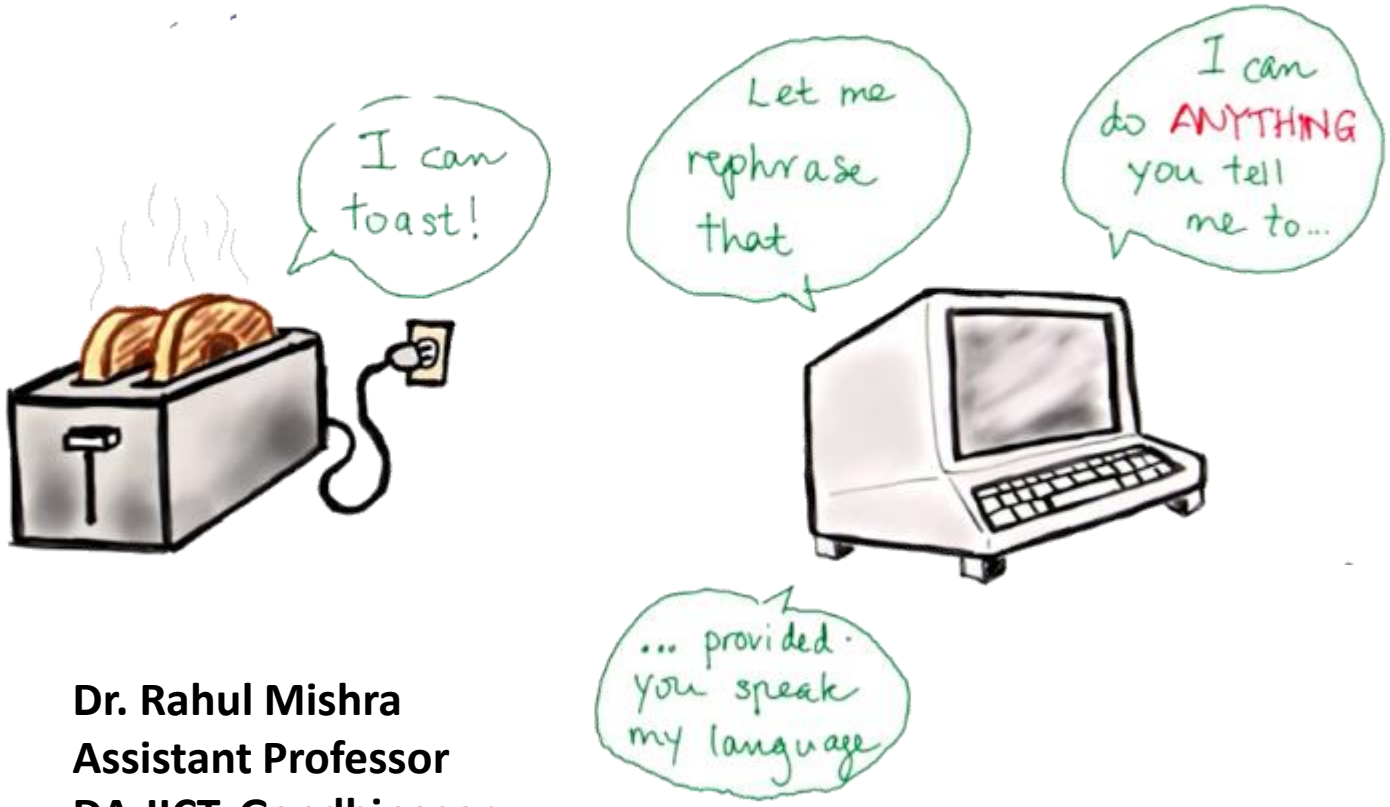
Programming Lab

Autumn Semester

Course code: PC503



Dr. Rahul Mishra
Assistant Professor
DA-IICT, Gandhinagar

Images are collected from the web and may be subject to copyright

Lecture 12

Dictionaries and Modules

Dictionaries

Looping Techniques

When looping through a sequence, the position index and corresponding value can be retrieved at the same time using the enumerate() function.

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
...     print(i, v)
...
0 tic
1 tac
2 toe
```

To loop over two or more sequences at the same time, the entries can be paired with the zip() function

```
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
...     print('What is your {0}?  It is {1}.'.format(q, a))
...
What is your name?  It is lancelot.
What is your quest?  It is the holy grail.
What is your favorite color?  It is blue.
```

To loop over a sequence in reverse, first specify the sequence in a forward direction and then call the reversed() function.

```
>>> for i in reversed(range(1, 10, 2)):
...     print(i)
...
9
7
5
3
1
```

To loop over a sequence in sorted order, use the sorted() function which returns a new sorted list while leaving the source unaltered.

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange',
'banana']
>>> for i in sorted(basket):
...     print(i)
...
apple
apple
banana
orange
orange
pear
>>>
```

Using set() on a sequence eliminates duplicate elements.

The use of sorted() in combination with set() over a sequence is an idiomatic way to loop over unique elements of the sequence in sorted order.

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange',
'banana']
>>> for f in sorted(set(basket)):
...     print(f)
...
apple
banana
orange
pear
>>>
```

It is sometimes tempting to change a list while you are looping over it; however, it is often simpler and safer to create a new list instead.

```
>>> import math
>>> raw_data = [56.2, float('NaN'), 51.7, 55.3, 52.5,
float('NaN'), 47.8]
>>> filtered_data = []
>>> for value in raw_data:
...     if not math.isnan(value):
...         filtered_data.append(value)
...
>>> filtered_data
[56.2, 51.7, 55.3, 52.5, 47.8]
>>>
```

More on Conditions

- The conditions used in **while and if** statements can contain any operators, not just comparisons.
- The comparison **operators in and not** in are membership tests that determine whether **a value is in (or not in) a container.**
- The **operators is and is not compare whether two objects are really the same object**.
- All comparison operators have the same priority, which is lower than that of all numerical operators.
- Comparisons can be chained. **For example, a < b == c tests** whether a is less than b, and moreover b equals c.
- Comparisons may be combined using the Boolean operators **and and or, and** the outcome of a comparison (or of any other Boolean expression) may be negated with not.

More on Conditions

It is possible to assign the result of a comparison or other Boolean expression to a variable.

For example,

```
>>> string1, string2, string3 = '', 'Trondheim', 'Hammer
Dance'
>>> non_null = string1 or string2 or string3
>>> non_null
'Trondheim'
```

Comparing Sequences and Other Types

```
>>> (1, 2, 3)              < (1, 2, 4)
True
>>> (1, 2, 3)              < (1, 2, 4)
True
>>> [1, 2, 3]              < [1, 2, 4]
True
>>> 'ABC' < 'C' < 'Pascal' < 'Python'
True
>>> (1, 2, 3, 4)           < (1, 2, 4)
True
>>> (1, 2)                 < (1, 2, -1)
True
>>> (1, 2, 3)              == (1.0, 2.0, 3.0)
True
>>> (1, 2, ('aa', 'ab'))   < (1, 2, ('abc', 'a'), 4)
```

We will have a new text editor today……

Modules

- If you want to write a somewhat longer program, you are better off using a ***text editor to prepare*** the input for the interpreter and running it with that file as input instead. This is known as creating a script.

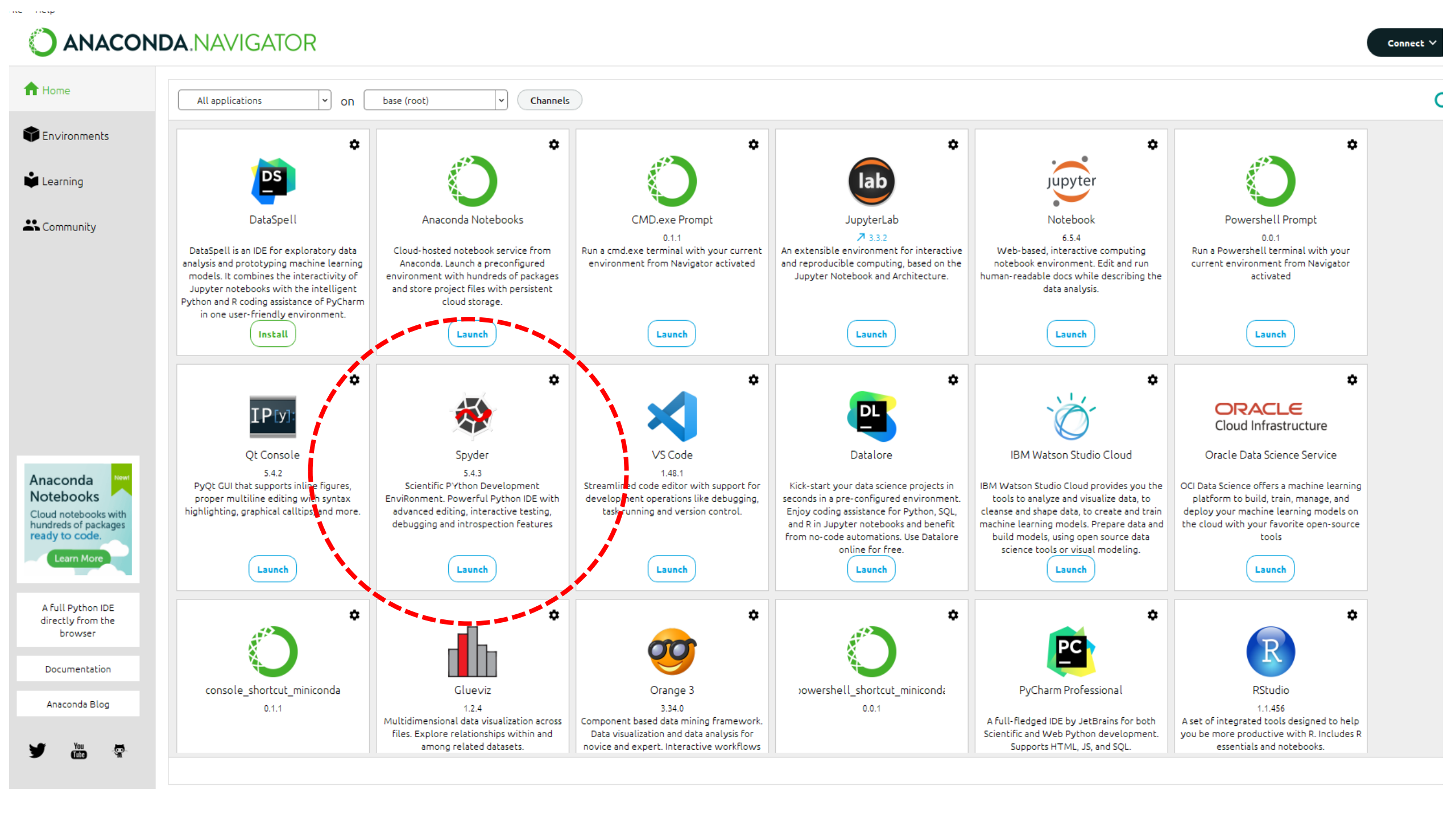
- As ***your program gets longer, you may want to split it into several files*** for easier maintenance. You may also want to use a handy function that you've written in several programs without copying its definition into each program.

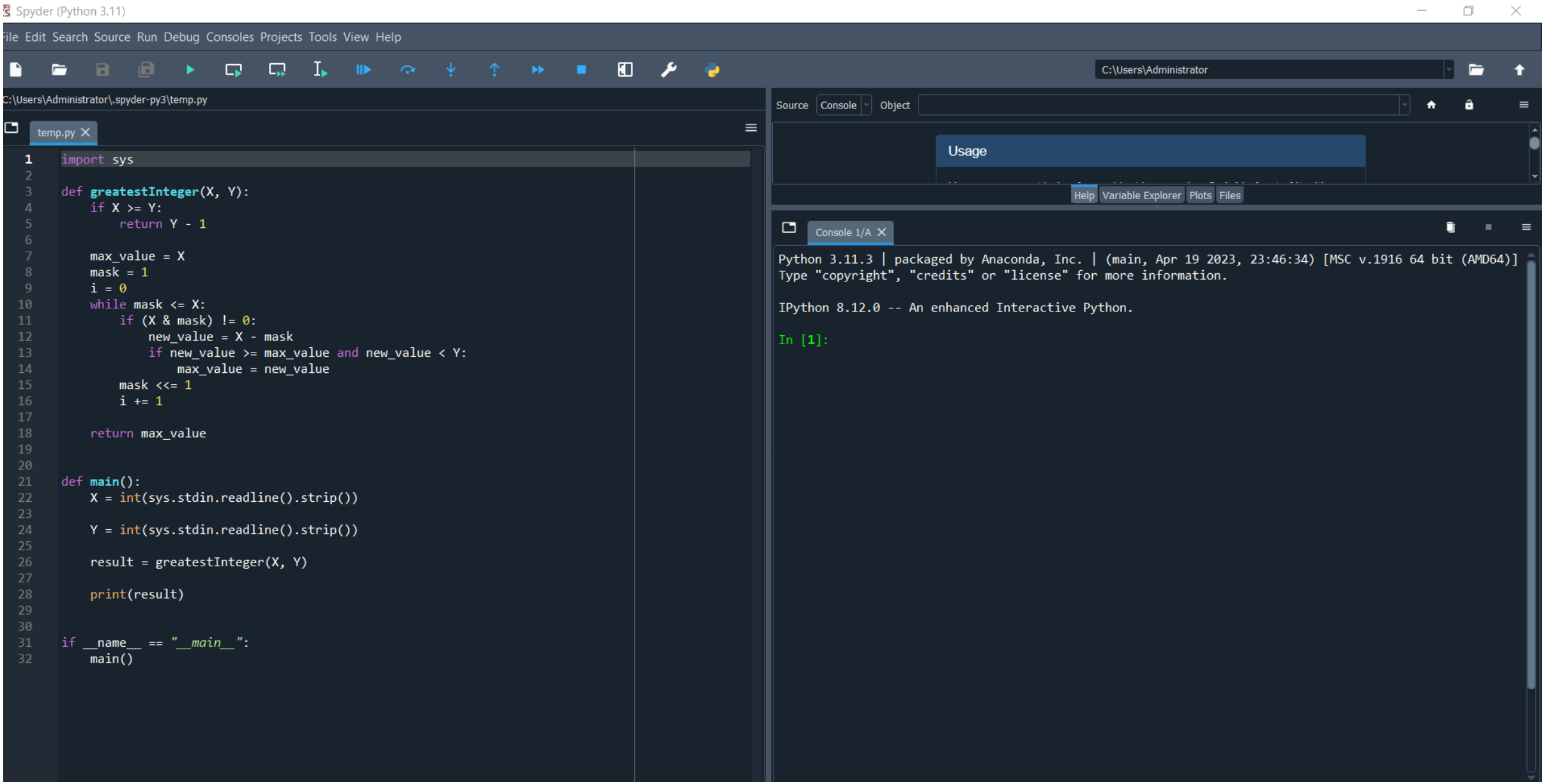
- To support this, ***Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter.***

- Such a file is called a **module**; definitions from a module can be imported into other modules or into the main module (***the collection of variables that you have access to in a script executed at the top level and in calculator mode***).

Modules

- A module is a file containing Python definitions and statements.
- The file name is the module name with the ***suffix .py*** appended.
- Within a module, the module's name (as a string) is available as the value of the global variable __name__.
- For instance, use your favorite text editor to create a file called fibo.py


ANACONDA.NAVIGATOR
Connect
Home
Environments
Learning
Community
All applications
on
base (root)
Channels
DataSpell
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.
Install
Anaconda Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.
Launch
CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch
JupyterLab
3.3.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.5.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated
Launch
Qt Console
5.4.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
Spyder
5.4.3
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
VS Code
1.48.1
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch
Datalore
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.
Launch
IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.
Launch
Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools
Launch
Anaconda Notebooks
New!
Cloud notebooks with hundreds of packages ready to code.
Learn More
A full Python IDE directly from the browser
Documentation
Anaconda Blog
console_shortcut_miniconda
0.1.1
Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Orange 3
3.34.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows
powershell_shortcut_miniconda
0.0.1
PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.


Spyder (Python 3.11)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Administrator
C:\Users\Administrator\.spyder-py3\temp.py
temp.py
import sys
def greatestInteger(X, Y):
if X >= Y:
return Y - 1
max_value = X
mask = 1
i = 0
while mask <= X:
if (X & mask) != 0:
new_value = X - mask
if new_value >= max_value and new_value < Y:
max_value = new_value
mask <<= 1
i += 1
return max_value
def main():
X = int(sys.stdin.readline().strip())
Y = int(sys.stdin.readline().strip())
result = greatestInteger(X, Y)
print(result)
if __name__ == "__main__":
main()
Source Console Object
Usage
Help Variable Explorer Plots Files
Console 1/A
Python 3.11.3 | packaged by Anaconda, Inc. | (main, Apr 19 2023, 23:46:34) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 8.12.0 -- An enhanced Interactive Python.
In [1]:

fibo.py

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

main.py

```
import fibo

fibo.fib(1000)

#if this not working
then use print()


fibo.fib2(100)

fibo.__name__
```