



# Topic 1

## Introduction to Data Programming

### Learning Outcomes

After completing this topic and the recommended reading, you should be able to:

- Set up and run Jupyter Notebook on a Windows, Mac or Linux operating system.
- Use Jupyter Notebook to write and edit code.
- Write and explain simple Python programs using variables and mathematical operators.

# **1. Introduction to Data Programming**

## ***Data (definition)***

- “Facts and statistics collected together for reference or analysis.”  
[Oxford English Dictionary]
- “Information, especially facts or numbers, collected to be examined and considered and used to help decision-making, or information in an electronic form that can be stored and used by a computer.”  
[Cambridge Dictionary]
- “Factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation.”  
[Merriam-Webster]
- “Data are individual facts, statistics, or items of information, often numeric, that are collected through observation. In a more technical sense, data are a set of values of qualitative or quantitative variables about one or more persons or objects.”  
[Wikipedia]

## ***Information (definition)***

- “Facts provided or learned about something or someone.”  
[Oxford English Dictionary]
- “Facts or details about a situation, person, event, etc.”  
[Cambridge Dictionary]

- “Knowledge obtained from investigation, study, or instruction.”

[Merriam-Webster]

- “Knowledge communicated or received concerning a particular fact or circumstance; knowledge gained through study, communication, research, instruction, etc.”

[Dictionary.com]

## ***Data vs. Information***


- ***Data***
  - Raw, unorganised facts that need to be processed.
  - Unusable until it is organised.
- ***Information***
  - Created when data is processed, organised, and structured.
  - Needs to be put in an appropriate context in order to become useful.

### **Data Science**




## ***Programming and Data***

- Tasks to undertake for data programming
  - Data collection
  - Data processing (wrangling)
  - Data visualisation

- Train and apply algorithms from fields such as machine learning, statistics, data mining, optimisation, image processing, etc.
- ***Programming***
  - The process of producing an executable computer program that performs a specific task.
  - The purpose is to find a sequence of instructions that automate the implementation of the task for solving a given problem.
- ***Programming Language***
  - The source code of a program is written in one or more languages that are intelligible to humans, rather than machine code, which is directly executed by the CPU.
  - Python  python<sup>™</sup>
    - <https://www.python.org/>

## **2. Introduction to Development Environments**

### ***Source-code Editors***

- ***Source-code editor***, or programming text editor, is a fundamental programming tools designed specifically for editing source code of computer programs.
- It highlights the syntax elements of your programs, and provides many features that aid in your program development.
- Examples:
  - Visual Studio Code [<https://code.visualstudio.com/>]
  - Notepad++ (Windows only) [<https://notepad-plus-plus.org/>]
  - Vim [<https://www.vim.org/>]
  - Sublime Text (not open source) [<https://www.sublimetext.com/>]
  - Atom [<https://atom.io/>]
  - Emacs [<https://www.gnu.org/software/emacs/>]
  - TextMate (Macs only) [<https://macromates.com/>]
  - Jupyter 
    - <https://jupyter.org/>

### ***Integrated Development Environments (IDEs)***

- ***Integrated development environment*** is a software application that provides comprehensive facilities to computer programmers for software development.
- An IDE normally consists of at least a source code editor, build automation tools and a debugger.
- Examples:
  - Spyder [<https://www.spyder-ide.org/>]

- RStudio [<https://rstudio.com>]
- Eclipse [<https://www.eclipse.org/>]
- Microsoft Visual Studio [<https://visualstudio.microsoft.com/vs/>]
- Wing Python IDE [<https://wingware.com>]

## ***Markdown / Markup Languages***

- ***Markdown*** is a markup language that consists of a set of rules for adding formatting elements to plain text documents
  - Boldface, italics, headers, paragraphs, lists, code blocks, images, etc.
  - <https://www.markdownguide.org/>
- Invented by *John Gruber*
  - The overriding design goal for Markdown's formatting syntax is to make it as readable as possible.
  - The idea is that a Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions
- Examples
  - HTML; XML; LaTeX

## ***Version Control Systems***

- ***Version Control*** is a class of systems responsible for managing changes to computer programs, documents, large websites, or other collections of information.
- ***Version Control Systems*** (VCS) are software tools that help software teams manage changes to source code over time.
  - Undertakes the tedious task of keeping track of the changes to all project's files and who made them

- Allows users to recover any previous version at any given time
- Examples:
  - Subversion [<https://subversion.apache.org>]
  - Git 
    - <https://git-scm.com/>
  - GitHub 
    - <https://github.com/>

## ***Package/Environment Manager***

- ***Package manager***, or package management system, is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer in a consistent manner. Also deals with packages, distributions of software and data in archive files.
- ***Environment manager*** enables personalised, consistent desktop environments without cumbersome roaming profiles or scripts.
- Example:

- Anaconda 
  - <https://www.anaconda.com/>

## ***Installing Anaconda***

- Go to *Anaconda*, download *Anaconda Individual Edition*
  - <https://www.anaconda.com/products/distribution>
- Packages include
  - *conda*
    - package management system

- *pandas, scikit-learn, nltk*
  - packages for data science
- *Anaconda Navigator*
  - a graphical user interface
- *QtConsole*
  - an interactive Python environment
- *Spyder*
  - a standard cross-platform IDE for Python
- *Jupyter Notebook*
  - an interactive web-browser based application for creating and sharing code

## ***Package Installer for Python (pip)***

- ***pip*** is the de facto and recommended package-management system written in Python and is used to install and manage software packages.
- It connects to an online repository of public packages, called the Python Package Index.
- We use ***pip*** to install packages from the Python Package Index
- Examples
  - *pip install beautifulsoup4*
  - *pip install -r dependencies*
    - Install packages based on dependencies in code
  - *pip freeze*
    - See all the packages installed



### **3. Introduction to Python**

- Open-source, interpreted, high-level, object-oriented, general-purpose, easy to download, write and read
- Named for the British comedy group *Monty Python*
- Simpler language, allow us to focus less on the language and more on problem solving
- Many of the best parts of other languages are included
  - Data structures
  - Controls
  - Many packages for common tasks

#### ***Variables***

- ***Variable*** is a named piece of memory whose value can change during the running of the program; ***constant*** is a value which cannot change as the program runs.
  - Python doesn't use constant
- We use ***variable names*** to represent objects (number, data structures, functions, etc.) in our program, to make our program more readable.
  - All variable names must be one word, spaces are never allowed.
  - Can only contain alpha-numeric characters and underscores.
  - Must start with a letter or the underscores character.
  - Cannot begin with a number.
  - Case-sensitive
  - Standard way for most things named in Python is lower with under
    - Lower case with separate words joined by an underscore

#### ***Comments***

- Not processed by the computer, valued by other programmers.
- Header comments
  - Appear at beginning of a program or a module
  - Provide general information
- Step comments or in-line comments
  - Appear throughout program
  - Explain the purpose of specific portion of code
- Often comments delineated by
  - `//` comment goes here
  - `/*` comment goes here `*/`
  - `#` Python uses this

## ***Python Operations***

- Assignment Operator
  - “=”
  - Example:
    - `a = 67890/12345`  
*# compute the ratio, store the result in ram, assign to a*  
*# the value of a is 5.499392*
    - `b = a`  
*# b pointing to value of a*
- Output
  - “print()”
  - Example:
    - `print('Hello World!')`      *# print the string literals*
    - `print(a)`                      *# print the value of a*

## ***Data Types in Python***

- Declaration of variables in Python is not needed
  - Use an assignment statement to create a variable

- **Float**

- Stores real numbers
- $a = 4.6$
- `print(type(a))`

```
<class 'float'>
```

- **Integer**

- Stores integers
- $b = 10$
- `print(type(b))`

```
<class 'int'>
```

- **Conversion**

- `int(a)`                      *# convert float to int*                       $\Rightarrow 4$
- `float(b)`                      *# convert int to float*                       $\Rightarrow 10.0$

- **Basic arithmetic operators**

- $3 + 2$                       *# Addition*                       $\Rightarrow 5$
- $5 - 2$                       *# Subtraction*                       $\Rightarrow 3$
- $5 * -2$                       *# Multiplication*                       $\Rightarrow -10$
- $5 / 2.5$                       *# Division*                       $\Rightarrow 2.0$
- $2 ** 2$                       *# Exponentiation*                       $\Rightarrow 4$
- $10 \% 3$                       *# Modulus*                       $\Rightarrow 1$

- $10 // 3$        $\#Floor\ Division$        $\Rightarrow 3$

- **String**

- Stores strings
- `phrase = 'All models are wrong, but some are useful.'`
- `phrase[0:3]` *# slicing character 0 up to 2*  
=> All
- `phrase.find('models')` *# find the starting index of word*  
=> 4
- `phrase.find('right')` *# word not found*  
=> -1
- `phrase.lower()` *# set to lower case*  
=> 'all models are wrong, but  
some are useful.'
- `phrase.upper()` *# set to upper case*  
=> 'ALL MODELS ARE  
WRONG, BUT SOME ARE  
USEFUL.'
- `phrase.split(',')` *# split strings into list, base on delimiter*  
=> ['All models are wrong',  
' but some are useful.']

- **Boolean**

- Stores logical or Boolean values of TRUE or FALSE
  - $k = 1 > 3$
  - `print(k)`
- ```
False
```
- `print(type(k))`

```
<class 'bool'>
```

- Logical operators
  - Conjunction (AND): “**and**”
  - Disjunction (OR): “**or**”
  - Negation (NOT): “**not**”

| <u><b>a</b></u> | <u><b>b</b></u> | <u><b>a and b</b></u> | <u><b>a or b</b></u> | <u><b>not a</b></u> |
|-----------------|-----------------|-----------------------|----------------------|---------------------|
| T               | T               | T                     | T                    | F                   |
| T               | F               | F                     | T                    | F                   |
| F               | T               | F                     | T                    | T                   |
| F               | F               | F                     | F                    | T                   |

## ***Data Structures in Python***

- **Tuples**
  - Store ordered collection of objects
  - Immutable: elements cannot be modified, added or deleted
  - Written with round brackets “( )”
    - *tuple1 = (“apple”, “banana”, “cherry”, “orange”, “kiwi”, “melon”, “mango”)*
    - *tuple2 = (“Handsme Koh”, 4896, 13.14, True)*
  - Accessing elements by indexing
    - *tuple1[0]*    #first element index    => ‘apple’
    - *tuple1[-1]*    # last element index    => ‘mango’
    - *tuple1[2:5]*    # range of elements    => (‘cherry’, ‘orange’, ‘kiwi’)
- **Lists**

- Store ordered collection of objects; mutable
- Written with square brackets “[ ]”
  - `list1 = ["apple", "banana", "cherry"]`
  - `list2 = ["Handsome Koh", 4896, 13.14, True]`
- Changing elements
  - `list1.append("orange")` *# add to last position*  
      $\Rightarrow$  `['apple', 'banana', 'cherry', 'orange']`
  - `list1[2] = "coconut"` *# modify index element*  
      $\Rightarrow$  `['apple', 'banana', 'coconut', 'orange']`
  - `list1.remove("apple")` *# delete elements*  
      $\Rightarrow$  `['banana', 'coconut', 'orange']`
  - `list1.insert(2, "durian")` *# insert element at position*  
      $\Rightarrow$  `['banana', 'coconut', 'durian', 'orange']`
- Sets
  - Store unordered, unindexed, nonduplicates collection of objects
  - Written with square brackets "{ }"
    - `set1 = {"apple", "banana", "cherry"}`
    - `set2 = {"apple", "samsung"}`
  - Set operations
    - `set1.union(set2)` *# Union both sets*  
      $\Rightarrow$  `{'apple', 'banana', 'cherry', 'samsung'}`
    - `set1.intersection(set2)` *# Intersect both sets*  
      $\Rightarrow$  `{'apple'}`

- **Dictionaries**

- Store unordered collection of objects
- Written with square brackets “{ }”, and “key:value” pair
  - *thisdict* = {“brand”: “Ford”, “model”: “Mustang”,  
“year”: 1964}
- Accessing/modifying elements by key name
  - *thisdict*[“model”]                      => ‘Mustang’
  - *thisdict*[“year”] = 2018              => {‘brand’: ‘Ford’,  
   *thisdict*[“color”] = “red”              ‘model’: ‘Mustang’,  
                                                  ‘year’: 2018,  
                                                  ‘color’: ‘red’}

## **4. Introduction to Jupyter Notebook**

- **Jupyter Notebook** is a web-based interactive computing platform.
- “Julia” + “Python” + “R”
- Integrate code and output into a single document contains:
  - Live code, mathematical equations, visualisations, and explanatory/narrative text, interactive dashboards and other media
- Can be easily shared
  - Notebook files have “.ipynb” extension
  - Export to “.html” and “.pdf” extensions
- Launch “Jupyter Notebook” from “Anaconda Navigator”
- Create new notebook
  - “File” → “New Notebook” → “Python 3”
- Exporting notebook
  - “File” → “Download as” → “HTML (.html)”
  - “File” → “Print Preview” (for PDF)
- Shutting Down Jupyter
  - “File” → “Close and Halt”
  - Quit



## **5. Exercises**

### ***1.301 Practice Exercises (Coursera)***

- Refers to “1.301 part-1.html”

### ***1.302 A bit more Python – our first downloadable notebook! (Coursera)***

- Refers to “1.302 pythonPractice.html”

### ***1.304 World’s Population***

- Refers to “1.304 Topic 1 - Lab.html”

## **6. Practice Quiz**

- Work on *Practice Quiz 01* posted on Canvas.

## **Useful Resources**

- - <http://>