Python for scientists

Lesson 8
Python scripts and modules

```
def complementary(seq):
    nt_comp = {
        'A' : 'T',
        'C' : 'G',
        'G' : 'C',
        'T' : 'A',
    }
    for nt in seq:
        compseq += nt_comp[nt]
    return compseq
```

Alvaro Sebastián Yagüe

www.sixthresearcher.com

@SixthResearcher

Python scripts

Python scripts

A Python script is a collection of commands in a file designed to be executed like a program.

The script file will contain variables, functions and any Python code, but the idea is that **it will be run or executed from the command line or from within a Python interactive shell to perform a specific task**.

The Python interpreter (command line or shell) was very useful for learning, but as you probably noticed, when you quit from the interpreter and enter it again, the definitions you have made of functions and variables are lost.

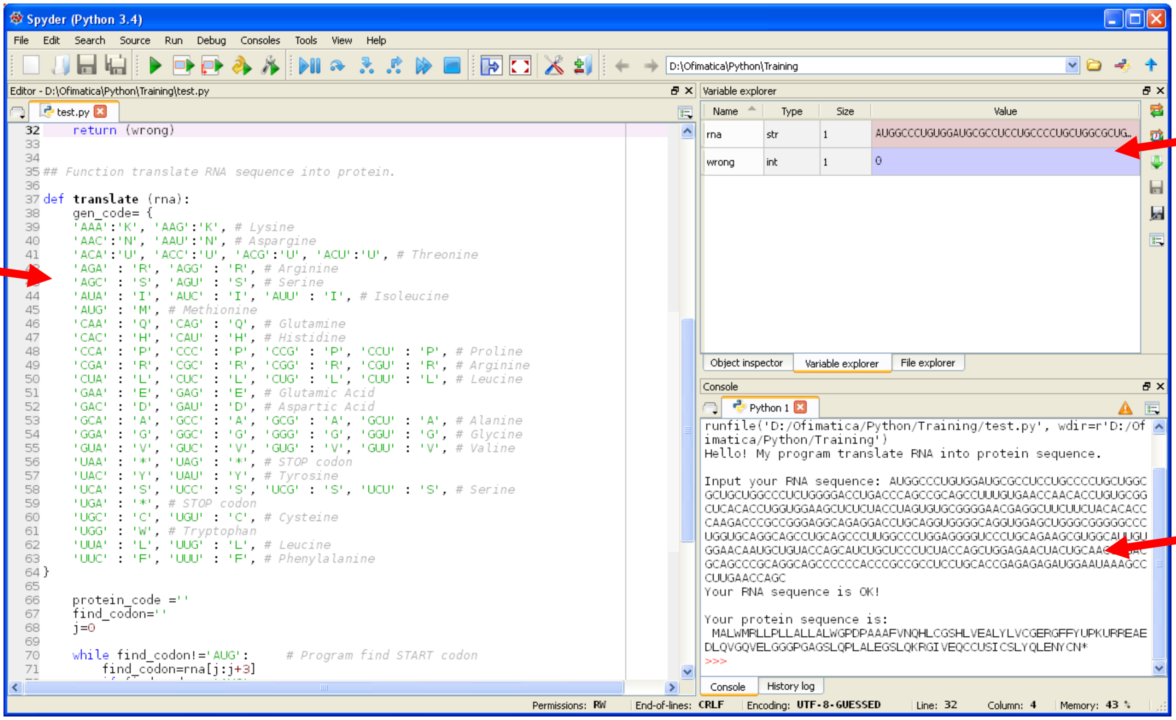
Therefore, **if you want to write a long Python script**, you should use a more sophisticated tool, for example:

- **Spyder: An integrated development environment (IDE).**
- **Notepad++: An advanced text editor.**

Spyder

Spyder (formerly Pydee) includes:

- **An advanced editor** with syntax highlighting and code completion help.
- **A console** to run and test the script.
- **A debugger** to find errors in the code running it line by line and checking the content of the variables.



```
    return (wrong)


## Function translate RNA sequence into protein.

def translate (rna):
    gen_code= {
    'AAA':'K', 'AAG':'K', # Lysine
    'AAC':'N', 'AAU':'N', # Aspargine
    'ACA':'U', 'ACC':'U', 'ACG':'U', 'ACU':'U', # Threonine
    'AGA' : 'R', 'AGG' : 'R', # Arginine
    'AGC' : 'S', 'AGU' : 'S', # Serine
    'AUA' : 'I', 'AUC' : 'I', 'AUU' : 'I', # Isoleucine
    'AUG' : 'M', # Methionine
    'CAA' : 'Q', 'CAG' : 'Q', # Glutamine
    'CAC' : 'H', 'CAU' : 'H', # Histidine
    'CCA' : 'P', 'CCC' : 'P', 'CCG' : 'P', 'CCU' : 'P', # Proline
    'CGA' : 'R', 'CGC' : 'R', 'CGG' : 'R', 'CGU' : 'R', # Arginine
    'CUA' : 'L', 'CUC' : 'L', 'CUG' : 'L', 'CUU' : 'L', # Leucine
    'GAA' : 'E', 'GAG' : 'E', # Glutamic Acid
    'GAC' : 'D', 'GAU' : 'D', # Aspartic Acid
    'GCA' : 'A', 'GCC' : 'A', 'GCG' : 'A', 'GCU' : 'A', # Alanine
    'GGA' : 'G', 'GGC' : 'G', 'GGG' : 'G', 'GGU' : 'G', # Glycine
    'GUA' : 'V', 'GUC' : 'V', 'GUG' : 'V', 'GUU' : 'V', # Valine
    'UAA' : '*', 'UAG' : '*', # STOP codon
    'UAC' : 'Y', 'UAU' : 'Y', # Tyrosine
    'UCA' : 'S', 'UCC' : 'S', 'UCG' : 'S', 'UCU' : 'S', # Serine
    'UGA' : '*', # STOP codon
    'UGC' : 'C', 'UGU' : 'C', # Cysteine
    'UGG' : 'W', # Tryptophan
    'UUA' : 'L', 'UUG' : 'L', # Leucine
    'UUC' : 'F', 'UUU' : 'F', # Phenylalanine
}

    protein_code =''
    find_codon=''
    j=0

    while find_codon!='AUG':      # Program find START codon
        find_codon=rna[j:j+3]
```

Name	Type	Size	Value
rna	str	1	AUGGCCCUGUGGAUGCGCCUCCUGCCCCUGCUGGCGCUG..
wrong	int	1	0

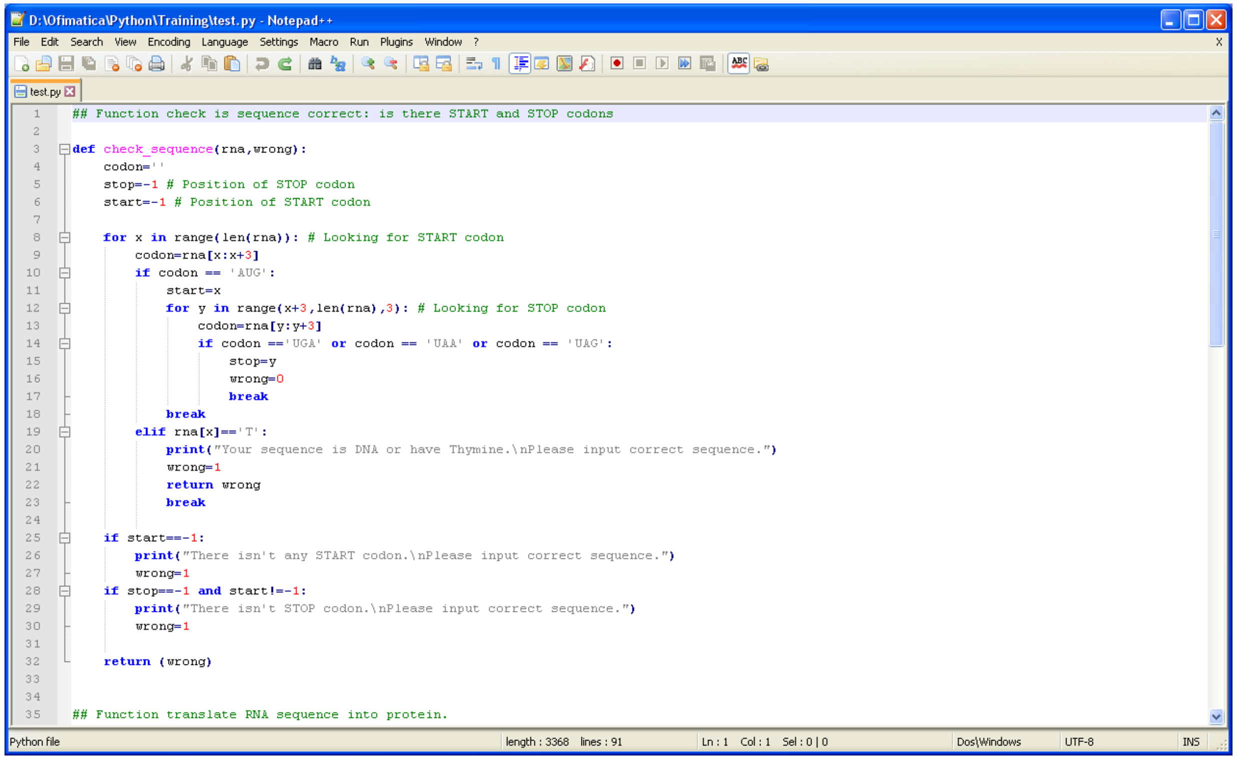
```
runfile('D:/Ofimatica/Python/Training/test.py', wdir=r'D:/Of
imatica/Python/Training')
Hello! My program translate RNA into protein sequence.

Input your RNA sequence: AUGGCCCUGUGGAUGCGCCUCCUGCCCCUGCUGGC
GCUGCUGGCCCUCUGGGGACCUGACCCAGCCGCAGCCUUUGUGAACCAACACCUGUGCGG
CUCACACCUGGUGGAAGCUCUCUACCUAGUGUGCGGGGAACGAGGCUUCUUCUACACACC
CAAGACCCGCCGGGAGGCAGAGGACCUGCAGGUGGGGCAGGUGGAGCUGGGCGGGGGCCC
UGGUGCAGGCAGCCUGCAGCCCUUGGCCCUGGAGGGGUCCCUGCAGAAGCGUGGCAUUGU
GGAACAAUGCUGUACCAGCAUCUGCUCCCUCUACCAGCUGGAGAACUACUGCAACUAGAC
GCAGCCCGCAGGCAGCCCCCCACCCGCCGCCUCCUGCACCGAGAGAGAUGGAAUAAAGCC
CUUGAACCAGC
Your RNA sequence is OK!

Your protein sequence is:
 MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYUPKURREAE
DLQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCUSICSLYQLENYCN*
>>>
```

Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8-GUESSED   Line: 32   Column: 4   Memory: 43 %

Notepad++

Notepad++ is less powerful than Spyder for Python programming, its main advantages are syntax highlighting and code auto-completion.

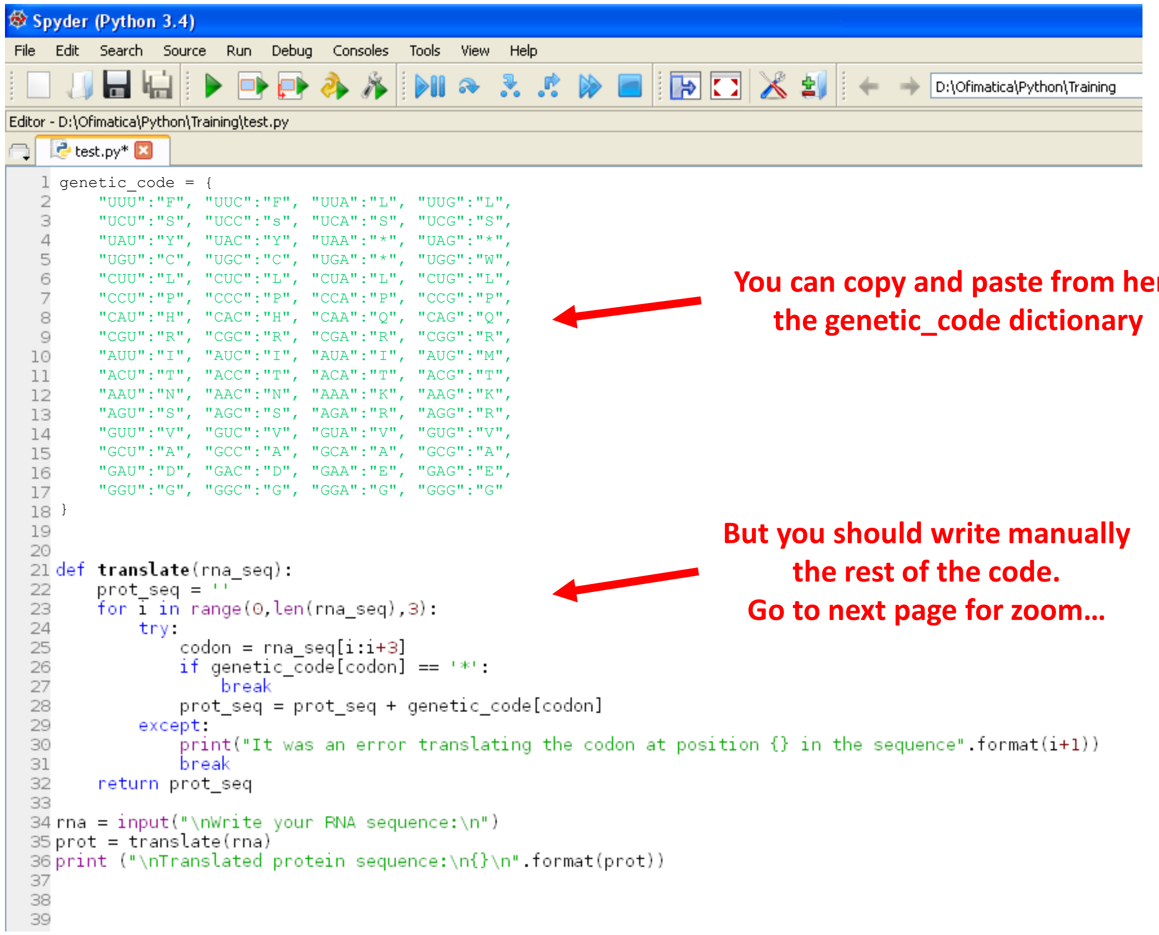
But Notepad++ can be used to programing in many languages, not only Python, and it can be more intuitive and user-friendly for some people.


D:\Ofimatica\Python\Training\test.py - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
test.py

```python
## Function check is sequence correct: is there START and STOP codons

def check_sequence(rna,wrong):
    codon=''
    stop=-1 # Position of STOP codon
    start=-1 # Position of START codon

    for x in range(len(rna)): # Looking for START codon
        codon=rna[x:x+3]
        if codon == 'AUG':
            start=x
            for y in range(x+3,len(rna),3): # Looking for STOP codon
                codon=rna[y:y+3]
                if codon =='UGA' or codon == 'UAA' or codon == 'UAG':
                    stop=y
                    wrong=0
                    break
            break
        elif rna[x]=='T':
            print("Your sequence is DNA or have Thymine.\nPlease input correct sequence.")
            wrong=1
            return wrong
            break

    if start==-1:
        print("There isn't any START codon.\nPlease input correct sequence.")
        wrong=1
    if stop==-1 and start!=-1:
        print("There isn't STOP codon.\nPlease input correct sequence.")
        wrong=1

    return (wrong)


## Function translate RNA sequence into protein.
```

Creating a script in Spyder

Let’s create in Spyder a script that asks for a RNA sequence and translates it into a protein:


Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
D:\Ofimatica\Python\Training
Editor - D:\Ofimatica\Python\Training\test.py
test.py*

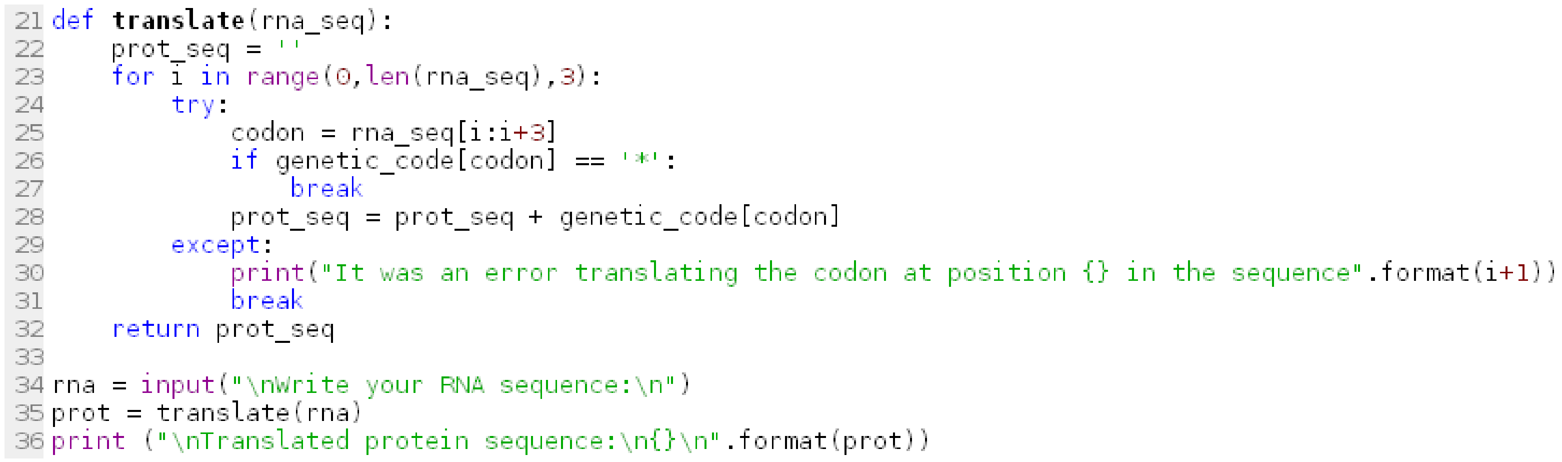
```python
genetic_code = {
    "UUU":"F", "UUC":"F", "UUA":"L", "UUG":"L",
    "UCU":"S", "UCC":"s", "UCA":"S", "UCG":"S",
    "UAU":"Y", "UAC":"Y", "UAA":"*", "UAG":"*",
    "UGU":"C", "UGC":"C", "UGA":"*", "UGG":"W",
    "CUU":"L", "CUC":"L", "CUA":"L", "CUG":"L",
    "CCU":"P", "CCC":"P", "CCA":"P", "CCG":"P",
    "CAU":"H", "CAC":"H", "CAA":"Q", "CAG":"Q",
    "CGU":"R", "CGC":"R", "CGA":"R", "CGG":"R",
    "AUU":"I", "AUC":"I", "AUA":"I", "AUG":"M",
    "ACU":"T", "ACC":"T", "ACA":"T", "ACG":"T",
    "AAU":"N", "AAC":"N", "AAA":"K", "AAG":"K",
    "AGU":"S", "AGC":"S", "AGA":"R", "AGG":"R",
    "GUU":"V", "GUC":"V", "GUA":"V", "GUG":"V",
    "GCU":"A", "GCC":"A", "GCA":"A", "GCG":"A",
    "GAU":"D", "GAC":"D", "GAA":"E", "GAG":"E",
    "GGU":"G", "GGC":"G", "GGA":"G", "GGG":"G"
}


def translate(rna_seq):
    prot_seq = ''
    for i in range(0,len(rna_seq),3):
        try:
            codon = rna_seq[i:i+3]
            if genetic_code[codon] == '*':
                break
            prot_seq = prot_seq + genetic_code[codon]
        except:
            print("It was an error translating the codon at position {} in the sequence".format(i+1))
            break
    return prot_seq

rna = input("\nWrite your RNA sequence:\n")
prot = translate(rna)
print ("\nTranslated protein sequence:\n{}\n".format(prot))
```

You can copy and paste from here the genetic_code dictionary

But you should write manually the rest of the code.
Go to next page for zoom…

Creating a script in Spyder

‘translate()’ function and rest of the code:



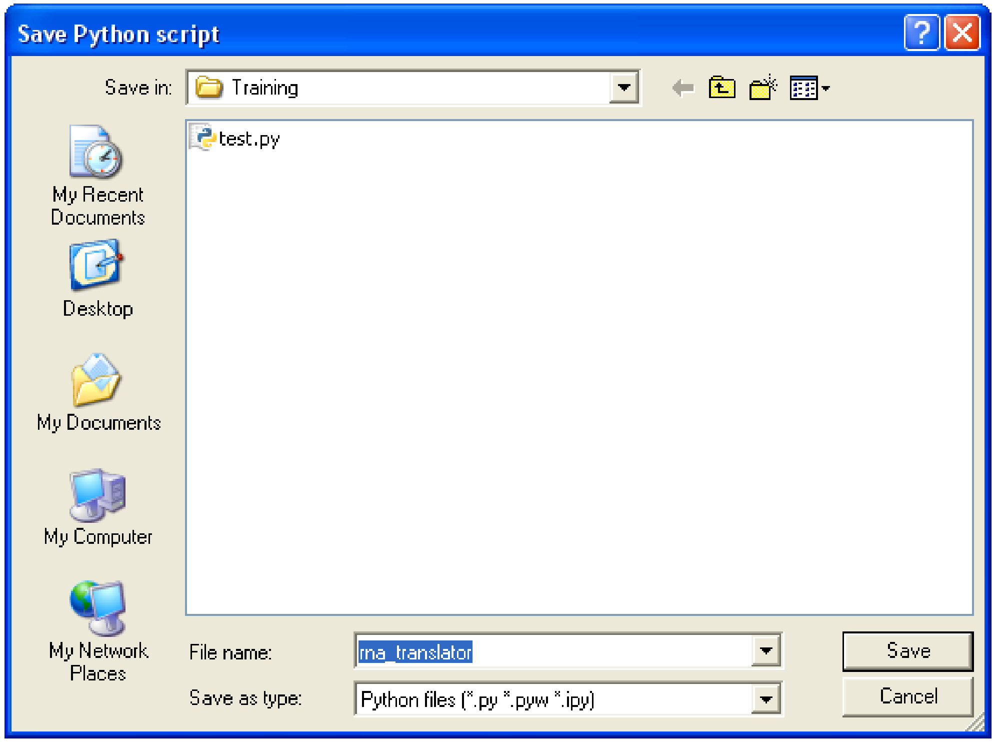
```python
def translate(rna_seq):
    prot_seq = ''
    for i in range(0,len(rna_seq),3):
        try:
            codon = rna_seq[i:i+3]
            if genetic_code[codon] == '*':
                break
            prot_seq = prot_seq + genetic_code[codon]
        except:
            print("It was an error translating the codon at position {} in the sequence".format(i+1))
            break
    return prot_seq

rna = input("\nWrite your RNA sequence:\n")
prot = translate(rna)
print ("\nTranslated protein sequence:\n{}\n".format(prot))
```

RNA sequence to try the script later:

AUGGCCCUGUGGAUGCGCCUCCUGCCCCUGCUGGCGCUGCUGGCCCUCUGGGGACCUGACCCAGCCGCAGCCUUU
GUGAACCAACACCUGUGCGGCUCACACCUGGUGGAAGCUCUCUACCUAGUGUGCGGGGAACGAGGCUUCUUCUAC
ACACCCAAGACCCGCCGGGAGGCAGAGGACCUGCAGGUGGGGCAGGUGGAGCUGGGCGGGGGCCCUGGUGCAGG
CAGCCUGCAGCCCUUGGCCCUGGAGGGGUCCCUGCAGAAGCGUGGCAUUGUGGAACAAUGCUGUACCAGCAUCUG
CUCCCUCUACCAGCUGGAGAACUACUGCAACUAGACGCAGCCCGCAGGCAGCCCCCCACCCGCCGCCUCCUGCACCG
AGAGAGAUGGAAUAAAGCCCUUGAACCAGC

Saving a script in Spyder

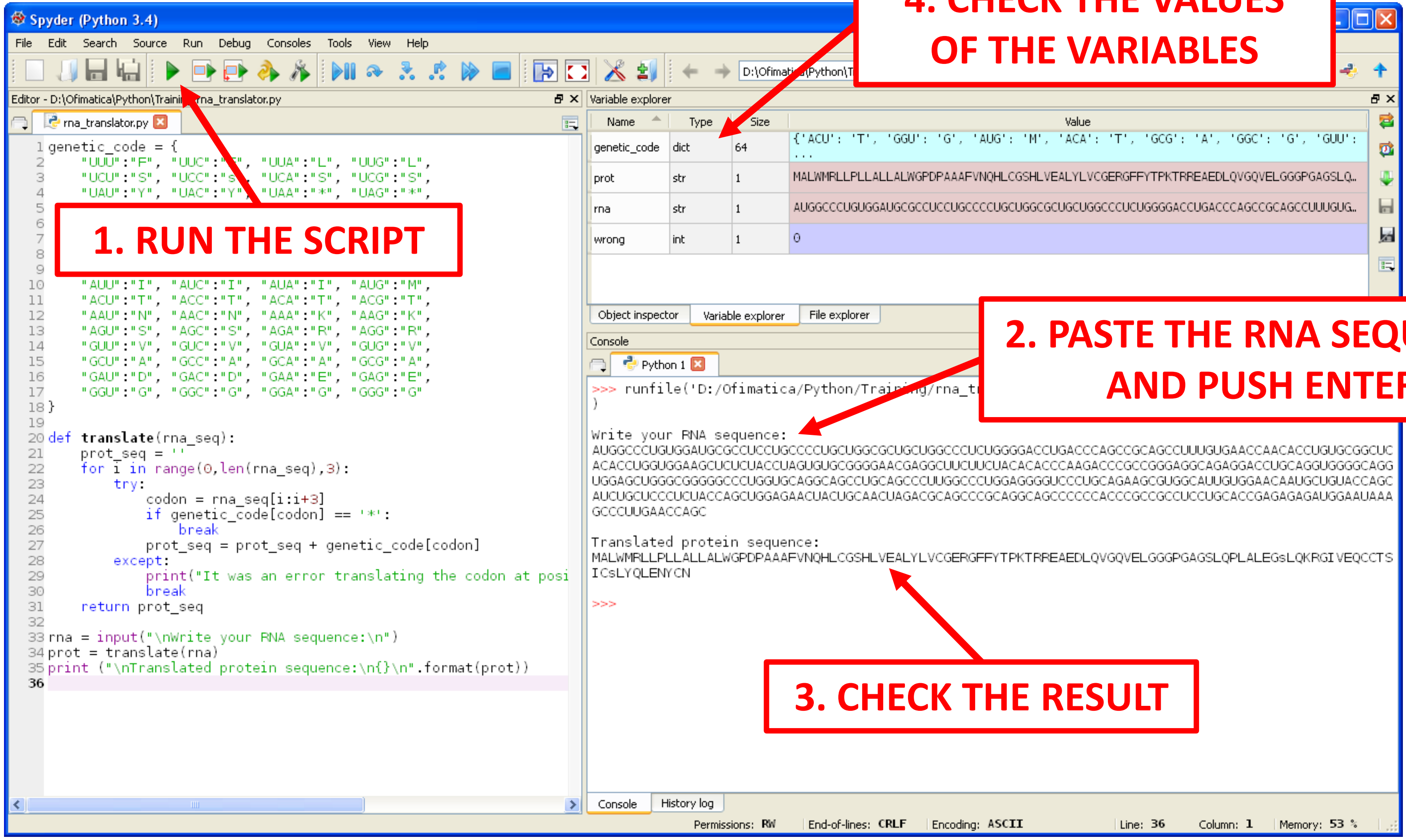
Save the script with the name 'rna_translator'. Spyder will automatically add the **typical Python script extension '.py'**, so the final name of the script will be 'rna_translator.py'.



File name: rna_translator

Save as type: Python files (*.py *.pyw *.ipy)

Running a script in Spyder



4. CHECK THE VALUES OF THE VARIABLES

1. RUN THE SCRIPT

2. PASTE THE RNA SEQUENCE AND PUSH ENTER

3. CHECK THE RESULT

```python
genetic_code = {
    "UUU":"F", "UUC":"F", "UUA":"L", "UUG":"L",
    "UCU":"S", "UCC":"S", "UCA":"S", "UCG":"S",
    "UAU":"Y", "UAC":"Y", "UAA":"*", "UAG":"*",

    "AUU":"I", "AUC":"I", "AUA":"I", "AUG":"M",
    "ACU":"T", "ACC":"T", "ACA":"T", "ACG":"T",
    "AAU":"N", "AAC":"N", "AAA":"K", "AAG":"K",
    "AGU":"S", "AGC":"S", "AGA":"R", "AGG":"R",
    "GUU":"V", "GUC":"V", "GUA":"V", "GUG":"V",
    "GCU":"A", "GCC":"A", "GCA":"A", "GCG":"A",
    "GAU":"D", "GAC":"D", "GAA":"E", "GAG":"E",
    "GGU":"G", "GGC":"G", "GGA":"G", "GGG":"G"
}

def translate(rna_seq):
    prot_seq = ''
    for i in range(0,len(rna_seq),3):
        try:
            codon = rna_seq[i:i+3]
            if genetic_code[codon] == '*':
                break
            prot_seq = prot_seq + genetic_code[codon]
        except:
            print("It was an error translating the codon at posi
            break
    return prot_seq

rna = input("\nWrite your RNA sequence:\n")
prot = translate(rna)
print ("\nTranslated protein sequence:\n{}\n".format(prot))
```

Name	Type	Size	Value
genetic_code	dict	64	{'ACU': 'T', 'GGU': 'G', 'AUG': 'M', 'ACA': 'T', 'GCG': 'A', 'GGC': 'G', 'GUU': ...
prot	str	1	MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVGQVELGGGPGAGSLQ...
rna	str	1	AUGGCCCUGUGGAUGCGCCUCCUGCCCCUGCUGGCGCUGCUGGCCCUCUGGGGACCUGACCCAGCCGCAGCCUUUGUG...
wrong	int	1	0

```
>>> runfile('D:/Ofimatica/Python/Training/rna_t
)

Write your RNA sequence:
AUGGCCCUGUGGAUGCGCCUCCUGCCCCUGCUGGCGCUGCUGGCCCUCUGGGGACCUGACCCAGCCGCAGCCUUUGUGAACCAACACCUGUGCGGCUC
ACACCUGGUGGAAGCUCUCUACCUAGUGUGCGGGGAACGAGGCUUCUUCUACACACCCAAGACCCGCCGGGAGGCAGAGGACCUGCAGGUGGGGCAGG
UGGAGCUGGGCGGGGGCCCUGGUGCAGGCAGCCUGCAGCCCUUGGCCCUGGAGGGGUCCCUGCAGAAGCGUGGCAUUGUGGAACAAUGCUGUACCAGC
AUCUGCUCCCUCUACCAGCUGGAGAACUACUGCAACUAGACGCAGCCCGCAGGCAGCCCCCCACCCGCCGCCUCCUGCACCGAGAGAGAUGGAAUAAA
GCCCUUGAACCAGC

Translated protein sequence:
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVGQVELGGGPGAGSLQPLALEGsLQKRGIVEQCCTS
ICsLYQLENYCN

>>>
```

Alvaro Sebastián Yagüe

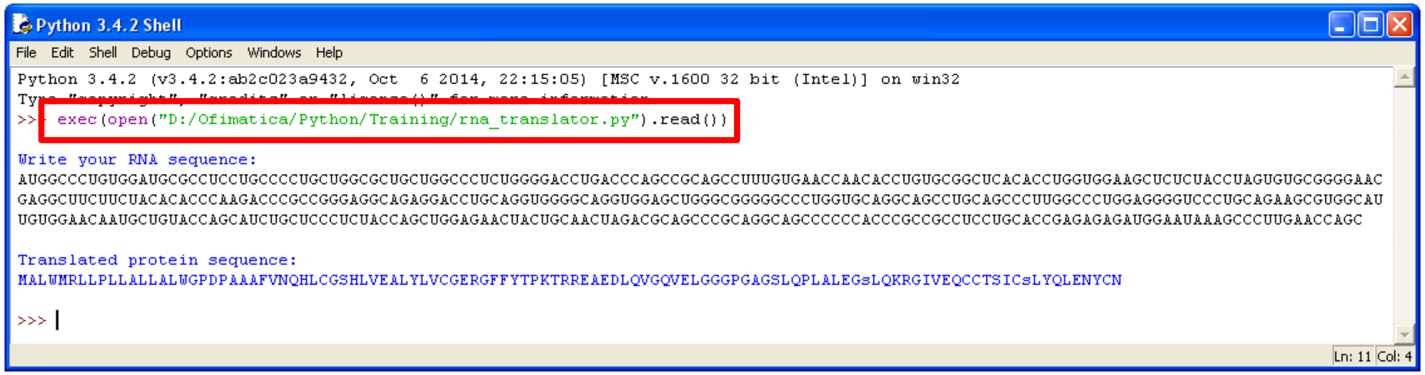
www.sixthresearcher.com

@SixthResearcher

Running a script

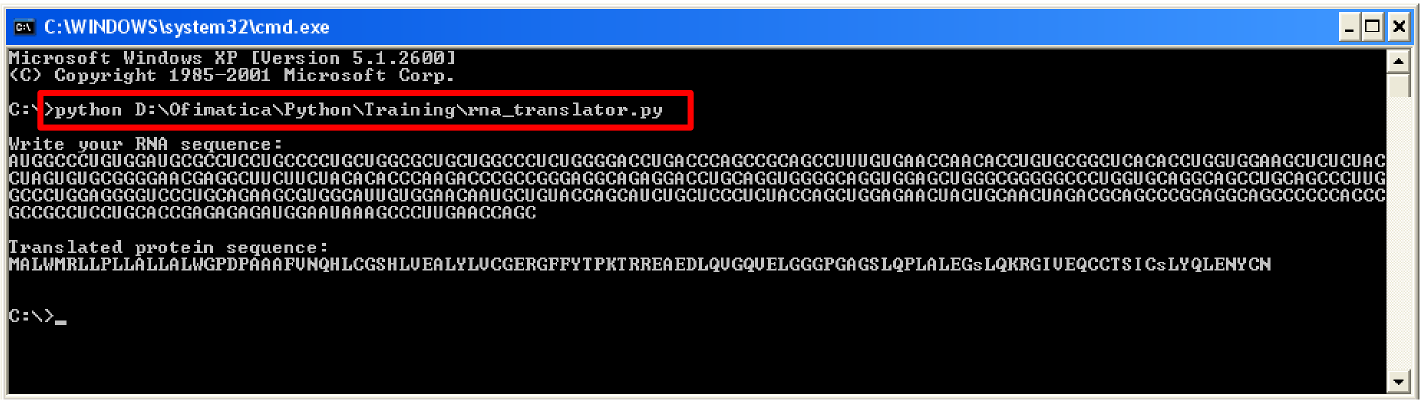
We can also **run the script with Python or Windows command line interpreters**:

- **Python interpreter (IDLE):**



```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:15:05) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> exec(open("D:/Ofimatica/Python/Training/rna_translator.py").read())

Write your RNA sequence:
AUGGCCCUGUGGAUGCGCCUCCUGCCCCUGCUGGCGCUGCUGGCCCUCUGGGGACCUGACCCAGCCGCAGCCUUUGUGAACCAACACCUGUGCGGCUCACACCUGGUGGAAGCUCUCUACCUAGUGUGCGGGGAAC
GAGGCUUCUUCUACACACCCAAGACCCGCCGGGAGGCAGAGGACCUGCAGGUGGGGCAGGUGGAGCUGGGCGGGGGCCCUGGUGCAGGCAGCCUGCAGCCCUUGGCCCUGGAGGGGUCCCUGCAGAAGCGUGGCAU
UGUGGAACAAUGCUGUACCAGCAUCUGCUCCCUCUACCAGCUGGAGAACUACUGCAACUAGACGCAGCCCGCAGGCAGCCCCCCACCCGCCGCCUCCUGCACCGAGAGAGAUGGAAUAAAGCCCUUGAACCAGC

Translated protein sequence:
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVGQVELGGGPGAGSLQPLALEGsLQKRGIVEQCCTSICsLYQLENYCN

>>> |
```

- **Windows command line (CMD):**



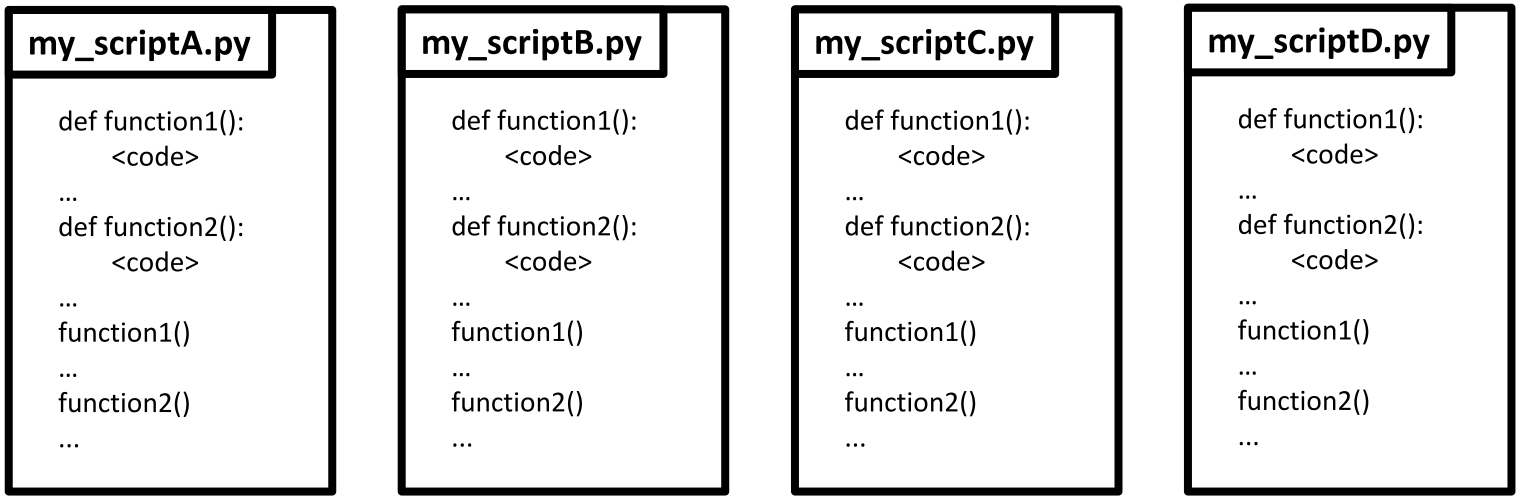
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>python D:\Ofimatica\Python\Training\rna_translator.py

Write your RNA sequence:
AUGGCCCUGUGGAUGCGCCUCCUGCCCCUGCUGGCGCUGCUGGCCCUCUGGGGACCUGACCCAGCCGCAGCCUUUGUGAACCAACACCUGUGCGGCUCACACCUGGUGGAAGCUCUCUAC
CUAGUGUGCGGGGAACGAGGCUUCUUCUACACACCCAAGACCCGCCGGGAGGCAGAGGACCUGCAGGUGGGGCAGGUGGAGCUGGGCGGGGGCCCUGGUGCAGGCAGCCUGCAGCCCUUG
GCCCUGGAGGGGUCCCUGCAGAAGCGUGGCAUUGUGGAACAAUGCUGUACCAGCAUCUGCUCCCUCUACCAGCUGGAGAACUACUGCAACUAGACGCAGCCCGCAGGCAGCCCCCCACCC
GCCGCCUCCUGCACCGAGAGAGAUGGAAUAAAGCCCUUGAACCAGC

Translated protein sequence:
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVGQVELGGGPGAGSLQPLALEGsLQKRGIVEQCCTSICsLYQLENYCN


C:\>_
```

Python modules

Python modules

A module is a Python file containing functions and variables used by multiple scripts. The module file has the module name with the suffix '.py' appended (like scripts, ex. my_module.py).

Modules are the solution to functions used by several scripts without writing them many times (once per script). Remember that functions are pieces of code writen once and used several times by a single script.



my_scriptA.py

```
def function1():
        <code>
…
def function2():
        <code>
…
function1()
…
function2()
…
```

my_scriptB.py

```
def function1():
        <code>
…
def function2():
        <code>
…
function1()
…
function2()
…
```

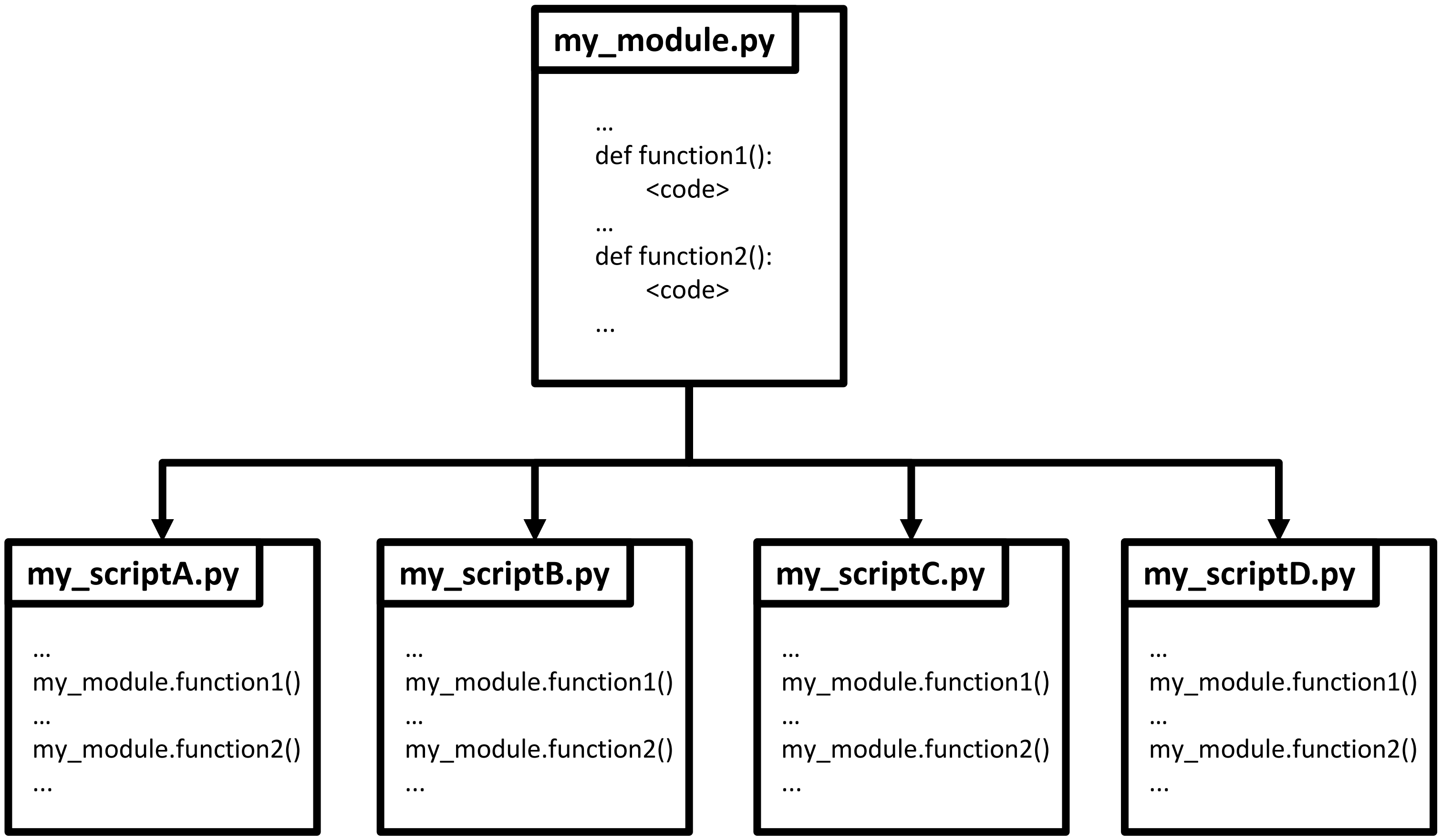
my_scriptC.py

```
def function1():
        <code>
…
def function2():
        <code>
…
function1()
…
function2()
…
```

my_scriptD.py

```
def function1():
        <code>
…
def function2():
        <code>
…
function1()
…
function2()
…
```

Python modules



Python modules

To use a module we have to import it with the ‘import’ command at the beginning of the script.

The first time a module is loaded into a running Python script, it is initialized by executing the code in the module once. If another module in your code imports the same module again, it will not be loaded twice but only once.

Functions included in the module can be called with the module name plus ‘.’ (dot) and the function name.

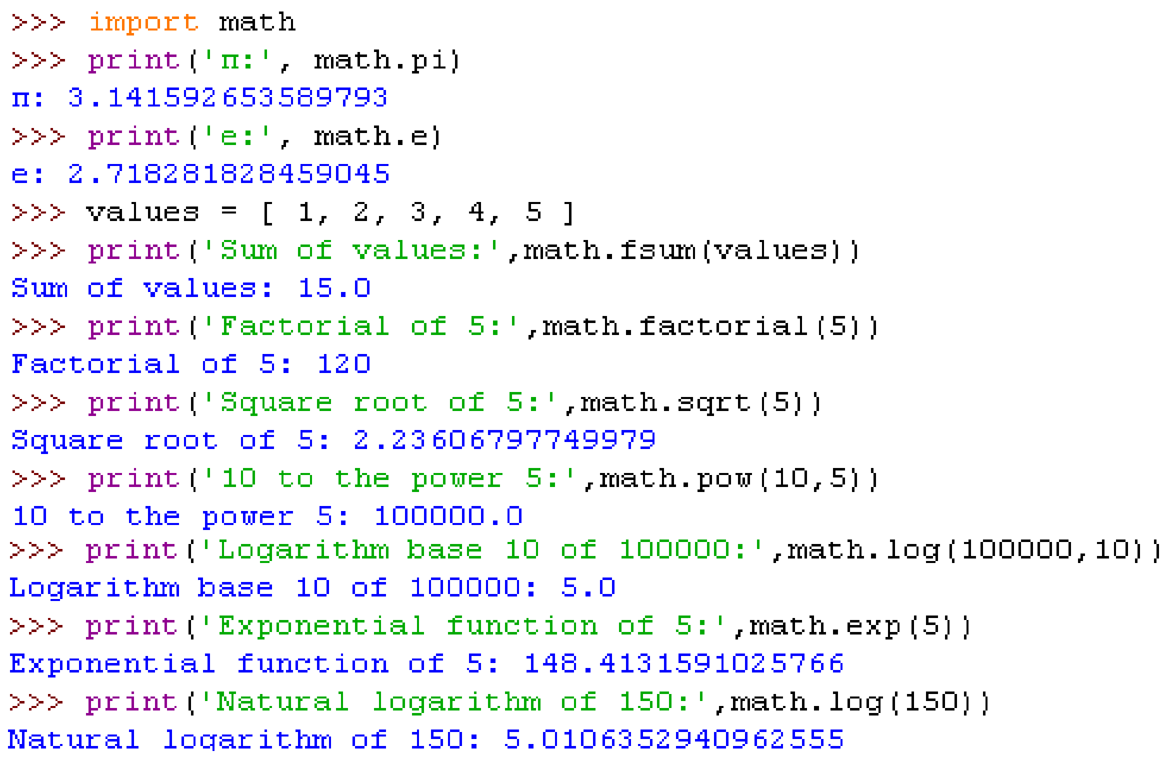
We already used the module ‘os’ for working with files and folders:

```
>>> import os
>>> os.getcwd()
>>> os.makedirs(<path>)
>>> os.chdir(<path>)
>>> os.listdir(<path>)
```

You can check out the full list of built-in modules in the Python standard library here.

Python modules

The module ‘math’ includes useful mathematical operations:



```
>>> import math
>>> print('π:', math.pi)
π: 3.141592653589793
>>> print('e:', math.e)
e: 2.718281828459045
>>> values = [ 1, 2, 3, 4, 5 ]
>>> print('Sum of values:',math.fsum(values))
Sum of values: 15.0
>>> print('Factorial of 5:',math.factorial(5))
Factorial of 5: 120
>>> print('Square root of 5:',math.sqrt(5))
Square root of 5: 2.23606797749979
>>> print('10 to the power 5:',math.pow(10,5))
10 to the power 5: 100000.0
>>> print('Logarithm base 10 of 100000:',math.log(100000,10))
Logarithm base 10 of 100000: 5.0
>>> print('Exponential function of 5:',math.exp(5))
Exponential function of 5: 148.4131591025766
>>> print('Natural logarithm of 150:',math.log(150))
Natural logarithm of 150: 5.0106352940962555
```

If you want to learn more… https://pymotw.com/3/math/

Python modules

Let’s practice with the module ‘datetime’. This module includes functions for working with dates and times.

‘today’ is a new data type called object

```
>>> import datetime
>>> today = datetime.date.today()
>>> print(today)
2016-12-15
>>> print('ctime:', today.ctime())
ctime: Thu Dec 15 00:00:00 2016
>>> print('Year:', today.year)
Year: 2016
>>> print('Month:', today.month)
Month: 12
>>> print('Day:', today.day)
Day: 15
>>> one_day = datetime.timedelta(days=1)
>>> print('One day:', one_day)
One day: 1 day, 0:00:00
>>> yesterday = today - one_day
>>> print('Yesterday:', yesterday)
Yesterday: 2016-12-14
>>> tomorrow = today + one_day
>>> print('Tomorrow:', tomorrow)
Tomorrow: 2016-12-16
```

If you want to learn more… https://pymotw.com/3/datetime/

Python modules

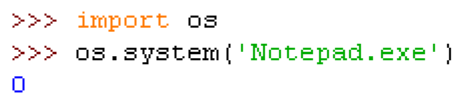
The ‘os’ module include many functions to interact with the file system:

```
>>> import os
>>> dir = os.getcwd()
>>> print(dir)
D:\Ofimatica\Python
>>> filenames = os.listdir(dir)
>>> print(filenames)
['DLLs', 'Doc', 'include', 'Lib', 'libs', 'LICENSE.txt', 'Logs', 'NEWS.txt
', 'python.exe', 'pythonw.exe', 'qt.conf', 'README.txt', 'Removespyder.exe
', 'Scripts', 'spyder-wininst.log', 'tcl', 'Tools', 'Training']
>>> for filename in filenames:
        print(filename)
        print(os.path.join(dir, filename))


DLLs
D:\Ofimatica\Python\DLLs
Doc
D:\Ofimatica\Python\Doc
include
D:\Ofimatica\Python\include
Lib
D:\Ofimatica\Python\Lib
...
```

If you want to learn more… https://pymotw.com/3/os/

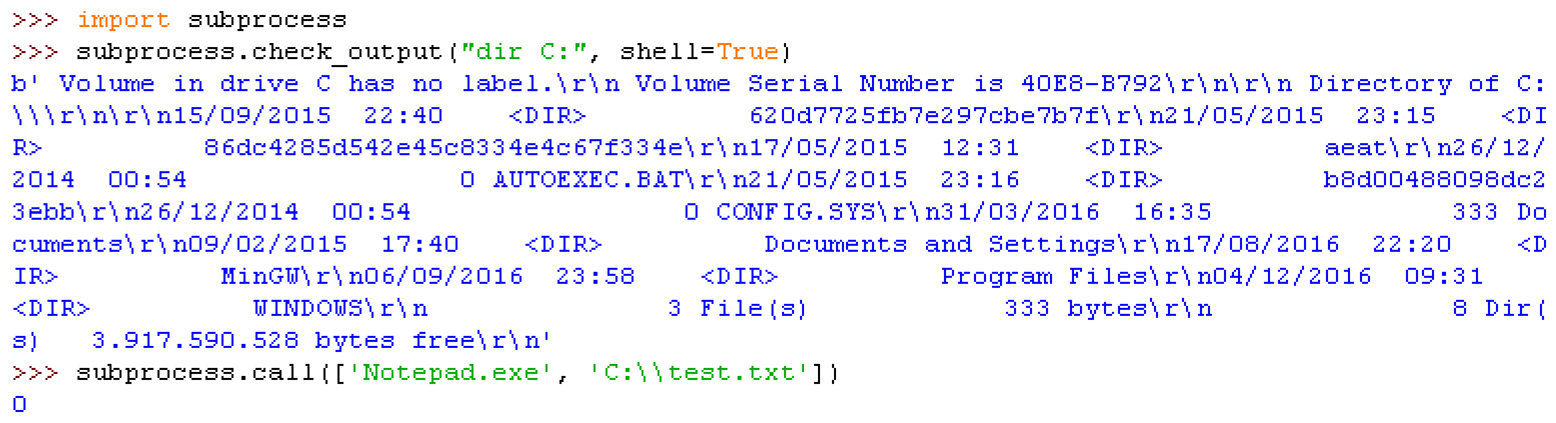
Python modules

'os.system()' is a useful function that allows to execute external commands/programs:



```
>>> import os
>>> os.system('Notepad.exe')
0
```

But 'os.system()' functionality is limited and it has been improved and extended by the module 'subprocess':



```
>>> import subprocess
>>> subprocess.check_output("dir C:", shell=True)
b' Volume in drive C has no label.\r\n Volume Serial Number is 40E8-B792\r\n\r\n Directory of C:
\\\r\n\r\n15/09/2015  22:40    <DIR>          620d7725fb7e297cbe7b7f\r\n21/05/2015  23:15    <DI
R>          86dc4285d542e45c8334e4c67f334e\r\n17/05/2015  12:31    <DIR>          aeat\r\n26/12/
2014  00:54                0 AUTOEXEC.BAT\r\n21/05/2015  23:16    <DIR>          b8d00488098dc2
3ebb\r\n26/12/2014  00:54                0 CONFIG.SYS\r\n31/03/2016  16:35              333 Do
cuments\r\n09/02/2015  17:40    <DIR>          Documents and Settings\r\n17/08/2016  22:20    <D
IR>          MinGW\r\n06/09/2016  23:58    <DIR>          Program Files\r\n04/12/2016  09:31
<DIR>          WINDOWS\r\n               3 File(s)          333 bytes\r\n               8 Dir(
s)   3.917.590.528 bytes free\r\n'
>>> subprocess.call(['Notepad.exe', 'C:\\test.txt'])
0
```

If you want to learn more… https://pymotw.com/3/subprocess/

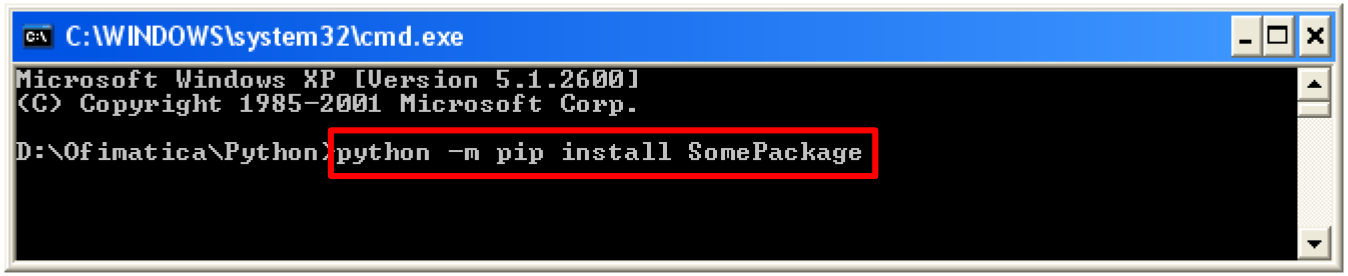
Python modules

You can install additional modules into Python appart from the built-in ones:

- **Python interpreter (IDLE):**

```
>>> import subprocess
>>> subprocess.call(['pip', 'install', SomePackage])
```

- **Windows command line (CMD):**

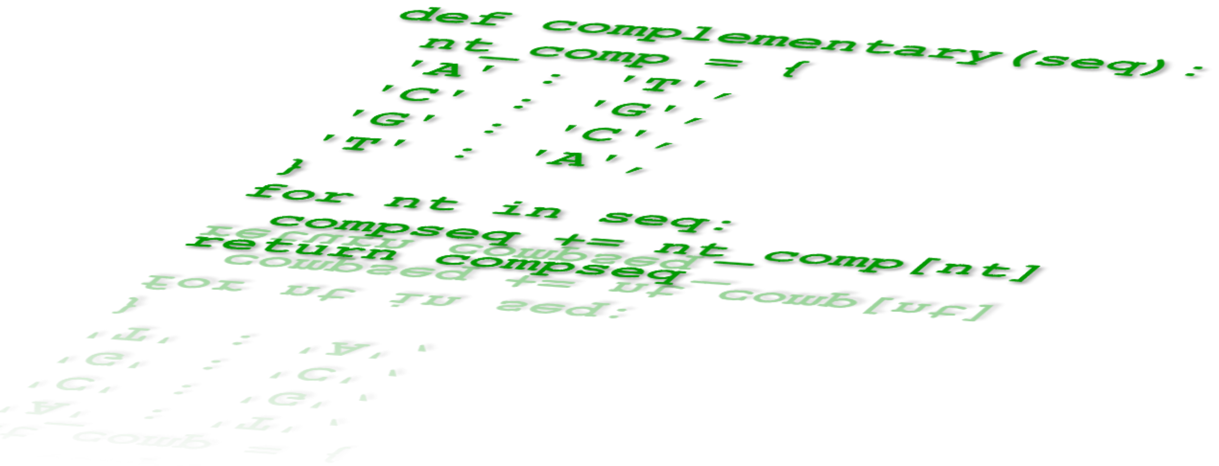


```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

D:\Ofimatica\Python>python -m pip install SomePackage
```

Python for scientists

Next lesson...

NumPy, SciPy and Matplotlib



```
def complementary(seq):
    nt_comp = {
        'A' : 'T',
        'C' : 'G',
        'G' : 'C',
        'T' : 'A',
    }
    for nt in seq:
        compseq += nt_comp[nt]
    return compseq
```