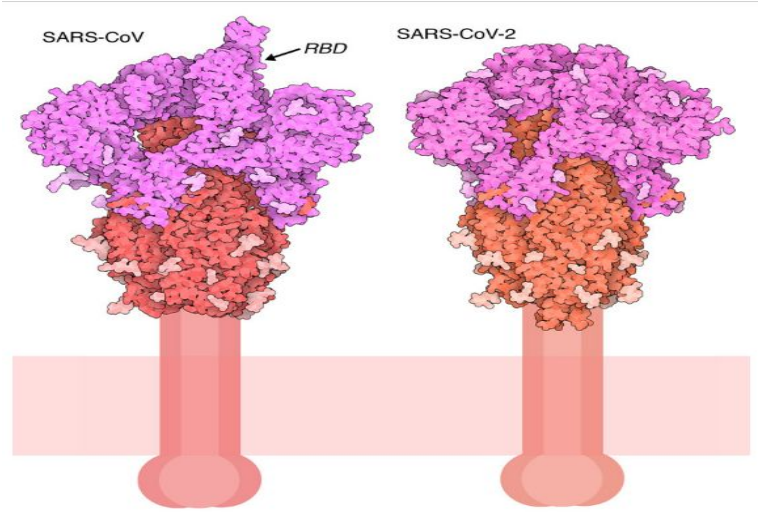
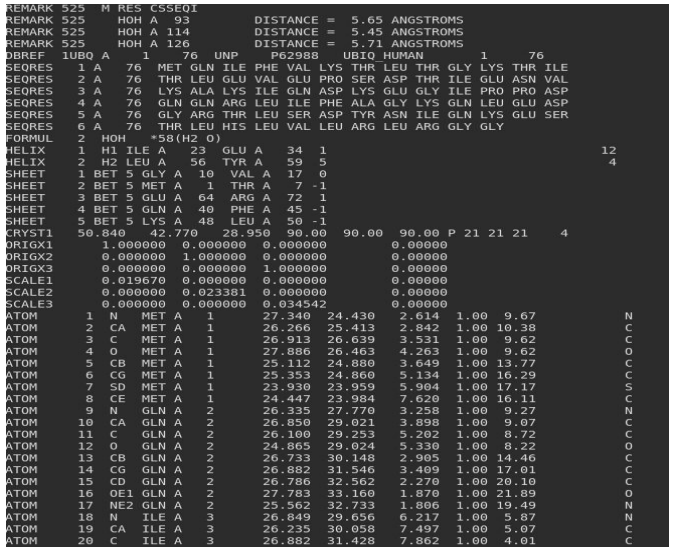
COMP 125 Programming with Python
Modules & Formatted Output



Spike protein from SARS-CoV, with one receptor binding domain (RBD) in the up position, and a closed conformation of the SARS-CoV-2 spike. The S1 fragment is shown in magenta and the S2 fragment in red, with glycosylation in lighter shades.



```
REMARK 525  M RES CSSEQI
REMARK 525    HOH A  93        DISTANCE =  5.65 ANGSTROMS
REMARK 525    HOH A 114        DISTANCE =  5.45 ANGSTROMS
REMARK 525    HOH A 126        DISTANCE =  5.71 ANGSTROMS
DBREF  1UBQ A     1    76  UNP    P62988   UBIQ_HUMAN       1    76
SEQRES   1 A   76  MET GLN ILE PHE VAL LYS THR LEU THR GLY LYS THR ILE
SEQRES   2 A   76  THR LEU GLU VAL GLU PRO SER ASP THR ILE GLU ASN VAL
SEQRES   3 A   76  LYS ALA LYS ILE GLN ASP LYS GLU GLY ILE PRO PRO ASP
SEQRES   4 A   76  GLN GLN ARG LEU ILE PHE ALA GLY LYS GLN LEU GLU ASP
SEQRES   5 A   76  GLY ARG THR LEU SER ASP TYR ASN ILE GLN LYS GLU SER
SEQRES   6 A   76  THR LEU HIS LEU VAL LEU ARG LEU ARG GLY GLY
FORMUL   2  HOH    *58(H2 O)
HELIX    1  H1 ILE A   23  GLU A   34  1                                  12
HELIX    2  H2 LEU A   56  TYR A   59  5                                   4
SHEET    1 BET 5 GLY A  10  VAL A  17  0
SHEET    2 BET 5 MET A   1  THR A   7 -1
SHEET    3 BET 5 GLU A  64  ARG A  72  1
SHEET    4 BET 5 GLN A  40  PHE A  45 -1
SHEET    5 BET 5 LYS A  48  LEU A  50 -1
CRYST1   50.840   42.770   28.950  90.00  90.00  90.00 P 21 21 21    4
ORIGX1      1.000000  0.000000  0.000000        0.00000
ORIGX2      0.000000  1.000000  0.000000        0.00000
ORIGX3      0.000000  0.000000  1.000000        0.00000
SCALE1      0.019670  0.000000  0.000000        0.00000
SCALE2      0.000000  0.023381  0.000000        0.00000
SCALE3      0.000000  0.000000  0.034542        0.00000
ATOM      1  N   MET A   1      27.340  24.430   2.614  1.00  9.67           N
ATOM      2  CA  MET A   1      26.266  25.413   2.842  1.00 10.38           C
ATOM      3  C   MET A   1      26.913  26.639   3.531  1.00  9.62           C
ATOM      4  O   MET A   1      27.886  26.463   4.263  1.00  9.62           O
ATOM      5  CB  MET A   1      25.112  24.880   3.649  1.00 13.77           C
ATOM      6  CG  MET A   1      25.353  24.860   5.134  1.00 16.29           C
ATOM      7  SD  MET A   1      23.930  23.959   5.904  1.00 17.17           S
ATOM      8  CE  MET A   1      24.447  23.984   7.620  1.00 16.11           C
ATOM      9  N   GLN A   2      26.335  27.770   3.258  1.00  9.27           N
ATOM     10  CA  GLN A   2      26.850  29.021   3.898  1.00  9.07           C
ATOM     11  C   GLN A   2      26.100  29.253   5.202  1.00  8.72           C
ATOM     12  O   GLN A   2      24.865  29.024   5.330  1.00  8.22           O
ATOM     13  CB  GLN A   2      26.733  30.148   2.905  1.00 14.46           C
ATOM     14  CG  GLN A   2      26.882  31.546   3.409  1.00 17.01           C
ATOM     15  CD  GLN A   2      26.786  32.562   2.270  1.00 20.10           C
ATOM     16  OE1 GLN A   2      27.783  33.160   1.870  1.00 21.89           O
ATOM     17  NE2 GLN A   2      25.562  32.733   1.806  1.00 19.49           N
ATOM     18  N   ILE A   3      26.849  29.656   6.217  1.00  5.87           N
ATOM     19  CA  ILE A   3      26.235  30.058   7.497  1.00  5.07           C
ATOM     20  C   ILE A   3      26.882  31.428   7.862  1.00  4.01           C
```


Mehmet Sayar

Koç University

Midterm 3

Sunday Dec. 6

9:30-11:30

Homework 3

Dec 8 by midnight

Importing Modules

- Modules: Files that store existing code
  - Help organize functions not built into the interpreter
  - The standard library modules come by default with Python

- To call a function stored in a module, need to write an import statement
  - Usually written at the top of the program (remember from karel?)
  - Format: `import module_name`

Example: Math and Random Modules

- `math` module: part of the standard library containing functions that are useful for performing mathematical calculations

```
import math
```

- `random` module: part of the standard library containing functions for generating and working with various types of random numbers

```
import random
```

Importing Parts of a Module

- Importing a module, copies over everything to your current file!
- What if you only want a small subset (e.g. only trigonometric functions from math, only randint from random etc.)
- A new keyword: `from`

```
from math import sin, cos, tan
print(sin(3.14159/2)) # math.sin when math is imported
print(cos(3.14159/3))
print(tan(3.14159))
```

What if you want to rename the functions you import?

- If you only import specific functions (or classes), their nmaes might overlap with the names of other things in other modules
- You may want to rename them. We use the `as` keyword along with `from`

```
from math import sin as ma_sin
from math import cos as ma_cos
from math import tan as ma_tan
print(ma_sin(3.14159/2))
print(ma_cos(3.14159/3))
print(ma_tan(3.14159))
```

What if you want to rename the module you import?

- Very similar ideas and very common:

```
import numpy as np
print(np.sin(3.14159/2))
print(np.cos(3.14159/3))
print(np.tan(3.14159))
```

- Note: We have done more examples for the things that are on this slide deck with Spyder so watch the video numbered 16

How to create your own modules?

Let’s create a Python file with the following function:

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Dec  2 23:33:49 2020

@author: msayar
"""

def myfunc():
    print("Hello COMP125 !")

```

Save it under
C:\\Users\\msayar\hello.py

Sys module

Where to find the modules?

```
In [1]: import sys

In [2]: sys.path
Out[2]:
['C:\\ProgramData\\Anaconda3\\python38.zip',
 'C:\\ProgramData\\Anaconda3\\DLLs',
 'C:\\ProgramData\\Anaconda3\\lib',
 'C:\\ProgramData\\Anaconda3',
 '',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\win32',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\win32\\lib',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\Pythonwin',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\IPython\
\extensions',
 'C:\\Users\\msayar\\.ipython']
```

Append a new directory

```
In [3]: sys.path.append('C:\\Users\\msayar')

In [4]: sys.path
Out[4]:
['C:\\ProgramData\\Anaconda3\\python38.zip',
 'C:\\ProgramData\\Anaconda3\\DLLs',
 'C:\\ProgramData\\Anaconda3\\lib',
 'C:\\ProgramData\\Anaconda3',
 '',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\win32',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\win32\\lib',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\Pythonwin',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\IPython\
\extensions',
 'C:\\Users\\msayar\\.ipython',
 'C:\\Users\\msayar']
```

Dir function

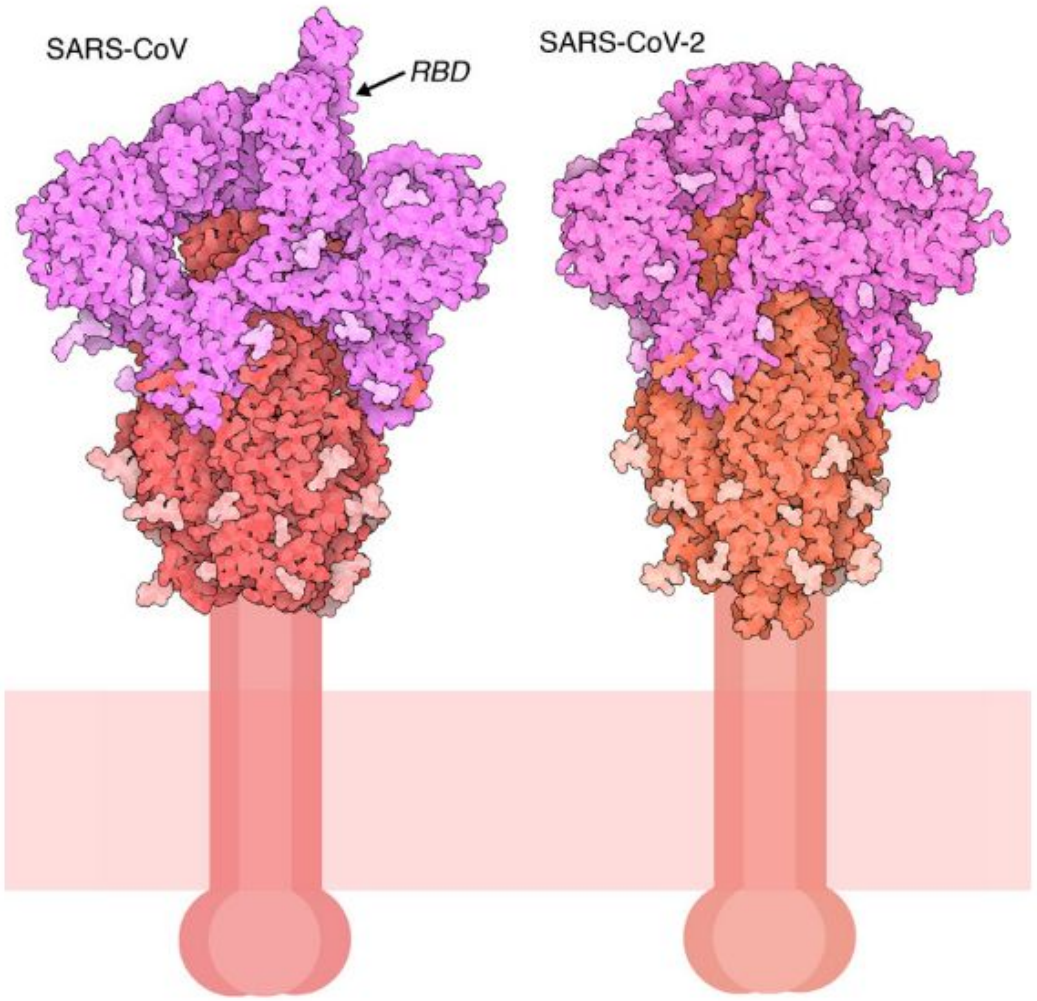
Current list of variables, functions, modules

```
In [10]: dir()
Out[10]:
['In',
 'Out',
 '_',
 '_2',
 '_4',
 '_6',
 '_8',
 '_9',
 '__',
 '___',
 '__builtin__',
 '__builtins__',
 '__doc__',
 '__loader__',
 '__name__',
 '__package__',
```

Let’s import our own module !

```
In [15]: import hello

In [16]: hello.myfunc()
Hello COMP125 !
```

How to store data on the hard disk?

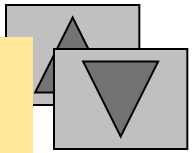


Spike protein from SARS-CoV, with one receptor binding domain (RBD) in the up position, and a closed conformation of the SARS-CoV-2 spike. The S1 fragment is shown in magenta and the S2 fragment in red, with glycosylation in lighter shades.

```
REMARK 525  M RES CSSEQI
REMARK 525    HOH A  93        DISTANCE =  5.65 ANGSTROMS
REMARK 525    HOH A 114        DISTANCE =  5.45 ANGSTROMS
REMARK 525    HOH A 126        DISTANCE =  5.71 ANGSTROMS
DBREF  1UBQ A    1    76  UNP    P62988   UBIQ_HUMAN       1     76
SEQRES   1 A   76  MET GLN ILE PHE VAL LYS THR LEU THR GLY LYS THR ILE
SEQRES   2 A   76  THR LEU GLU VAL GLU PRO SER ASP THR ILE GLU ASN VAL
SEQRES   3 A   76  LYS ALA LYS ILE GLN ASP LYS GLU GLY ILE PRO PRO ASP
SEQRES   4 A   76  GLN GLN ARG LEU ILE PHE ALA GLY LYS GLN LEU GLU ASP
SEQRES   5 A   76  GLY ARG THR LEU SER ASP TYR ASN ILE GLN LYS GLU SER
SEQRES   6 A   76  THR LEU HIS LEU VAL LEU ARG LEU ARG GLY GLY
FORMUL   2  HOH   *58(H2 O)
HELIX    1  H1 ILE A   23  GLU A   34  1                                  12
HELIX    2  H2 LEU A   56  TYR A   59  5                                   4
SHEET    1 BET 5 GLY A  10  VAL A  17  0
SHEET    2 BET 5 MET A   1  THR A   7 -1
SHEET    3 BET 5 GLU A  64  ARG A  72  1
SHEET    4 BET 5 GLN A  40  PHE A  45 -1
SHEET    5 BET 5 LYS A  48  LEU A  50 -1
CRYST1   50.840   42.770   28.950  90.00  90.00  90.00 P 21 21 21    4
ORIGX1      1.000000  0.000000  0.000000        0.00000
ORIGX2      0.000000  1.000000  0.000000        0.00000
ORIGX3      0.000000  0.000000  1.000000        0.00000
SCALE1      0.019670  0.000000  0.000000        0.00000
SCALE2      0.000000  0.023381  0.000000        0.00000
SCALE3      0.000000  0.000000  0.034542        0.00000
ATOM      1  N   MET A   1      27.340  24.430   2.614  1.00  9.67           N
ATOM      2  CA  MET A   1      26.266  25.413   2.842  1.00 10.38           C
ATOM      3  C   MET A   1      26.913  26.639   3.531  1.00  9.62           C
ATOM      4  O   MET A   1      27.886  26.463   4.263  1.00  9.62           O
ATOM      5  CB  MET A   1      25.112  24.880   3.649  1.00 13.77           C
ATOM      6  CG  MET A   1      25.353  24.860   5.134  1.00 16.29           C
ATOM      7  SD  MET A   1      23.930  23.959   5.904  1.00 17.17           S
ATOM      8  CE  MET A   1      24.447  23.984   7.620  1.00 16.11           C
ATOM      9  N   GLN A   2      26.335  27.770   3.258  1.00  9.27           N
ATOM     10  CA  GLN A   2      26.850  29.021   3.898  1.00  9.07           C
ATOM     11  C   GLN A   2      26.100  29.253   5.202  1.00  8.72           C
ATOM     12  O   GLN A   2      24.865  29.024   5.330  1.00  8.22           O
ATOM     13  CB  GLN A   2      26.733  30.148   2.905  1.00 14.46           C
ATOM     14  CG  GLN A   2      26.882  31.546   3.409  1.00 17.01           C
ATOM     15  CD  GLN A   2      26.786  32.562   2.270  1.00 20.10           C
ATOM     16  OE1 GLN A   2      27.783  33.160   1.870  1.00 21.89           O
ATOM     17  NE2 GLN A   2      25.562  32.733   1.806  1.00 19.49           N
ATOM     18  N   ILE A   3      26.849  29.656   6.217  1.00  5.87           N
ATOM     19  CA  ILE A   3      26.235  30.058   7.497  1.00  5.07           C
ATOM     20  C   ILE A   3      26.882  31.428   7.862  1.00  4.01           C
```

2.7 String Formatting

- Strings
  - Unlike other languages strings are a built in data type
    - Allows for easy string manipulation
  - Double quote strings
    - Single quotes need not be escaped
  - Single quote strings
    - Double quotes need not be escaped
  - Triple quoted strings
    - Do not need any escape sequence
    - Used for large blocks of text



Outline

```
1
2    # Printing multiple lines with a single statement.
3
4    print("Welcome\nto\n\nPython!")
```

The **\n** is used to make the text appear on the next line

Fig02_05.py

Program Output

```
Welcome
To

Python!
```

Escape Characters

Escape Sequence	Description
`\n`	Newline. Move the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Move the screen cursor to the beginning of the current line; do not advance to the next line.
`\b`	Backspace. Move the screen cursor back one space.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Print a backslash character.
`\"`	Double quote. Print a double quote character.
`\'`	Single quote. Print a single quote character.

Fig. 2.6 Escape sequences.

Outline

```
# Fig. 2.18: fig02_18.py
# Creating strings and using quote characters in strings.

print("This is a string with \"double quotes.\"")
print('This is another string with "double quotes."')
print('This is a string with \'single quotes.\'')
print("This is another string with 'single quotes.'")
print("""This string has "double quotes" and 'single quotes'.
   You can even do multiple lines.""")
print('''This string also has "double" and 'single' quotes.''')
```

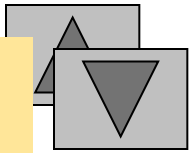
Strings in single quotes need not have double quotes escaped

Strings with double quotes need not escape single quotes

Strings in triple quotes do not have to escape anything and can span many lines

Output

```
This is a string with "double quotes."
This is another string with "double quotes."
This is a string with 'single quotes.'
This is another string with 'single quotes.'
This string has "double quotes" and 'single quotes'.
   You can even do multiple lines.
This string also has "double" and 'single' quotes.
```



Outline

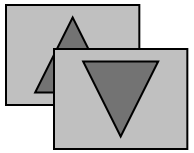
Fig02_19.py

```
# Fig. 2.19: fig02_19.py
# String formatting.

integerValue = 4237
print("Integer ", integerValue)
print ("Decimal integer %d" % integerValue)
print("Hexadecimal integer %x\n" % integerValue)

floatValue = 123456.789
print("Float", floatValue)
print("Default float %f" % floatValue)
print("Default exponential %e\n" % floatValue)

print("Right justify integer (%8d)" % integerValue)
print("Left justify integer  (%-8d)\n" % integerValue)

stringValue = "String formatting"
print("Force eight digits in integer %.8d" % integerValue)
print("Five digits after decimal in float %.5f" % floatValue)
print("Fifteen and five characters allowed in string:")
print("(%.15s) (%.5s)" % ( stringValue, stringValue ))
```

The %e is used to format the string to scientific

Formats the string to contain exactly a specified amount of letters

Formats the string to only allow so many characters



Outline

Fig02_19.py
Program Output

```
Integer  4237
Decimal integer 4237
Hexadecimal integer 108d

Float 123456.789
Default float 123456.789000
Default exponential 1.234568e+005

Right justify integer (    4237)
Left justify \integer  (4237    )

Force eight digits in integer 00004237
Five digits after decimal in float 123456.78900
Fifteen and five characters allowed in string:
(String formatti) (Strin)
```

String Formatting

Conversion Specifier Symbol	Meaning
c	Single character (i.e., a string of length one) or the integer representation of an ASCII character.
s	String or a value to be converted to a string.
d	Signed decimal integer.
u	Unsigned decimal integer.
o	Unsigned octal integer.
x	Unsigned hexadecimal integer (with hexadecimal digits a through f in lowercase letters).
X	Unsigned hexadecimal integer (with hexadecimal digits A through F in uppercase letters).
f	Floating-point number.
e, E	Floating-point number (using scientific notation).
g, G	Floating-point number (using least-significant digits).

Fig. 2.20 String-formatting characters.

Old School Formatting vs An Improved String Formatting Syntax

Option #1: %-formatting

```
In [32]: name="Comp. 125"

In [33]: "Hello %s" % name
Out[33]: 'Hello Comp. 125'
```

If we update “name”

```
In [34]: name="Comp. 125 Sec. 2"

In [35]: "Hello %s" % name
Out[35]: 'Hello Comp. 125 Sec. 2'
```

Option #2: str.format()

```
In [43]: name="Greta"

In [44]: lastname="Thunberg"

In [45]: "Hello, {0}. You are doing a great job {0} {1}.".format(name, lastname)
Out[45]: 'Hello, Greta. You are doing a great job Greta Thunberg.'
```

f-string

```
In [46]: f"Hello, {name}. You are doing a great job {name} {lastname}."
Out[46]: 'Hello, Greta. You are doing a great job Greta Thunberg.'
```

```
In [47]: name="Atlas"

In [48]: lastname="Sarrafoğlu"

In [49]: f"Hello, {name}. You are doing a great job {name} {lastname}."
Out[49]: 'Hello, Atlas. You are doing a great job Atlas Sarrafoğlu.'
```

How do you want to align your text?

```
In [73]: f"{name:<10}"
Out[73]: 'Atlas     '

In [74]: f"{name:>10}"
Out[74]: '     Atlas'

In [75]: f"{name:^10}"
Out[75]: '  Atlas   '
```

F-string: further control

Type declaration not required in f-string !

```
In [58]: import math

In [59]: f"{2 * math.pi}"
Out[59]: '6.283185307179586'

In [60]: f"{2 * math.pi:10.5}"
Out[60]: '    6.2832'
```

Switch to different formats:

```
In [5]: f"{2 * math.pi:e}"
Out[5]: '6.283185e+00'

In [6]: f"{2 * math.pi:g}"
Out[6]: '6.28319'
```

Old style formatting for floating point numbers

```
In [63]: "%f" % (2 * math.pi)
Out[63]: '6.283185'

In [64]: "%10.5f" % (2 * math.pi)
Out[64]: '   6.28319'
```