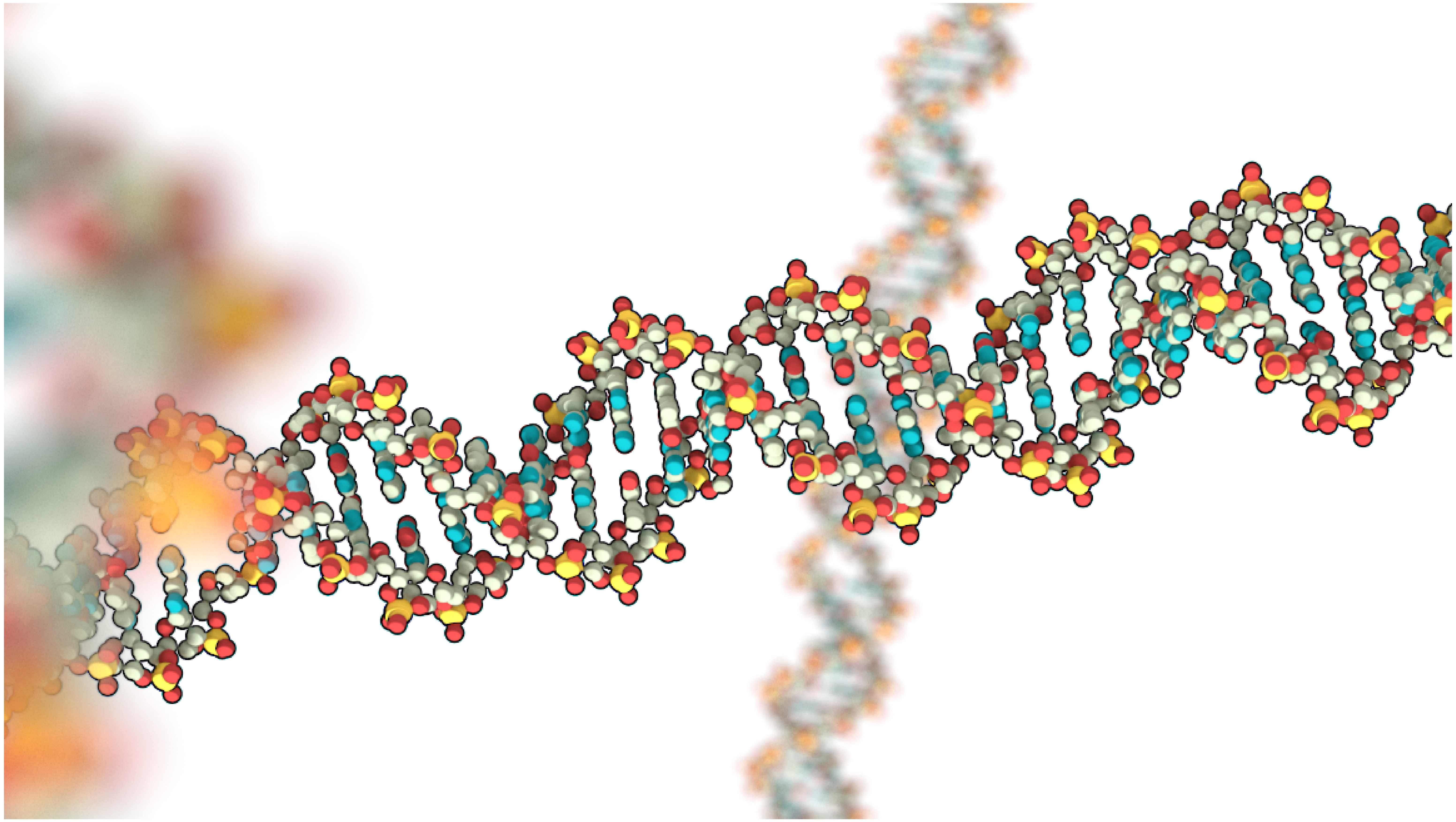
BIOS1100 H17 uke 10

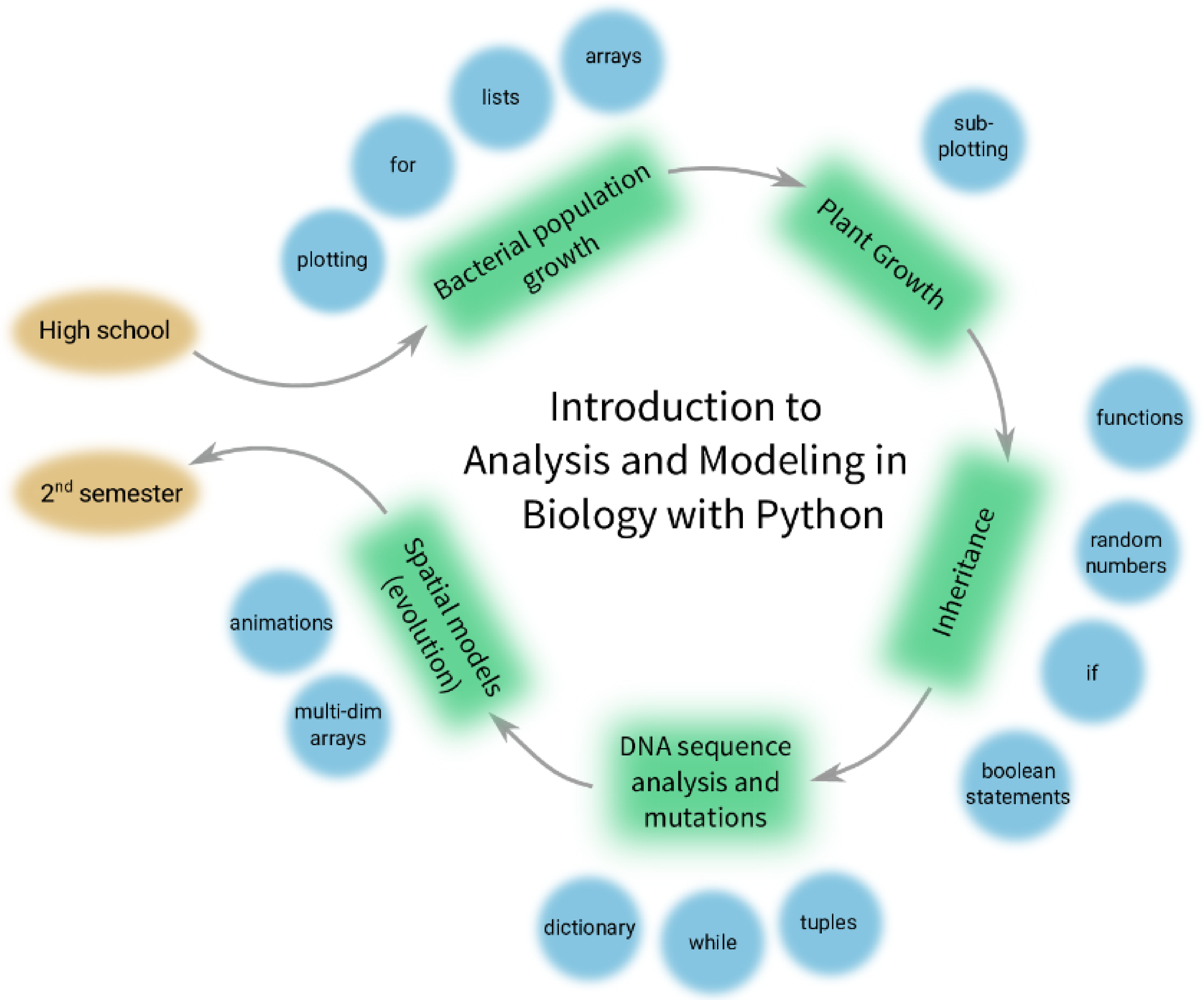
Lex Nederbragt



Ukens forelesning

- nytt stoff denne uken
- utvalgte øvelser

Undervisningsplan



Læringsmål denne uke

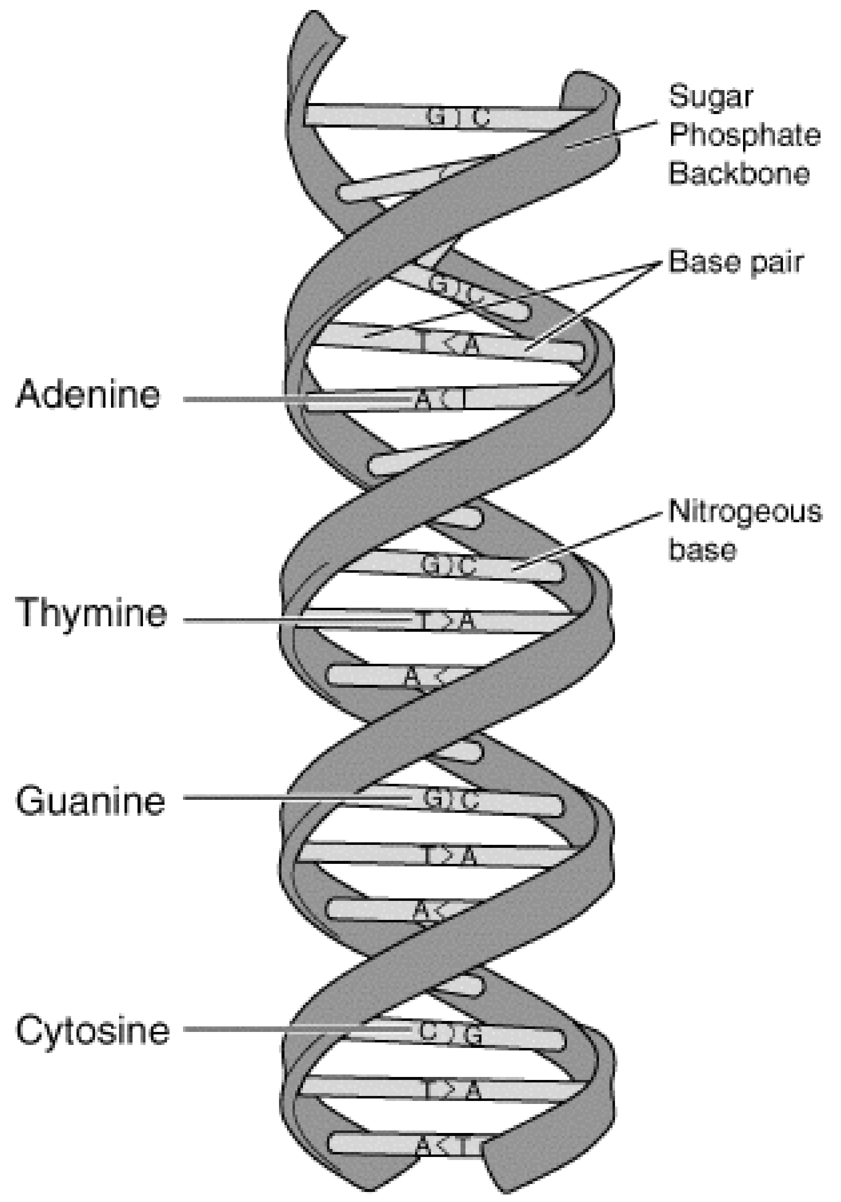
Biologi

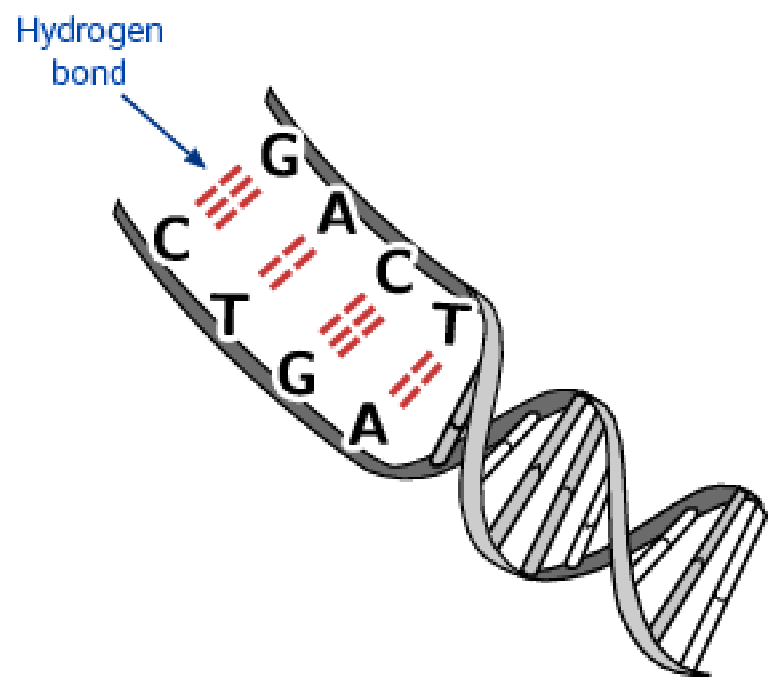
- forstår hvordan DNA kan anses som 'string'

Programmering

- telle nukleotider med `string.count`
- kunne bruke `elif`
- dictionaries ('ordbok')
    - kunne forklare hva det er og hvorfor det brukes
    - hvordan å lage og bruke en dictionary
    - kunne gjøre om to lister til en dictionary

DNA som tekst





Normal hemoglobin subunit

```
ACATTTGCTTCTGACACAACTGTGTTCACTAGCAACCTCAAACAGACACCATGGTGCATCTGACTCCTGA
GGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGTGGATGAAGTTGGTGGTGAGGCCCTGGGC
AGGCTGCTGGTGGTCTACCCTTGGACCCAGAGGTTCTTTGAGTCCTTTGGGGATCTGTCCACTCCTGATG
CTGTTATGGGCAACCCTAAGGTGAAGGCTCATGGCAAGAAAGTGCTCGGTGCCTTTAGTGATGGCCTGGC
TCACCTGGACAACCTCAAGGGCACCTTTGCCACACTGAGTGAGCTGCACTGTGACAAGCTGCACGTGGAT
CCTGAGAACTTCAGGCTCCTGGGCAACGTGCTGGTCTGTGTGCTGGCCCATCACTTTGGCAAAGAATTCA
CCCCACCAGTGCAGGCTGCCTATCAGAAAGTGGTGGCTGGTGTGGCTAATGCCCTGGCCCACAAGTATCA
CTAAGCTCGCTTTCTTGCTGTCCAATTTCTATTAAAGGTTCCTTTGTTCCCTAAGTCCAACTACTAAACT
GGGGGATATTATGAAGGGCCTTGAGCATCTGGATTCTGCCTAATAAAAAACATTTATTTTCATTGC
```

Hva gjør denne koden?

```
e_coli_dna = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAA"
print(e_coli_dna.count("A"))
```

Hva gjør denne koden?

```python
A_count = 0
for nucleotide in e_coli_dna:
    if nucleotide == "A":
        A_count = A_count + 1

print(A_count)
```

Telle nukleotider

```python
e_coli_dna = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAA"
print(e_coli_dna.count("A"))
```

```python
A_count = 0
for nucleotide in e_coli_dna:
    if nucleotide == "A":
        A_count = A_count + 1

print(A_count)
```

Telle nukleotider

```python
e_coli_dna = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAA"
print(e_coli_dna.count("A"))
```

```python
A_count = 0
for nucleotide in e_coli_dna:
    if nucleotide == "A":
        A_count += 1

print(A_count)
```

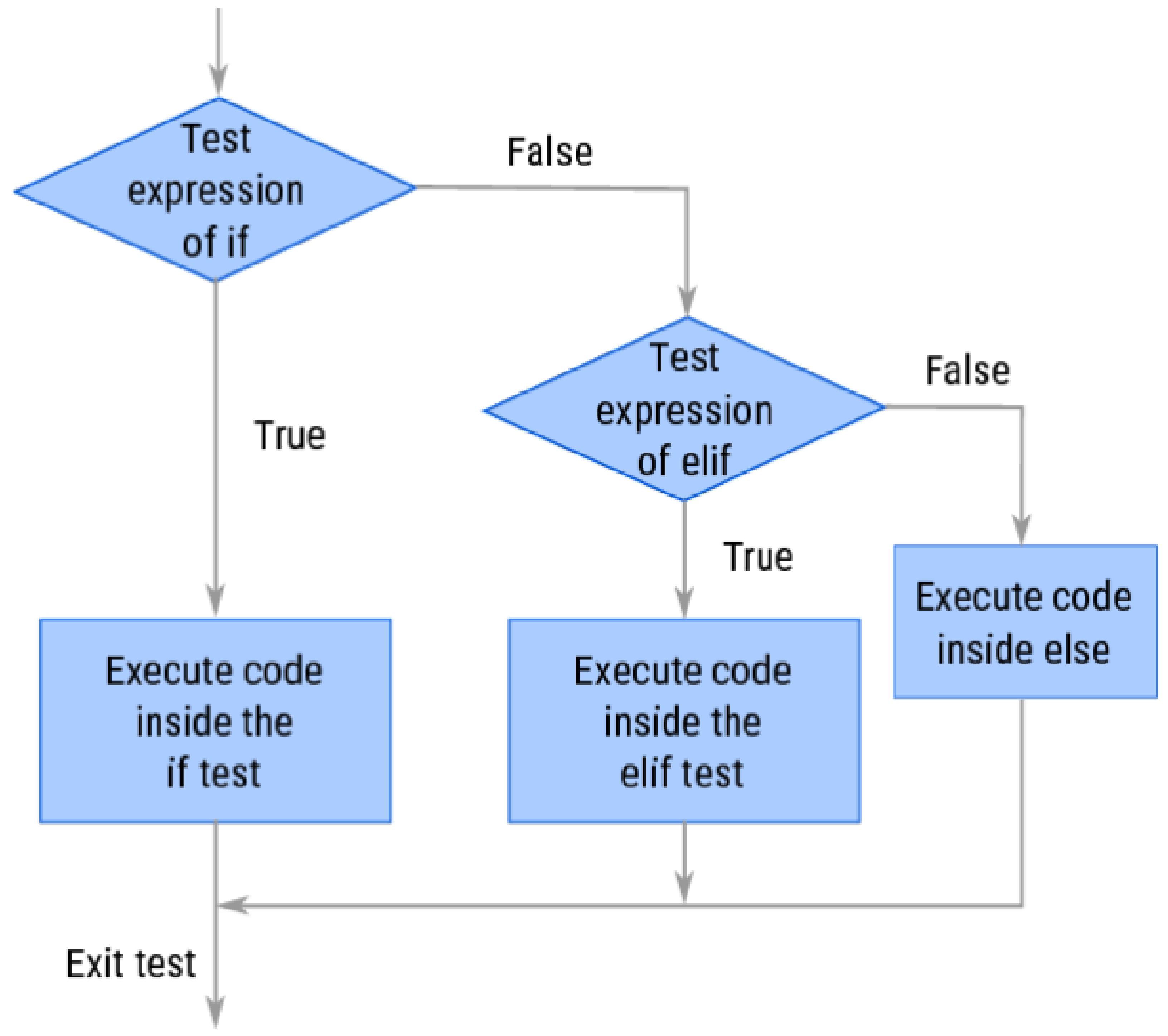
Hva gjør denne koden?

```
A_count = 0
T_count = 0
G_count = 0
C_count = 0

for nucleotide in e_coli_dna:
    if nucleotide == "A":
        A_count += 1
    elif nucleotide == "T":
        T_count += 1
    elif nucleotide == "G":
        G_count += 1
    elif nucleotide == "C":
        C_count += 1
    else:
        print("could not recognize the nucleotide")
```

`if - elif - else`

`if - elif - else` **--> if - else if - else**



Finn feilen

```
nucleotides = ["A", "C", "G", "T"]
nucleotides_counts = [0, 0, 0, 0]

for nucleotide in e_coli_dna:
    if nucleotide == "A":
        nucleotides_counts[0] += 1
    elif nucleotide == "T":
        nucleotides_counts[1] += 1
    elif nucleotide == "G":
        nucleotides_counts[2] += 1
    elif nucleotide == "C":
        nucleotides_counts[3] += 1
    else:
        print("could not recognize the nucleotide")
```

Hva gjør denne koden?

```
nucleotides_count = ______

for nucleotide in e_coli_dna:
    nucleotides_count[nucleotide] += 1
```

Hva gjør denne koden?

```
e_coli_dna = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAA"
```

Inne i løkken:

```
nucleotides_count["A"] += 1
nucleotides_count["G"] += 1
nucleotides_count["C"] += 1
nucleotides_count["T"] += 1
nucleotides_count["T"] += 1
nucleotides_count["T"] += 1
...
```

Dictionary ('ordbok')

Python liste: index er et tall

```python
my_list = ["A", "B", "C", "D"]
print(my_list[2])
```

Dictionary ('ordbok')

Python liste: index er et tall

```
my_list = ["A", "B", "C", "D"]
print(my_list[2])
```

Python dictionary: index kan være en string

```
my_dict = _____
print(my_dict["fish"])
```

Dictionary ('ordbok')

Python tom liste

```
my_list = []
```

Python tom dictionary

```
my_dict = {}
```

Dictionary ('ordbok')

Python liste med innhold

```python
my_list = ["A", "B", "C", "D"]
```

Python dictionary med innhold

```python
my_dict = {"cat": "katt", "dog": "hund", "fish": "fisk"}
```

Dictionary ('ordbok')

Python liste med innhold

```
my_list = ["A",
           "B",
           "C",
           "D"]
```

Python dictionary med innhold

```
my_dict = {"cat": "katt", "dog": "hund", "fish": "fisk"}
```

```
key : value
```

Dictionary ('ordbok')

Python liste med innhold

```
my_list = ["A", "B", "C", "D"]
```

Python dictionary med innhold

```
my_dict = {"cat": "katt",
           "dog": "hund",
           "fish": "fisk"}
```

Dictionary ('ordbok')

Python liste: index er et tall

```
my_list = ["A", "B", "C", "D"]
print(my_list[2])
```

```
C
```

Python dictionary: index kan være en string

```
my_dict = {"cat": "katt", "dog": "hund", "fish": "fisk"}
print(my_dict["fish"])
```

```
fisk
```

Dictionary ('ordbok')

Python liste: legge til et element

```
my_list = ["A", "B", "C", "D"]
my_list.append("E")
```

Python dictionary: legge til et element

```
my_dict = {"cat": "katt", "dog": "hund", "fish": "fisk"}
my_dict["bird"] = "fugl"
```

Dictionary ('ordbok')

Python liste: gå over alle elementer

```python
my_list = ["A", "B", "C", "D"]
for element in my_list:
    # do something with element
```

Dictionary ('ordbok')

Python liste: gå over alle elementer

```
my_list = ["A", "B", "C", "D"]
for element in my_list:
    # do something with element
```

Python dictionary: gå over alle elementer

```
my_dict = {"cat": "katt", "dog": "hund", "fish": "fisk"}
for term in my_dict:
    print(term, "in Norwegian is", my_dict[term])
```

Løkken går over alle **keys**.

Dictionary ('ordbok')

Python dictionary: gå over alle elementer

```python
my_dict = {"cat": "katt", "dog": "hund", "fish": "fisk"}
for term in my_dict:
    print(term, "in Norwegian is", my_dict[term])
```

```python
for key in my_dict:
    print("The key is", key, "and value is", my_dict[key])
```

Hva gjør koden?

```
english = ["cat", "fish", "dog"]
norsk = ["katt", "fisk", "hund"]

my_dict = {}

for index in range(len(english)):
    my_dict[english[index]] = norsk[index]
```

Hva gjør koden?

```python
my_dict = {"cat": "katt", "dog": "hund", "fish": "fisk"}

english = []
norsk = []

for term in my_dict:
    english.append(term)
    norsk.append(my_dict[term])
```

Telle nukleotider

```
e_coli_dna = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAA"

nucleotides_count = {
    "A": 0,
    "T": 0,
    "G": 0,
    "C": 0 }

for nucleotide in nucleotides_count:
    nucleotides_count[nucleotide] = e_coli_dna.count(nucleotide)

print(nucleotides_count)
```

Samme kode?

```
for nucleotide in nucleotides_count:
    nucleotides_count[nucleotide] = e_coli_dna.count(nucleotide)

print(nucleotides_count)
```

```
A_count = 0
T_count = 0
G_count = 0
C_count = 0

for nucleotide in e_coli_dna:
    if nucleotide == "A":
        A_count += 1
    elif nucleotide == "T":
        T_count += 1
    elif nucleotide == "G":
        G_count += 1
    elif nucleotide == "C":
        C_count += 1
    else:
        print("could not recognize the nucleotide")
```

Dictionary

Fra oppgaver uke 09:

```
fragments = {"Empty": fragments_plasmid_empty,
             "PCR 1": fragments_plasmid_PCR_1,
             "PCR 2": fragments_plasmid_PCR_2}
plot_gel_electrophoresis(fragments, lambda_sizes)
show()
```

Dictionary

Values i en Python dictionary kan være:

Dictionary

Values i en Python dictionary kan være:

- int, float, string

Dictionary

Values i en Python dictionary kan være:

- int, float, string
- liste

Dictionary

Values i en Python dictionary kan være:

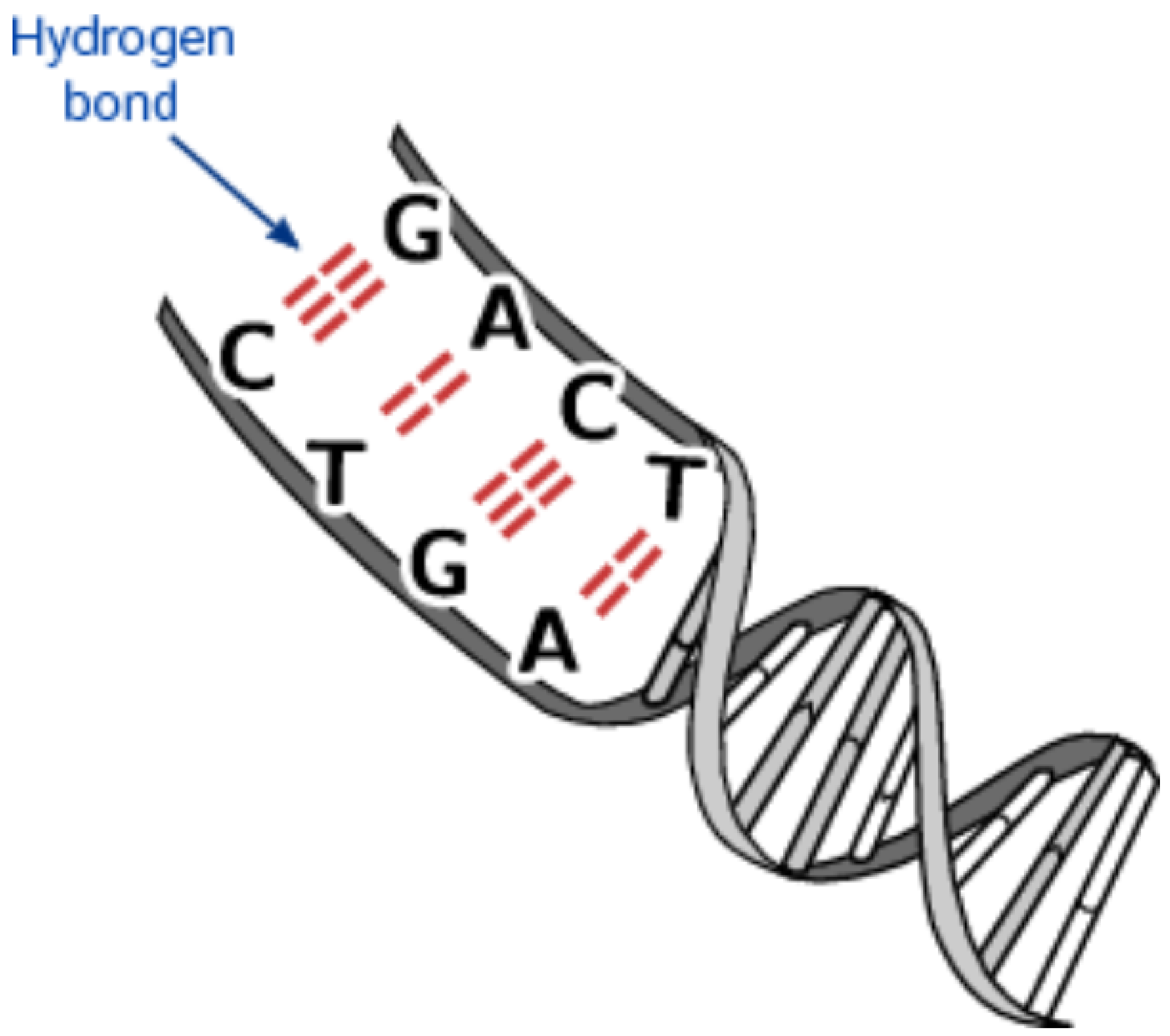
- int, float, string
- liste
- Numpy array

Dictionary

Values i en Python dictionary kan være:

- int, float, string
- liste
- Numpy array
- dictionary

GC percentage



GC percentage

Organism	Adenine	Thymine	Guanine	Cytosine
E. coli	26.0	23.9	24.9	25.2
Yeast	31.7	32.6	18.3	17.4
Turtle	28.7	27.9	22.0	21.3
Salmon	29.7	29.1	20.8	20.4
Chicken	28.0	28.4	22.0	21.6
Human	30.3	30.3	19.5	19.9

GC percentage

Organism	G+C
E. coli	50.1
Yeast	35.7
Turtle	43.3
Salmon	41.2
Chicken	43.6
Human	39.4

Utvalgte øvelser

- Exercise 2: Offline: Study code for digestion of DNA with Python
- Exercise 4: in silico digests for BIOS1110 lab exercises
- Exercise 6: Writing your own modulo function

Slicing a circular DNA string

```
def circular(dna, start_index, stop_index):
    """Slice a circular DNA string."""
    if stop_index <= start_index:
        stop_index += len(dna)

    fragment = ""
    index = start_index
    while index != stop_index:
        circular_index = index % len(dna)
        fragment += dna[circular_index]

        index += 1

    return fragment
```

Restriction digestion

```
def restriction_cutting(dna, restriction_site, cut_index):
    """Perform a restriction cutting of the given DNA strand."""
    # Find the cutting locations
    cutting_locations = []
    for index in range(0, len(dna)):
        region = circular(dna, index, index+len(restriction_site))

        if region == restriction_site:
            cutting_locations.append(index + cut_index)

    # Perform the cutting
    nr_fragments = len(cutting_locations)
    fragments = []

    for index in range(nr_fragments-1):
        fragment = circular(dna, cutting_locations[index],
cutting_locations[index+1])
        fragments.append(fragment)

    # Perform the last cutting
    fragment = circular(dna, cutting_locations[-1], cutting_locations[0])
    fragments.append(fragment)

    return fragments
```

Restriction digestion lineær DNA

```python
def restriction_cutting(dna, restriction_site, cut_index, linear=False):
    """Perform a restriction cutting of the given DNA strand."""
    # Find the cutting locations
    cutting_locations = []
    for index in range(0, len(dna)):
        if linear:
            region = dna[index:index+len(restriction_site)]
        else:
            region = circular(dna, index, index+len(restriction_site))

        if region == restriction_site:
            cutting_locations.append(index + cut_index)
```

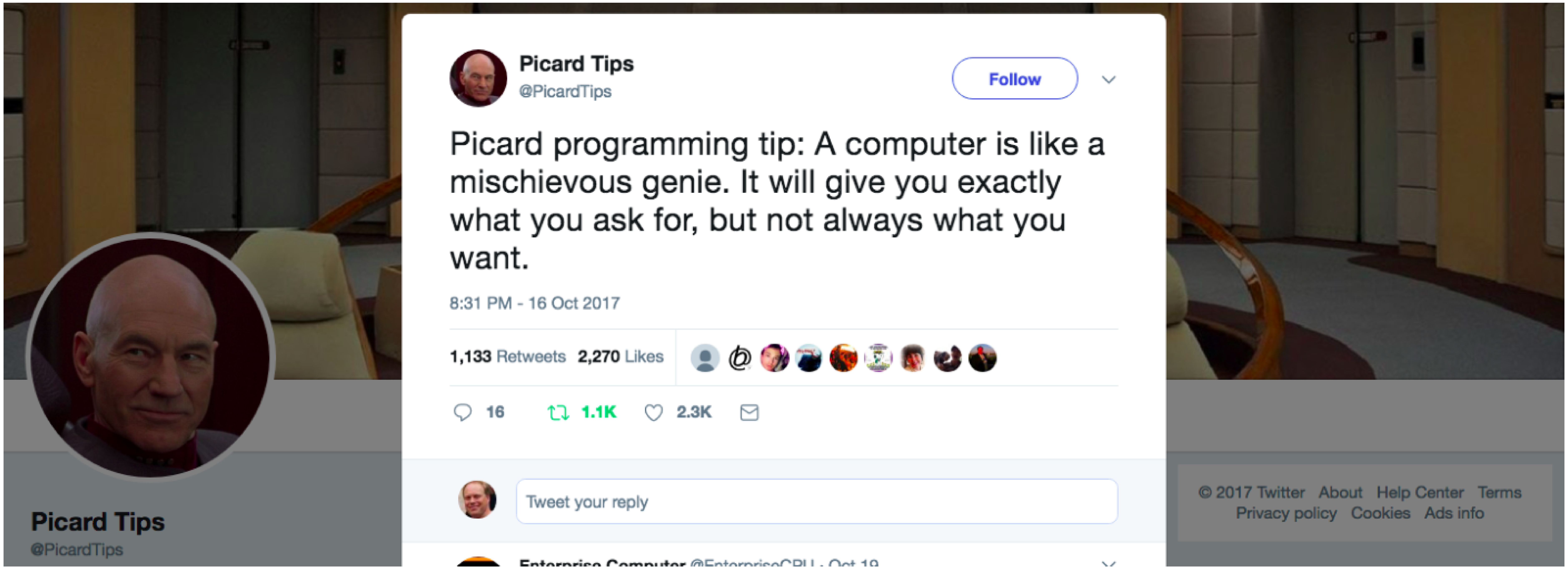
Restriction digestion lineær DNA

```
    # Need to add the first fragment only if linear:
    if linear:
        # Add first fragment
        fragment = dna[0:cutting_locations[0]]
        fragments.append(fragment)

    # For both linear and circular
    for index in range(nr_fragments-1):
        fragment = circular(dna, cutting_locations[index],
cutting_locations[index+1])
        fragments.append(fragment)

    # Perform the last cutting
    if linear:
        fragment = dna[cutting_locations[-1]:]
    else:
        fragment = circular(dna, cutting_locations[-1], cutting_locations[0])
    fragments.append(fragment)

    return fragments
```



Picard Tips
@PicardTips

Follow

Picard programming tip: A computer is like a mischievous genie. It will give you exactly what you ask for, but not always what you want.

8:31 PM - 16 Oct 2017

1,133 Retweets 2,270 Likes

16 1.1K 2.3K

Tweet your reply

© 2017 Twitter About Help Center Terms Privacy policy Cookies Ads info

Picard Tips
@PicardTips

Enterprise Computer @EnterpriseCPU · Oct 19