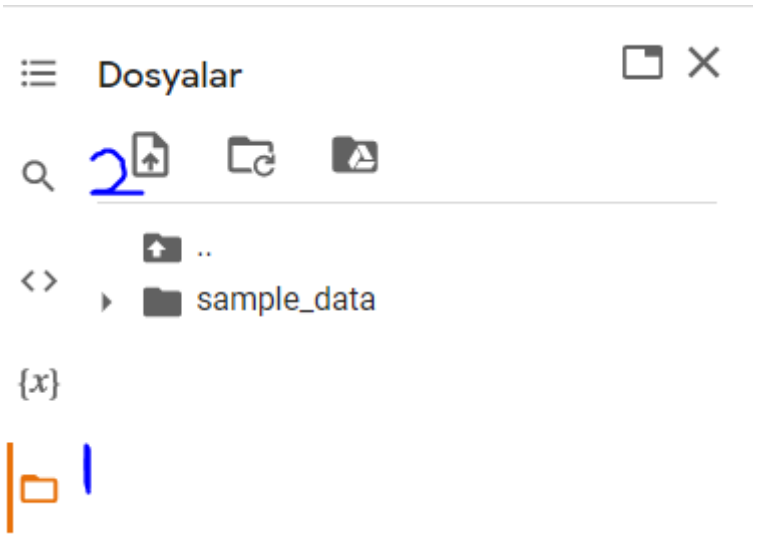


Create a workspace by choosing File > New notebook.

2) Uploading Data to Co Lab

https://drive.google.com/drive/folders/1ZMk4hsyf-pFj2p7UIvNhn6LUCSxNwIxU?usp=sharing

Access the folder with the file extension '.pdb'. Download the data reserved for your group in the lesson. After downloading the file, you don't need to try to run it on your computer because you probably don't have a program that can open a .pdb file. There is no need to open the file.



Open our Co Lab screen. Click on the file icon on the left. After the files section opens, click on the icon marked with a number 2 in the image above, and select the data we have just downloaded to our computer from the drive folder.

Alternatively, after opening the files screen, you can drag the data file you downloaded to your computer to the empty space on the files screen. Select OK from the screen that appears.

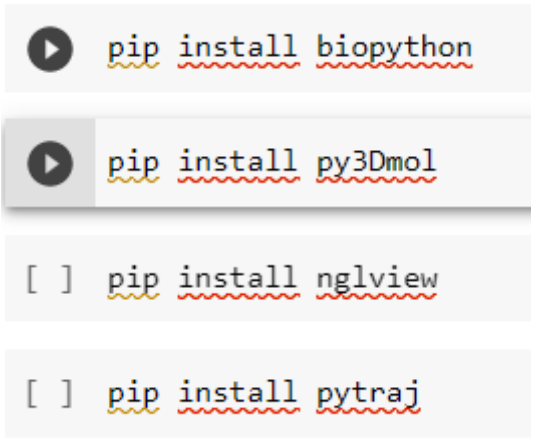
3) Adding a Code Block

Use separate code blocks to perform all operations. Create a CODE block, not text. You can create it by clicking on +Code. If it is not the first line, the +code part will also appear if you move the mouse tip to the center under the previous line.

To run the code blocks, drag the mouse to the brackets next to the block and click on the start icon. Code blocks can be executed more than once. If you click the start icon again, it will run from the beginning.

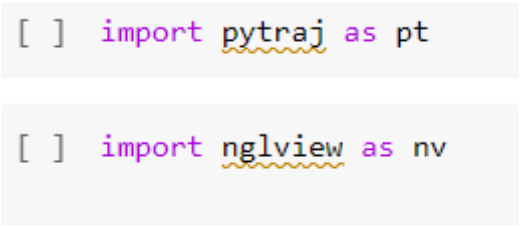
4) Installing Libraries

Libraries are code sources that contain previously written functions. We will use some libraries for protein modeling. You set up each library in separate code blocks and run the code one by one.



5) Importing the Libraries We Downloaded and İncluding Them in the Code

We have downloaded the libraries, but we haven't included them in the code yet, so we can't use the libraries right now. For this, we will import the libraries we will use and include them in the code.



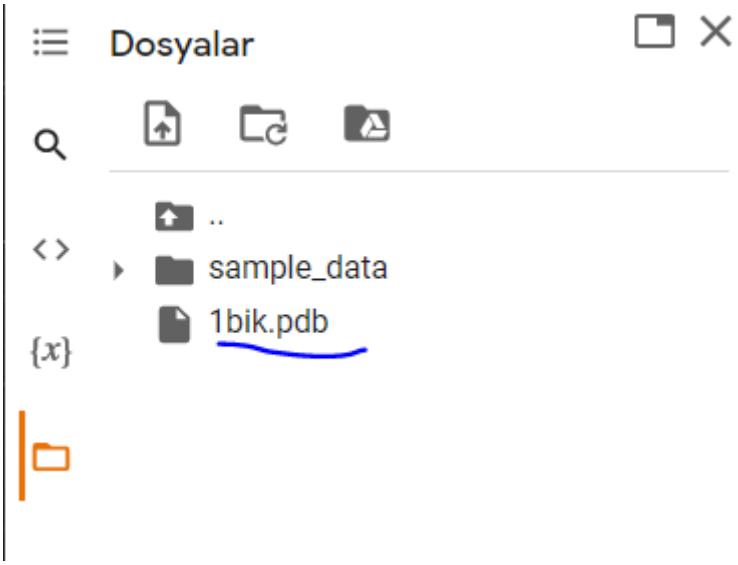
6) Defining the Data File in the Script

Here we assign the file we loaded to Co Lab in step 3 to the variable ncov_traj. The variable name has no meaning, but ncov_traj will be used in the next steps. There is pt.load statement in the code. "pt" is the abbreviation we defined for the pytraj library when importing. "load" is a function for data extraction defined in the pytraj library.

**** Do not run the code with the file path text. I explain how to find the file path below. Add your own file path without deleting the "" sign.

```
[ ]  ncov_traj = pt.load("dosya yolu")
```

7) File Path Discovery



From the Files screen, find the data you uploaded to colab. Right click on the file > select copy path. Now the file path is copied to your clipboard. In step 6, delete the file path text in the code and paste the file path you just copied instead (you can paste it by pressing ctrl+v)

8) Creating the Model

```
view3 = nv.show_pytraj(ncov_traj)
```

'nv' is the abbreviation of the nglview library created during import, the variable to which we assign our ncov_traj data

nv.show_pytraj

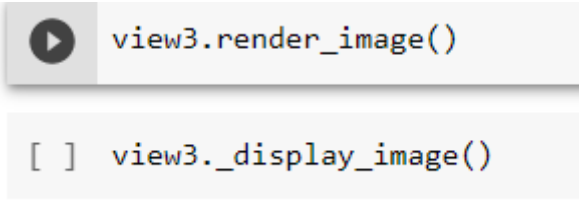
We use the Show_pytraj function which is available in the nglviex library. This function converts the dataset into a protein model and assigns it to a variable called View 3.

9) Model

```
[ ]  view3
```

You can examine the model in detail with the code.

10) Rendering and Visualizing the Model



We created the model and assigned it to the view3 variable. With the render_image command, we convert the model from analyzed output to an image file .png

With the display command, we call the image we have rendered, that is, we print it on the screen.

***It may not work. Sometimes Co Lab is overloaded, try again, you have tried 3 or 5 times, it does not work, you can ask each other or ask me.

*** You can access the algorithm scheme from the link. There may be some errors in the algorithm scheme based on what I wrote in the file

https://drive.google.com/file/d/1Nu9dWPBYgX9qbI85q2X-mqacLzUfprwh/view?usp=sharing

Computational Thinking Examples

DECOMPOSITION

Environmental pollution has become an increasing problem that threatens the lives of living things. Although many of these problems are human-induced, they are increasing day by day. Pesticide and fertilizer applications applied in agricultural production in order to achieve high productivity also pose serious problems.

What was the most challenging for me at this stage? The most challenging thing for me at this stage was to find the problem and the sub-topic to work on because there are so many problems and so many sub-topics for these problems. It was really difficult to decide.

I followed the following steps throughout the process: Researching on the internet, evaluating the information received, and choosing the best one among them.

How would I do it better if I did it again, or what would I change if I wanted to change it? If I did it again, I would probably research other topics more and try to come up with a better idea than before.

PATTERN RECOGNITION:

Due to the differences in the amounts, types, and application times of fertilizers and pesticides applied and the lack of knowledge in this field, living health and the environment are negatively affected. Problems such as salinization, heavy metal accumulation, nutrient imbalance, disruption of microorganism activity, eutrophication and nitrate accumulation in water, nitrogen and sulfur-containing gasses in the air, greenhouse effect, etc. occur due to wrong applications.

What challenged me the most at this stage? The most challenging part of this phase was to find the data that has been recorded so far on the topic, because it was difficult to find sources that gave proper explanations on the topic. I followed the following steps throughout the process: Researching, comparing the accuracy of the information with other sources, weeding out unnecessary information and transcribing it.

How would I do it better if I did it again, or what would I change if I wanted to change it? I would not change anything if I did it again because I think I did the best I could.

ABSTRACTION

To solve this problem, people need to be made aware and educated about pesticides. Many people use faulty pesticides and fertilizers without knowing the damage to the environment and their own production. Therefore, education is the first step. However, there are also people who do not understand the seriousness of the situation even though they have knowledge. These people need to be made aware and penalized when necessary.

What challenged me the most at this stage? The most difficult thing for me at this stage was deciding which solution to focus on because it was difficult to choose between the solutions. I think all our remedies were good. I followed the following steps throughout the process: Thinking of ideas, making choices, and writing them down.

If I were to do it again, how would I do it better, or if I wanted to change it, what would I change? If I were to do it again, I would try other solutions.

ALGORITHM DESIGN

In order to realize these practices, first of all, inspections should be increased. Along with the inspections, people who use faulty fertilizers and pesticides should be identified and training should be given seriously. After the training, controls should not be interrupted, and warnings should be given. When necessary, assistance such as money, seeds and medicines should be provided to protect the farmer. If mistakes continue, deterrent penalties should be applied.

What challenged me the most at this stage? The most difficult thing for me at this stage was to put things in order. I followed the following steps throughout the process: Identifying events, prioritizing them, and creating the algorithm.

How would I do it better if I did it again, or what would I change if I wanted to change it? If I wanted to change it, I think I would add a few more steps to the algorithm.

∝ Biyopython → BLAST
FASTA
GenBank
NCBI - expasy

∝ python operatörleri:
int → sayı (10)
float → ondalıklı sayı (10.0)
str → karakter ("10")

string metotları → replace
count (alt dizi)
strip
lower
upper
append.

UYGULAMA-1

Colab → kullanılacak ifadeler dosyası
↓
control-C
↓
Colab'a yapıştır. (Ctrl-V ile)
↓
print işlemi yapılır.

print ( )
↓
protein dizisi
↓
RUN
↓
yeni bir kod ekle (+KOD)

altgr ve 3 ile # işareti oluştur.
#sekans uzunluğu bulma (len) fonksiyonu
enter.

print ( )
print (len( ))
print (len(protein_dizisi))

— 86 —

SAYFA 1

1. # (sharp işareti) sekansta bulunan aminoasitleri sayma işlemi
2. print (protein_dizisi.count("M"))
3. enter
4. print (protein_dizisi.count("L"))
5. enter
6. print (protein_dizisi.count("P"))
7. run
8. 3 → metionin
   15 → lösin
   7 → prolin
   ] sayıları
9. #bütün sekansı büyük küçük harf olarak değiştirme
   print (protein_dizisi.lower())
   print (protein_dizisi.upper())
   çıktılar.
10. #indekse göre uygulamalar
    print (protein_dizisi[0])
    print (protein_dizisi[1])
    print (protein_dizisi[-1])
    print (protein_dizisi[25])
11. RUN
    V
    L
    A
    A
12. #istenilen index aralığını görme
    print (protein_dizisi[:10])
    (0'dan 10'a kadar yazdır. 10 dahil değil.)
    print (protein_dizisi[10:])
    (10 dahil ve kopsi)
    print (protein_dizisi[10:20])
    (10 - 19)
13. run => çıktıları

DNA dizisi:

① #dizinin uzunluğu
```
dna = ------
print (len(dna))
    60.
print ("DNA dizinin uzunluğu:", len(dna))
```

② +kod ile
#nükleotid sayma
```
print ("Adenin sayısı:", dna.count("A"))
```
run→19
T, C, G için satırı copy-past yap.

③ #DNA'yı RNA'ya çevirme
```
print ("RNA:", dna.replace("T", "U"))
```

④ #GC oranı hesaplama
```
GC-oran = ((dna.count("G")+
           dna.count("C"))/
            len(dna))
print ("GC oran:", GC-oran)
```
0.46

⑤ #motif sayma
```
motif = "ATG"
if motif in dna:
    print ("Girilen motiften dnada"
           dna.count(motif), "tane vardır.")
else:
    print ("Girilen motif bulunmaıadı.")
```
run→1 tane vardır.

#reverse complement
```
print(dna)
reverse-complement = dna[::-1]
```

```
for nükleotid in reverse-complent
  if nükleotid == "A":
  print ("T", end = " ")
   elif nukleotid == "T":
     print ("A", end=" ")
  elif nukleotid == "G":
     print ("C", end=" ")
  elif nukleotid == "C":
     print ("G", end=" ")
```
RUN (tamamlayıcı DNA)

DNA 3'lü kodonlara ayırma:

```
def dna-uclu-kodon(dna):
    kodon-list = [ ]
    for i in range(0, len(dna), 3):
        kodon-list.append(dna[i:i+3])
    return kodon-list
print (dna-uclu-kodon("DNA dizisini yaz"))
```
run
↳ ['ATG', 'CTA' ----]
↳ 3'erli gruplara ayırdı.