CT1112
Professional Skills I

Learning Manual

David O'Sullivan and Shahid Hussain, 2021

1. Introduction

CT1112 Professional Skills I is an introduction to **scientific skills**. In-class tutorials are delivered and will cover:

- Programming Skills:
    - Operating Systems
    - LaTeX
    - Python3
    - Excel

Learning outcomes will be assessed by a **video sharing** exercise and a major written report entitled '**Programming Skills'** similar to a shorter version of this Learning Manual and written in the LaTeX programming language.

Recommended Text

Zelle, J.M., Python programming: an introduction to computer science. Franklin, Beedle & Associates, Inc. (*Try searching Google for "Zelle Python programming pdf")*

2. Computer Programming

Programming is about creating software e.g. operating systems and computer applications that run on computer devices such as personal computers. Programs with an explicit list of instructions are executed sequentially by the computers central processing unit (CPU). CPUs execute instructions or source code exactly and unlike the words on this page are intolerant of errors. A program is similar to the instructions in a cooking recipe:

1. Ingredients
    a. Milk = 0.2ltr
    b. Eggs = 1
    c. Flour = 100g
    d. Mixing Bowl = 2ltr
    e. Large Spoon
2. Method
    a. Place Flour in Mixing Bowl
    b. Add Eggs
    c. Add 10ml of Milk
    d. Stir with Large Spoon and test thickness
    e. If thickness is high, then return to line 2c
    f. Else stop

The recipe or source code above is written in the English language. Computer programs are written in a programming language with its own unique syntax (rules) and semantics (meaning). The CPU can only understand sequences of 1's and 0's (e.g. 11010010…) arranged in bytes of mainly 32 or 64 bits. 64 bits means that there are 2^{64} or 18,446,744,073,709,551,616 unique memory addresses. Computer programs essentially label and then manipulate data stored in individual memory addresses. Each 1 or 0 manifests itself as electrical potential e.g. 5 volts or 0 volts, in tiny transistors within the CPU. There are approximately 300 million transistors in a CPU the size of a postage stamp. Various layers of system and application programs process high level human-readable language into this low level machine code. Programs in the lower layers include assembly language, operating systems (OS) and programming languages. Programs in the higher layers include word processor, game and graphics applications.

Operating Systems

The layer we experience when we first turn on a personal computer is the OS. Today, the OS is presented as a graphical user interface (GUI) containing windows, command menus and mouse pointer. Five of the most common operating systems for personal computers, smart phones and tablets are **Microsoft Windows, Apple macOS, Linux, Google Android and Apple iOS.** Applications are opened within the OS GUI by double-clicking on an icon of the application. In the past, there was no GUI and users needed to type the name of the application into the OS command prompt on a blank screen that looked something like this:

```
C:\>  word.exe
```

OS's define how a computer stores files, switches between applications, manages memory, maintains security, and interacts with peripherals such as keyboards, monitors, mouse and sensors. Common command line OS commands include:

Mac OS	Description	Windows OS
cd	Goto home directory	cd
cd <folder>	Change directory, e.g. cd Documents	cd <folder>
ls	Display name of files and subdirectories	dir
mkdir <dir>	Create new folder named <dir>	mkdir <dir>
open [file]	Opens a file	

pwd	Show path of current (working) directory	cd
rm -R <dir>	Delete a folder and its contents	rm <dir>
rm <file>	Delete a file (caution)	del <file>
rmdir <dir>	Delete an empty folder	rmdir <dir>
Ctrl + C	Stop whatever is running	Ctrl + Alt + Del
clear	Clear the screen	cls

Programming Languages

When we type a letter written in plain English, into say MS Word on the Personal Computer, the application is designed to show us exactly what we type on the computer screen. To achieve this, MS Word was written in the programming language, C++ and designed to accept our keyboard characters and display them on the screen verbatim. In doing this, the C++ programme converts our keyboard characters into machine code that is then manipulated by the CPU. There are hundreds of high level programming languages, like C++. They have all been designed around human-readable syntax. Listed below are some of the more common programming languages.

Language	**Description**
C/C++	General-purpose language
C#	Microsoft .NET platform
Java	Object-oriented language
JavaScript	Scripting language for web browsers and unrelated to Java
Python	General purpose language with readable code
PHP	Server side scripting and web app development
SQL	Relational database requests
Ruby	Web apps that use the Rails framework
NODE.js	JavaScript runtime environment for outside a browser
R	Like mathlab and used for queries
LaTeX	High level language for typesetting documents
Swift	Language for Apple applications
Excel	Spreadsheet application for data analytics

Computer programmers typically learn a number of programming languages e.g. Java, JavaScript, Python, C++, LaTeX and SQL. In addition, Programmers who specialise in web applications will also learn HTML and CSS. Python and JavaScript have fewer syntax requirements (fewer lines of code) and are good starting points for learning the logic of programming before progressing to other languages. LaTeX is a programming language used for formatting and layout of reports, papers and mathematical expressions. Examples of programming language syntax for four common languages are given below for the function 'print the characters *Hello World!* onto the computer screen'.

Python

```
print('Hello, world!')
```

LaTeX

```
\title{Hello world!}
```

C++

```
#include
Int main()
{
      std::cout << "Hello, world";
      return 0;
}
```

JavaScript

```
document.write('Hello, world!');
```

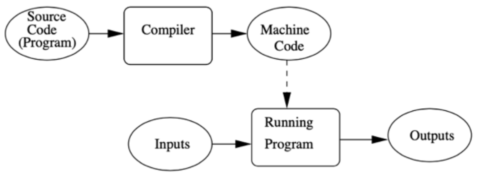
Writing Programs

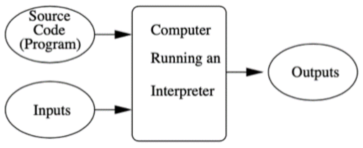
Simple text editors can be used to create programs or source code e.g. TextEdit (Mac) and Notepad (PC). Applications like Word are unsuitable because of hidden formatting that may later cause syntax errors. Python text files are saved with the .py file extension. Javascript, for example, uses .js and so on.

Integrated Development Environments (IDE) are preferred by programmers over simple text files for creating programs since they provide special features such as syntax highlighting (e.g. `print('Hello, world!')` and access to libraries of reusable source code called modules or packages. Common IDEs include:

- XCode (Apple Mac and iPhone languages)
- Visual Studio Code (Most languages)
- Android Studio (Android applications)
- RubyMine (Ruby language)
- PyCharm (Python language)

Running a program is achieved by either compiling the program into an executable (.exe) file or interpreting the source code in a text file (e.g. game.py) during computer run-time. Increasingly, programs utilise both compiled and interpreted elements. Common compiler languages include C/C++. Common interpreter languages include Python and JavaScript. Languages such as Java, C# and Python can use a combination of both. Interpreted languages can save time during program development because no compilation and linking is necessary.





Exercise

Using your PC's terminal window, create a new directory (aka folder) on your Desktop called Professional Skills. Make this folder your active directory and make new subfolders called LaTeX, Python and Excel where you can store your exercise files.

3. Python Programming

Python is one of the world's most commonly used programming languages and is also a stepping stone for learning other languages such as C++ and JavaScript. It is also the preferred language for developing artificial intelligence (AI) applications. C++ has more syntax rules and other conventions while Python imitates regular English. C++ is a good option for game development and large systems. Python is an object-oriented language and frequently used in software applications in artificial intelligence. Python has many prebuilt libraries available like scipy for advanced computing and pybrain for machine learning, making it one of the best languages for AI. Python is used in sites such as YouTube, Dropbox, Spotify, Quora and Instagram. It is one of the official languages used in Google Corporation.

Python interpreters are pre-installed in most personal computers and are typically invoked by typing "python" into the OS command prompt in the computers Terminal:

```
C:\>  python (on the PC) or
$ python (on the MAC)
>>> | (python command prompt)
```

To install the latest version of python, python3, on your computer go to python.org and follow the download and installation instructions. The command for running python3 from the OS command prompt is "python3". The exit() command exits python3 and returns the prompt to the OS:

```
$   python3
>>> |
>>> exit()
$   |
```

Python command lines can be typed directly into the python3 prompt or an entire python program (e.g. main.py) can be run from the OS prompt once the path name for the file is included e.g.

```
C:\> python3 main.py
$    python3 main.py
```

Programming

Python programs are made up of statements. Statements can include keywords, expressions and operators. Operators include symbols that make the computer to perform an action e.g. + - * /. Inputs or operands include numbers. The inputs (operands) and operators are called expressions. Examples of simple statements that can be type directly in the Python3 terminal include the following:

```
>>> 10 + 2
>>> 10 / 2 + 3
>>> 2 + 3 * 6
>>> (2 + 3) * 6
>>> 10 % 3
```

The terminal will return answers to these expressions in turn. IDEs, on the other hand, will not return answers to these expressions until the program is extended to include the print() function. Programs frequently contain three types of errors (or bugs): syntax, runtime, and semantic. Syntax errors is where something is wrong with language rules such as forgetting to place brackets or commas appropriately. Runtime errors is where the computer is unable to execute part of the code such as dividing a number by zero. Semantic errors is where the program runs perfectly but the output is not what the programmer was expecting.

Python Keywords

The following are Python keywords. Other key words that appear later such as print() are built-in functions.

Value: True, False, None
Operator: and, or, not, in, is
Control Flow: if, elif, else
Iteration: for, while, break, continue, else
Structure: def, class, with, as, pass, lambda
Returning: return, yield
Import: import, from, as
Exception-Handling: try, except, raise, finally, else, assert
Asynchronous Programming: async, await
Variable Handling: del, global, nonlocal

Print Function

The print() function outputs the specified message to the screen or other output device. The message can be a string, or other data type (e.g. integer, float or Boolean). Functions contain parenthesis () to store parameters and arguments (e.g. strings). Below, the argument "Hello World!" is passed into the print function for display on the terminal screen.

```
print("Hello world!")

print(" ______________ ")
print("|              |")
print("| Hello World!  |")
print("|               |")
print("|_______________|")

print('Hello World!')

print('Hello world "again"')

print("Hello \nWorld")                   # \n moves the word World to a new line



if 5 > 2:                                # if is a python keyword
  print("Five is greater than two!")

print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Comments

White space (spaces and empty lines) can be used liberally in most areas throughout the code to provide clarity and emphasis. Comments can be added by using the # (hash) character. The compiler ignores all characters after the # character. The # character can be used in front of lines of code to temporarily stop these lines of code from executing. Docstrings use triple single or double quotes (''' or """) at the start and end of a block of comments.

```
#This is a comment.
print("Hello, World!") #This is another comment
# print(Hello World!"  #This code is stopped from executing
"""This is a multiline docstring. Notice the triple quotes at end also."""
```

Variables

A variable is a container with a name that contains a value. The equal sign below means 'take the value on the right and store it in the variable on the left'. It does not mean that both sides are equal.

```
first_name = "David"
age = 20
is_male = True
```

"David" is a type of data called a string, 20 is called an integer and True is a Boolean data type. Other common data types include Floats (e.g. 20.5), Lists, Dictionaries and Classes (objects). Adding variables is called concatenation. Integers and Floats need to be converted to strings, using the `str()` function, before they can be concatenated.

```
print("My name is " + first_name, "and I am " + str(age) + " years old")
```

Numbers

```
print(- 4.4 + 4.5 + (3 * 5))
print(10 % 4)                   # Modula operator (i.e. remainder)

#####

num = 5
print(num)
print(abs(-5))                  # Absolute value of a number. abs() is a function
```

```
#####

x = 1
y = 2.8
z = 1j

print(type(x))                 # Verify the type of the variable
print(type(y))
print(type(z))
```

Strings

```
first_name = "mary"
print(first_name.capitalize())          # Capitalising the first letter
print(len(first_name))                  # Return length of the string
print(first_name[1])                    # Returning 2nd letter of the string
print(first_name.index("r"))            # Return index value of letter r
print(first_name.replace("ma", "an"))   # Replace a specific substring
```

Input Function

Programs can accept inputs from the user using the input() function.

```
name = input("Enter your name: ")
print("Hello " + name)
num1 = input("Enter your first number:")
num2 = input("Enter your second number:")
```

All inputs are stored as strings. These strings may need to be converted to an integer or float before using arithmetic operators.

```
result = float(num1) + float(num2)      # Convert strings to float
print("Hi " + name + " your number is: " + str(result))
```

Format Method

We can also combine strings and numbers by using the format() method that takes the arguments, formats them as variables and places them in the string placeholders {}.

```
num1 = input(float("Enter your first number:"))
num2 = input(float("Enter your second number:"))

print('{} + {} = '.format(num1, num2))
print(num1 + num2)
```

List Data Types

So far we have reviewed string, Boolean, integer and float data types. There are four data types for collection and storing multiple items (e.g. apples, pears, oranges): List, Tuple, Set and Dictionary.

List

```
names = ["John", "Tom", "Mary"]        # List labelled names
print(names)
```

List items are indexed, the first item has index [0], the second item has index [1] etc. Lists are ordered and that order will not change. The list is changeable, meaning that you can change, add, and remove items. If you add new items to a list, the new items will be placed at the end of the list. Lists allow duplicates.

```
#####

numbers = [8, 29, 12]
names.extend(numbers)                # Extend names list with numbers list
print(names)

for x in names:                      # using the for and in keywords
  print(x)
```

Functions

A function is a block of code which runs when called. You can pass data, known as arguments into the parameters defined in a function. A function can use the return keyword to return data to say a variable. Python has almost 70 built-in functions e.g. print(), input(), str(), open() etc. You can build your own Functions using the def keyword.

```
def say_hello():
    print("Hello world!")

# The function is defined above. Now you need to call the function:

say_hello()
```

Parameters and Arguments

A parameter (sometimes called **formal** parameter) is often used to refer to the variable as found in the function definition, while argument (sometimes called **actual** parameter) refers to the actual input supplied at function call.

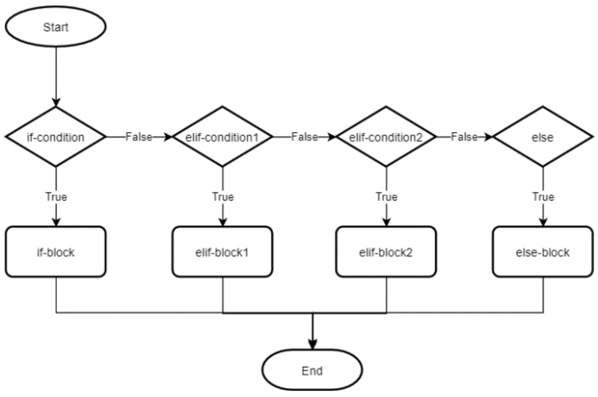
```
def call_out(name, age):       # Function with two defined parameters
    print("Hello " + name + "! You are:" + str(age))

call_out("David", "45")        # Arguments passed into the defined parameters


def my_function(fruits):                       # One defined parameter
  for x in fruits:
    print(x)

fruits = ["apple", "banana", "cherry"]         # List

my_function(fruits)                            # List items as arguments

#####

def cube(num):
    return num**3          # Return always ends the function

result = cube(4)           # Call function and pass argument 4 into num parameter

print(result)
```

If, Elif and Else Keywords

Using **if**, **elif**, **else**, **or** and **and** statements. Python also supports the usual logical conditions from mathematics:

Equals: a == b

Not Equals: a != b
Less than: a < b
Less than or equal to: a <= b
Greater than: a > b
Greater than or equal to: a >= b

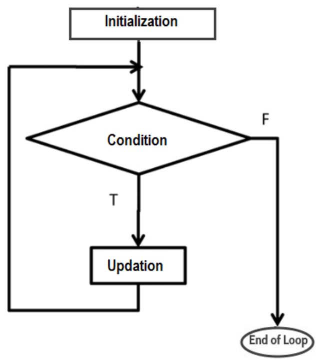


```
a = 33
b = 200
if b > a:
  print("b is greater than a")

#####

a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")

#####

a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")

#####

num1 = float(input("Enter first number: "))
op = input("Enter the operator: ")
num2 = float(input("Enter second number: "))

if op == "+":
    print("The answer is: ", num1 + num2)
elif op == "-":
    print("The answer is: ", num1 - num2)
elif op == "*":
    print("The answer is: ", num1 * num2)
elif op == "/":
    print("The answer is: ", num1 / num2)
else:
    print("Invalid operator. Run programme again")
```

Loops

WHILE Loop

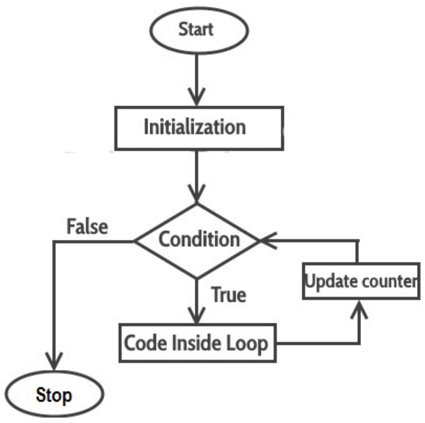
The 'while' keyword loop executes a set of statements as long as a condition is true.



```
i = 1
while i <= 10:
    print(i)
    i = i + 1   # alternatively use i += 1

print("done with loop")
```

FOR Loop

A 'for' keyword loop is used for iterating over a sequence (e.g. list).



```
friends = ["John", "Tom", "Mary"]
for name in friends:
    print(name)

#####

for index in range(10):
    print(index)

#####

def raise_to_power(base_num, pow_num):
    result = 1
    for index in range(pow_num):
        result = result * base_num
    return result
print(raise_to_power(2, 8))
```

External Files

The key function for working with files in Python is open() that takes two parameters; filename, and mode. There are four different modes: r (read), a (append), w (write) or c (create).

Reading Files

```
f = open("external_file.txt", "r")
#####
f.close()                   # Close files after use
f.closed                    # Check if file is closed (returns True if closed)
```

Writing To Files

```
f = open("external_file.txt", "a")
f.write("Mary - Reception")
f.close()
```

Classes and Objects

Python is an object oriented programming language. Almost everything in Python is an object, with its own properties and methods. A Class is like a "blueprint" or unique data type for any object e.g. mobile phone, student, university, etc.

```
class student:
   def __init__(self, name, discipline, gpa, is_female):
          self.name = name
          self.discipline = discipline
          self.gpa = gpa
          self.is_female = is_female

student1 = student("Jim", "Business", 3.1, False)     # Student object "Jim"

print(student1.gpa)
```

Modules

Python allows reuse of your own popular functions, variables, lists, etc. as modules or libraries. There are circa. 200k other public domain external modules that can also be imported into your projects from across the Python community.

```
import myown_code               # My own reusable functions, variables, etc …
import NumPy                    # Package for scientific computation
import Tkinter                  # Package for GUI
import scikit-learn             # Package for machine learning and AI
```

Python has many other prebuilt libraries available like **Scipy** for advanced computing, **Pybrain** for machine learning, **NLTK** for natural language programming and **TensorFlow** from Google, making it one of the best languages for programming AI applications.

Further Information

https://www.python.org/

Examples, Quizzes and Exercises:
Try out these examples, quizzes and exercises as you progress.
https://www.w3schools.com/python/

Exercise

Copy and improve the following simple Calculator Program where the user chooses a mathematical operator and two numbers to operate on. Consider adding a welcome function at the top of the program's code with a simple text based welcome banner and name your calculator. Add more error-handling, for example to ensure that the program continues to run even if the user types text or float when asked for a number. Add exception handling for when the user selects the division operator and types in 0 for their second number. Add additional mathematical operators i.e. `**` and `%`. Enhance code in other ways.

```
def calculate():
    operation = input('''
Please type in the math operation you would like to complete:
+ for addition
- for subtraction
* for multiplication
/ for division
''')

    num_1 = int(input('Please enter the first number: '))
    num_2 = int(input('Please enter the second number: '))

    if operation == '+':
        print('{} + {} = '.format(num_1, num_2))
        print(num_1 + num_2)

    elif operation == '-':
        print('{} - {} = '.format(num_1, num_2))
        print(num_1 - num_2)

    elif operation == '*':
        print('{} * {} = '.format(num_1, num_2))
        print(num_1 * num_2)

    elif operation == '/':
        print('{} / {} = '.format(num_1, num_2))
        print(num_1 / num_2)

    else:
        print('You have not typed a valid operator, please run the program again.')

    # Add again() function to calculate() function
    again()

def again():
    calc_again = input('''
Do you want to calculate again?
Please type Y for YES or N for NO.
''')

    if calc_again.upper() == 'Y':
        calculate()
    elif calc_again.upper() == 'N':
        print('See you later.')
    else:
        again()

calculate()
```

4. LaTeX Programming

LaTeX (pronounced LAY-tek) is a higher layer of the TeX programming language specifically designed to create professionally formatted documents that include complex mathematical expressions. LaTeX can be installed on your PC but most users now use cloud based applications e.g. www.overleaf.com (free). Overleaf gives instant access to the LaTeX programming language and has an extensive help environment. A LaTeX program has two parts: the **Preamble** containing packages or modules of Tex code and the **Body**. All keyword commands begin with a backslash \ and parameters or arguments are passed using curly brackets {}. Comments can be added using the % character. The program example below contains two statements in the Preamble and three in the Body:

```
\documentclass[a4paper,12pt]{article}     % Preamble section
\title{Hello world!}

\begin{document}                          % Body section
\maketitle
\end{document}
```

Preamble

The preamble defines the type of document you are writing, the spoken language you are writing in and the packages (modules) of Tex code you would like to use. Keywords are case sensitive e.g. writing "Documentclass" below will return a syntax error.

```
\documentclass{article}              % package for formatting articles
\usepackage[utf8]{inputenc}          % package for character encoding
\usepackage[margin=25mm]{geometry}   % package for formatting margins
\usepackage{natbib}                  % …formatting citations and bibliographies
\usepackage{graphicx}                % …formatting and numbering figures
\usepackage{amsmath}                 % …formatting equations
\usepackage{xcolor}                  % …text colours red, green, etc

\title{Report Title}
\author{Connor Adams, ID: 20379631}
\date{October 2020}
```

Body

The body of the program contains all of the printable text in addition to various formatting commands.

```
\begin{document}                     % begin the printable document
\maketitle                           % place title here (title, author, date,…)
\tableofcontents                     % place "table of contents" here
\pagebreak                           % place a page break here
```

Sections

Sections and subsections are very easy to create.

```
\section{First level section}
\subsection{Second level section}
\subsubsection{Third level section}
\section*{Unnumbered section}
\paragraph{...}
\subparagraph{...}
```

Sections can be labelled so that they can be referred to in the text.

```
\section{This is first section}
\label{sec1}
```

This label can now be referred to in the text using the ref{} and pageref{} commands as `\ref{sec1}` on page `\pageref{sec1}.`

Font Effects

Bold, italics, underlines etc. are created using the following commands. The \color command reguires \usepackage{xcolor} in the preamble.

```
These are \textit{words in italics}.
These are \textbf{words in bold}.
These are \textsf{sans serif words}.
These are \textrm{roman words}.
These are \underline{underlined words}.
These are \textbf{\textit{words in bold and italics}.
These are {\color{red}red coloured words}.
```

Lists

List are easy to create. They start with a `\begin{...}` command and end with the `\end{...}` command.

```
\begin{itemize}
\item Item
\item Item
\end{itemize}

%%%%%%%%%

\begin{enumerate}
\item Item 1
\item Item 2
\end{enumerate}
```

Equations

Equations can be inserted in either of two modes – inline and displayed. The displayed mode has two versions – numbered and un-numbered. Many math mode commands require the module `\usepackage{amsmath}` in the preamble.

To put your equations in *inline* mode use one of these delimiters: \(\) or $ $ or \begin{math} \end{math} For example $E=mc^2$.

The displayed mode has two versions: numbered and unnumbered.

```
The mass-energy equivalence is described by the famous equation:

\[ E=mc^2 \]

discovered in 1905 by Albert Einstein.
In natural units ($c = 1$), the formula expresses the identity:

\begin{equation}
E=m
\end{equation}
```

Verbatim

The `\begin{verbatim}` command prints text in monospaced font and prints spaces, tabs, etc. verbatim. This can be used for displaying code snippets:

```
\begin{verbatim}

#include
Int main()
    {
        std::cout << "Hello, world";
        return 0;
    }

\end{verbatim}
```

Tables

The tabular command is used to typeset tables. This is an example of a table and referral in the text to the table.

```
\begin{table}[h!]
\centering                        % centre the table on the page
\begin{tabular}{|c r l|}          % c = cjt column, l = ljt column, etc.
\hline                            % insert a horizontal line
Col1 & Col2 & Col2 \\
\hline
A1 & B1 & C1 \\                   % A1, A2, etc. are data
A2 & B2 & C2 \\
A3 & B3 & C3 \\
\hline
\end{tabular}
\caption{Table Caption}
\label{table:example}
\end{table}

Table \ref{table:example} is an example of referred table.
```

Figures

Figures can be included and automatically numbered sequentially and centre justified. Figures are stored outside the main.tex file. Images should be PDF, PNG, JPEG or GIF files. The `\usepackage{graphicx}` needs to be called within the preamble:

```
\usepackage{graphicx}

%%%%%%%%%

Insert the figure here:

\begin{figure}[h]                       % Insert the figure about here
\centering                              % This centres the image
\includegraphics[scale=2]{myimage}
\caption{My Image}
\label{fig:myimage}                     % Figures to be referred to in the text
\end{figure}
```

Figures are cited within the body text as follows:

```
Refer to figure \ref{fig:myimage} above and on the following page
\pageref{fig:myimage}
```

Bibliography

References are includes using the major bibliography management programs (packages): biblatex, natbib and bibtex. Bibliography is a list of references cited throughout the text with a References list placed at the end of the report. Bibliographies can be embedding at the end of your document as follows:

```
\begin{thebibliography}{9}

\bibitem{latexcompanion}
Michel Goossens, Frank Mittelbach, and Alexander Samarin.
\textit{The \LaTeX\ Companion}.
Addison-Wesley, Reading, Massachusetts, 1993.

\bibitem{knuthwebsite}
Knuth: Computers and Typesetting,
\\\texttt{http://www-cs-faculty.stanford.edu.html}

\end{thebibliography}
```

References are cited in text as follows:

```
This is a reference \cite{knuthwebsite} cited in the text
```

Further Information

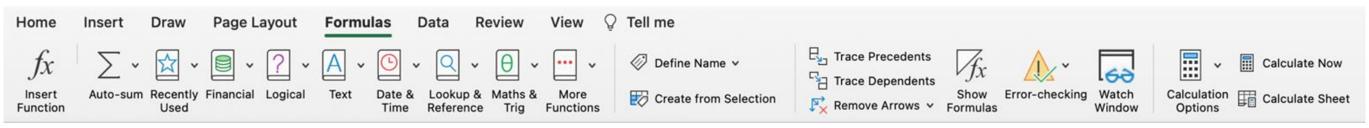
https://www.latex-project.org/
https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes

Exercise

Create an account in Overleaf.com or equivalent. Create a new document and practice the various commands above.

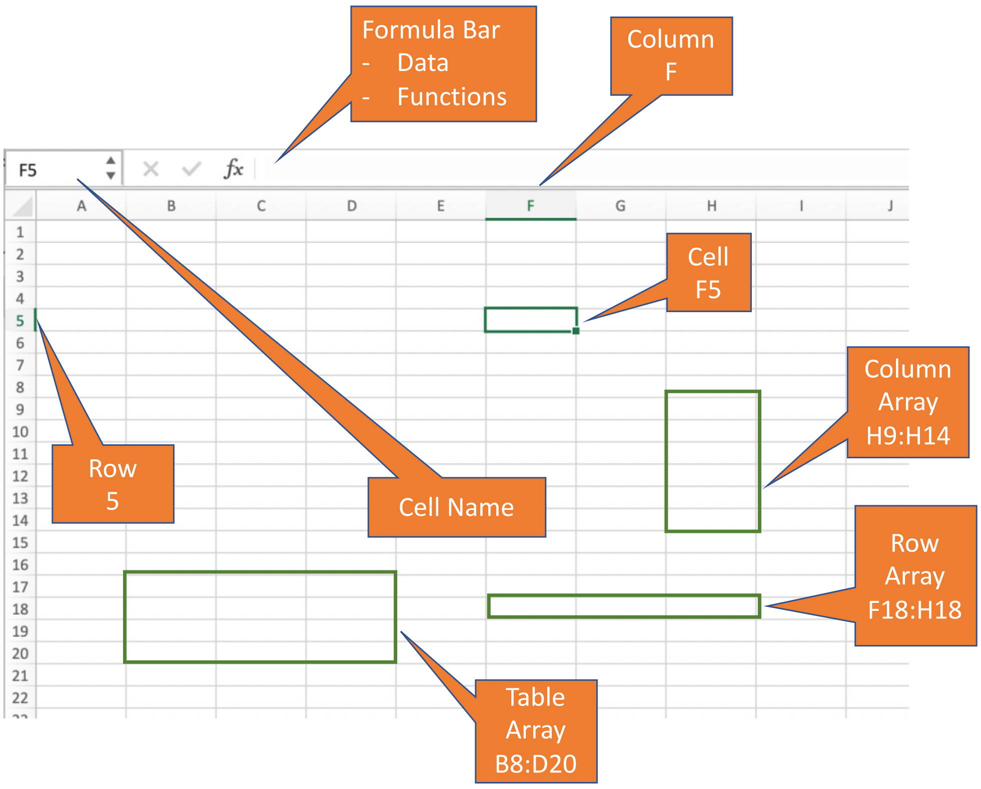
5. Excel Programming

Microsoft excel is the world's most popular application for data presentation and analysis and also a popular easy-to-use database. Excel is presented to the user as a tabular interface consisting of rows and columns of cells. Cells store data as either values (types) or formulas (functions). The menus below illustrate the large number of commands available. One of the most important menus is Formulas that offers over 450 functions.



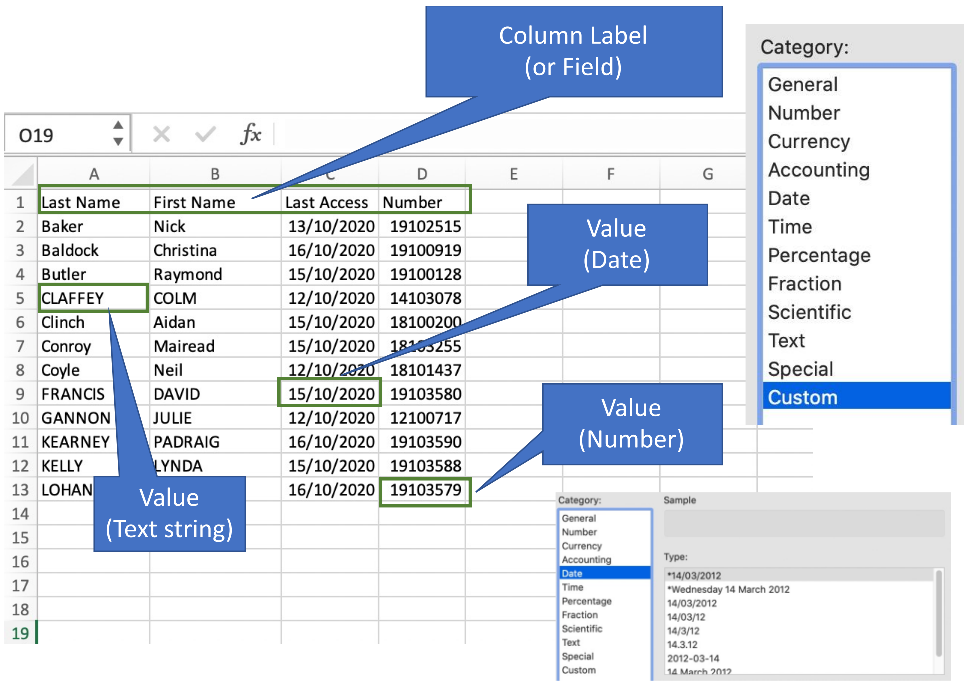
A single worksheet is illustrated below. Key features include:

- Columns
- Rows
- Cells
- Formula Bar
- Arrays – Column, Row and Table



Data tables are typically presented as illustrated below with key features including:

- Column Labels (or Fieldnames)
- Cell Values or types (Numbers, Text, Dates, etc)



	A	B	C	D
1	Last Name	First Name	Last Access	Number
2	Baker	Nick	13/10/2020	19102515
3	Baldock	Christina	16/10/2020	19100919
4	Butler	Raymond	15/10/2020	19100128
5	CLAFFEY	COLM	12/10/2020	14103078
6	Clinch	Aidan	15/10/2020	18100200
7	Conroy	Mairead	15/10/2020	18103255
8	Coyle	Neil	12/10/2020	18101437
9	FRANCIS	DAVID	15/10/2020	19103580
10	GANNON	JULIE	12/10/2020	12100717
11	KEARNEY	PADRAIG	16/10/2020	19103590
12	KELLY	LYNDA	15/10/2020	19103588
13	LOHAN		16/10/2020	19103579

```
*14/03/2012
*Wednesday 14 March 2012
14/03/2012
14/03/12
14/3/12
14.3.12
2012-03-14
14 March 2012
```

Excel Functions

All common mathematical operators can be used in formulas e.g.

```
=A1+A2-(B3*D4)^2/G2
```

COUNT Functions

```
=COUNT(D5:D12)            counts all numerical values
=COUNTA(D5:D12)           counts all values
=COUNTIF(D5:D12,">=21")
```

SUM and AVERAGE Functions

```
=SUM(D4:D14)
=AVERAGE(D4:D14)
=SUMIF(F4:F14, "f", D4:D14)   =SUMIF(range,criteria,sum_range)
```

XLOOKUP Function

```
=XLOOKUP(H24,E3:E14,C3:C14)
XLOOKUP(lookup_value,lookup_array,return_array,if_not_found,match_mode,search_mode)
```

IF and AND Functions

AND checks whether all arguments are TRUE, and returns TRUE value if all arguments are TRUE else FALSE

```
=IF(AND(D5>=D7,D5<=D8),D10,D11)
```

CONCATENATE Function

```
=A1&" "&B1&" "&C1                    or
=CONCATENATE(A1," ",B1," ",C1)
```

Absolute, Relative, and Mixed Cell References

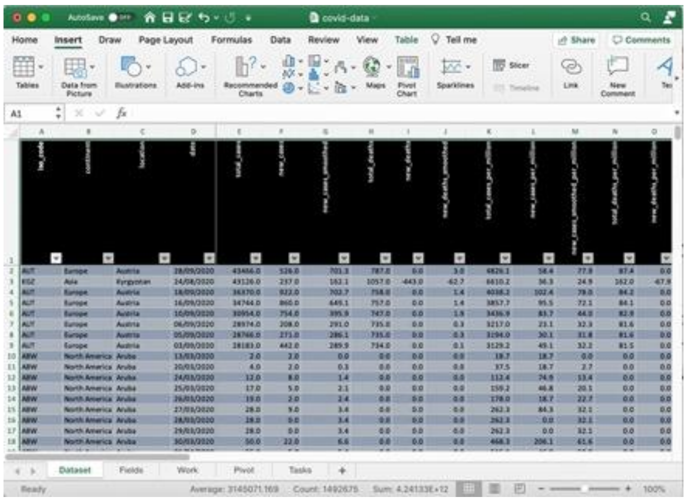
When a Formula/Function is copied or moved the cell(s) that it refers to may also change:

```
A1 – column and row may change
$A1 – always refers to column A but row may change
A$1 – always refers to row 1 but column may change
$A$1 – always refers to A1
```

Tables and Autofilter

Turn a range of cells into a fixed "Table" to manage and analyse large groups of data:

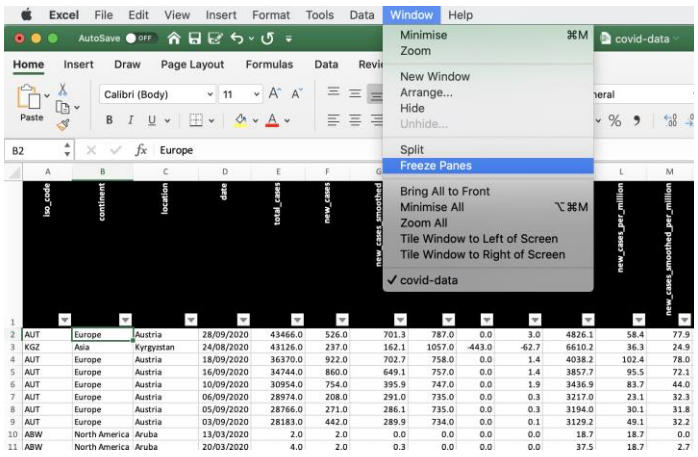
- Select cell range
- Insert "Table"



	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million	new_cases_per_million	new_cases_smoothed_per_million	total_deaths_per_million	new_deaths_per_million
2	AUT	Europe	Austria	28/09/2020	43466.0	526.0	701.3	787.0	0.0	3.0	4826.1	58.4	77.9	87.4	0.0
3	KGZ	Asia	Kyrgyzstan	24/08/2020	43126.0	237.0	162.1	1057.0	-443.0	-62.7	6610.2	36.3	24.9	162.0	-67.9
4	AUT	Europe	Austria	18/09/2020	36370.0	922.0	702.7	758.0	0.0	1.4	4038.2	102.4	78.0	84.2	0.0
5	AUT	Europe	Austria	16/09/2020	34744.0	860.0	649.1	757.0	0.0	1.4	3857.7	95.5	72.1	84.1	0.0
6	AUT	Europe	Austria	10/09/2020	30954.0	754.0	395.9	747.0	0.0	1.9	3436.9	83.7	44.0	82.9	0.0
7	AUT	Europe	Austria	06/09/2020	28974.0	208.0	291.0	735.0	0.0	0.3	3217.0	23.1	32.3	81.6	0.0
8	AUT	Europe	Austria	05/09/2020	28766.0	271.0	286.1	735.0	0.0	0.3	3194.0	30.1	31.8	81.6	0.0
9	AUT	Europe	Austria	03/09/2020	28183.0	442.0	289.9	734.0	0.0	0.1	3129.2	49.1	32.2	81.5	0.0
10	ABW	North America	Aruba	13/03/2020	2.0	2.0	0.0	0.0	0.0	0.0	18.7	18.7	0.0	0.0	0.0
11	ABW	North America	Aruba	20/03/2020	4.0	2.0	0.3	0.0	0.0	0.0	37.5	18.7	2.7	0.0	0.0
12	ABW	North America	Aruba	24/03/2020	12.0	8.0	1.4	0.0	0.0	0.0	112.4	74.9	13.4	0.0	0.0
13	ABW	North America	Aruba	25/03/2020	17.0	5.0	2.1	0.0	0.0	0.0	159.2	46.8	20.1	0.0	0.0
14	ABW	North America	Aruba	26/03/2020	19.0	2.0	2.4	0.0	0.0	0.0	178.0	18.7	22.7	0.0	0.0
15	ABW	North America	Aruba	27/03/2020	28.0	9.0	3.4	0.0	0.0	0.0	262.3	84.3	32.1	0.0	0.0
16	ABW	North America	Aruba	28/03/2020	28.0	0.0	3.4	0.0	0.0	0.0	262.3	0.0	32.1	0.0	0.0
17	ABW	North America	Aruba	29/03/2020	28.0	0.0	3.4	0.0	0.0	0.0	262.3	0.0	32.1	0.0	0.0
18	ABW	North America	Aruba	30/03/2020	50.0	22.0	6.6	0.0	0.0	0.0	468.3	206.1	61.6	0.0	0.0

Freeze Panes

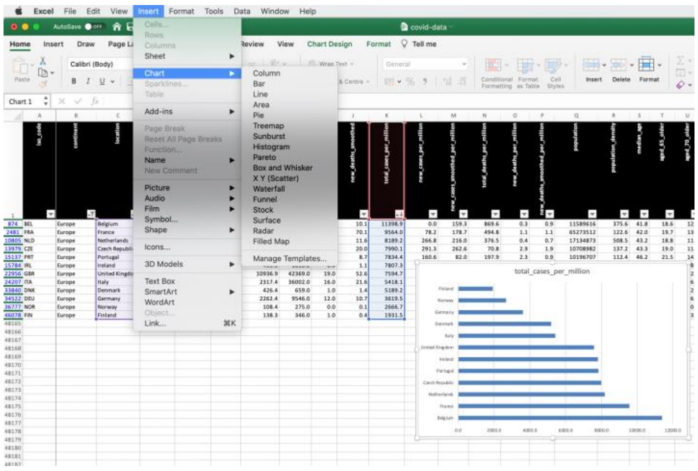
Stop specific rows and/or columns from scrolling



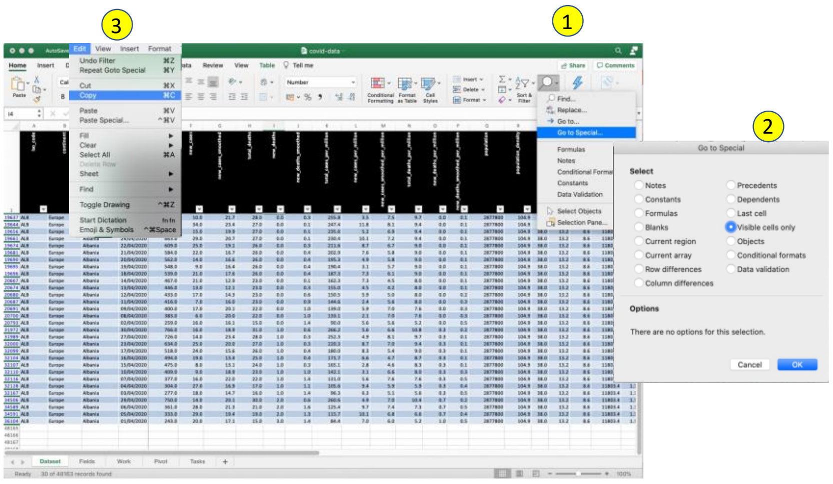
	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed					new_cases_per_million	new_cases_smoothed_per_million
2	AUT	Europe	Austria	28/09/2020	43466.0	526.0	701.3	787.0	0.0	3.0	4826.1	58.4	77.9
3	KGZ	Asia	Kyrgyzstan	24/08/2020	43126.0	237.0	162.1	1057.0	-443.0	-62.7	6610.2	36.3	24.9
4	AUT	Europe	Austria	18/09/2020	36370.0	922.0	702.7	758.0	0.0	1.4	4038.2	102.4	78.0
5	AUT	Europe	Austria	16/09/2020	34744.0	860.0	649.1	757.0	0.0	1.4	3857.7	95.5	72.1
6	AUT	Europe	Austria	10/09/2020	30954.0	754.0	395.9	747.0	0.0	1.9	3436.9	83.7	44.0
7	AUT	Europe	Austria	06/09/2020	28974.0	208.0	291.0	735.0	0.0	0.3	3217.0	23.1	32.3
8	AUT	Europe	Austria	05/09/2020	28766.0	271.0	286.1	735.0	0.0	0.3	3194.0	30.1	31.8
9	AUT	Europe	Austria	03/09/2020	28183.0	442.0	289.9	734.0	0.0	0.1	3129.2	49.1	32.2
10	ABW	North America	Aruba	13/03/2020	2.0	2.0	0.0	0.0	0.0	0.0	18.7	18.7	0.0
11	ABW	North America	Aruba	20/03/2020	4.0	2.0	0.3	0.0	0.0	0.0	37.5	18.7	2.7

Charts

Generate wide variety of charts from data.



Copy Visible Cells



Go to Special

Select

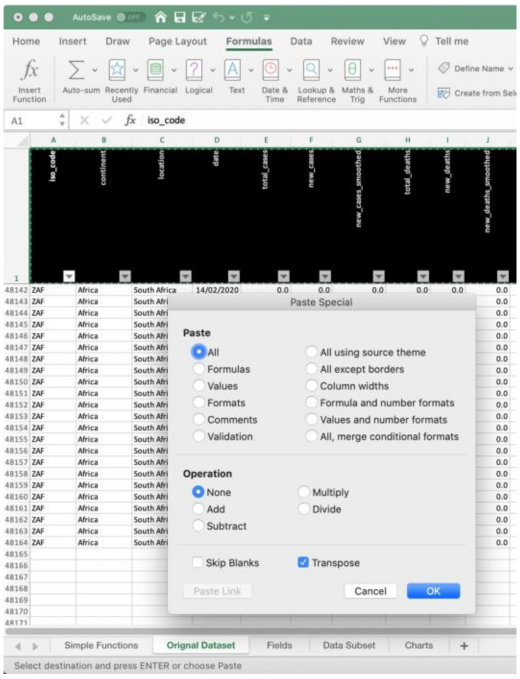
- Notes
- Constants
- Formulas
- Blanks
- Current region
- Current array
- Row differences
- Column differences
- Precedents
- Dependents
- Last cell
- [✓ Visible cells only]
- Objects
- Conditional formats
- Data validation

Options

There are no options for this selection.

Cancel OK

Paste Special > Transpose



Paste Special

Paste

[✓ All] All using source theme
Formulas All except borders
Values Column widths
Formats Formula and number formats
Comments Values and number formats
Validation All, merge conditional formats

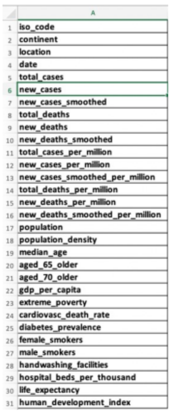
Operation

[✓ None] Multiply
Add Divide
Subtract

Skip Blanks [✓ Transpose]

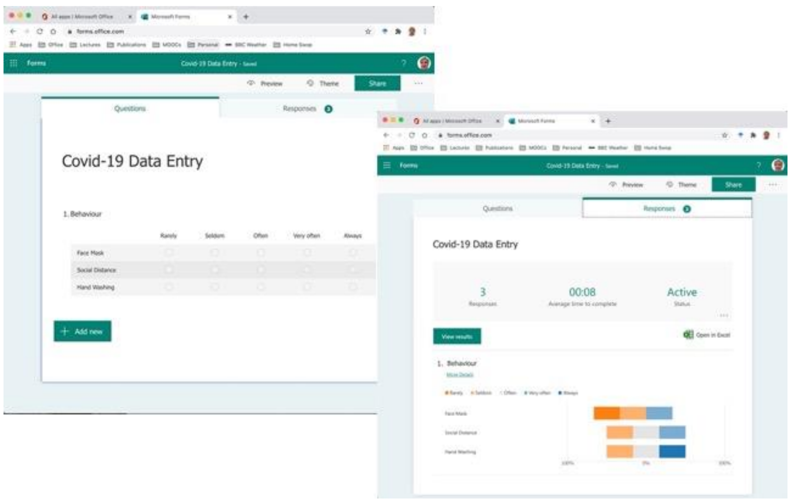
Paste Link Cancel OK

Select destination and press ENTER or choose Paste



	A
1	iso_code
2	continent
3	location
4	date
5	total_cases
6	new_cases
7	new_cases_smoothed
8	total_deaths
9	new_deaths
10	new_deaths_smoothed
11	total_cases_per_million
12	new_cases_per_million
13	new_cases_smoothed_per_million
14	total_deaths_per_million
15	new_deaths_per_million
16	new_deaths_smoothed_per_million
17	population
18	population_density
19	median_age
20	aged_65_older
21	aged_70_older
22	gdp_per_capita
23	extreme_poverty
24	cardiovasc_death_rate
25	diabetes_prevalence
26	female_smokers
27	male_smokers
28	handwashing_facilities
29	hospital_beds_per_thousand
30	life_expectancy
31	human_development_index

Data Entry (using MS Forms)



Covid-19 Data Entry

1. Behaviour

	Rarely	Seldom	Often	Very often	Always
Face Mask					
Social Distance					
Hand Washing					

Covid-19 Data Entry

3 Responses; 00:08 Average time to complete; Active Status

Data Analytics

Data analytics is a technique used in Artificial Intelligence that typically operated on large tables of data. A typical process includes:

- Define Research Hypothesis e.g. smoking and age are key factors in COVID-19 death rates
- Define Measurement Indicators e.g. COVID death rates, smoking habits, age profiles
- Collect Data
- Clean and Analyse Data
- Interpret Results

Solver

Open the RoboProject spreadsheet and use Excel Solver add-in to test the three questions posed using the constraint, variable and objective cells highlighted.

RoboProject

Eight major projects are identified A to H. €300k is available for development and 40 developers.

Project	A	B	C	D	E	F	G	H	Required	Available	Remaining
Cost	60	110	53	47	92	85	73	65	0	300	300
Developers	7	9	8	4	7	6	8	5	0	40	40
Benefit	36	82	29	16	56	61	48	41		0	
Select? (0 = no, 1 = yes)	0	0	0	0	0	0	0	0			

⬜	*Fixed Values - Inputs (A)*
🟥	*Constrained cell*
🟧	*Variable (changing) cells*
🟦	*Objective (target) cell - Output (B)*
🟪	*Calculated cells*

Problems
(a) Ask Solver which projects should be selected that maximise forecast benefit?
(b) Ask Solver which projects should be slected if Projects B and E must be developed together?
(c) Ask Solver which project should be selected if projects E and H must be included?

Exercise

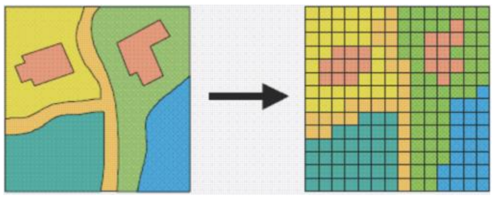
Visit 'Google Dataset Search' and search for 'coronavirus covid-19'.
Download the .xlsx file from https://ourworldindata.org/coronavirus-data

Complete the Following:

- Open the dataset in Excel
- Create a TABLE or add the Auto-filter
- Clean Data
- Analyse Data
- Which country in Europe has the second highest GDP?
- Which country in the world has the highest human_development_index?
- Which country in South America has the highest_population_density?
- Among top ten highest GDP countries in Europe on the 16/10/2020, where does Ireland rank in terms of new_cases_smoothed_per_million?
- What is the Excel formula for positive_rate (place formula between brackets)?
- On what date did Ireland's highest positive_rate occur?
- Insert a chart below of the new_cases_per_million ranked from lowest to highest for the subset of countries given
- Suggest a label for one new data field to add to the dataset
- Suggest one new function/formula to add to the dataset

Appendix: Graphics

There are two popular graphics formats for images, diagrams and pictures – vector and raster. These are illustrated in the figure below.



Vector Raster

Vector is accurate, scalable but slower to load on a web page or document. Raster is faster to load but the image can become grainy when enlarged. The vector models produced in say PowerPoint use mathematically based point and line segments to draw images while the raster model uses a series of cells or pixels to represent images. Raster images can come in a number of formats including .png, .bmp, .pdf, and .gif

Exercise

Create a vector image in PowerPoint for your writing project e.g. the figures shown earlier for 'If statements' and 'Loops'. When your vector image is fully complete create a raster image using the .png format and insert it in the appropriate location in your written report.

To create a raster image in PowerPoint select the vector image and all associated text that you have created and then right-click 'Save as Picture'. Choose raster image format and then save the raster image file in your files for later insertion into your report.

Appendix: LaTeX Template

Copy and paste the following code into a blank main.tex LaTeX file to help make a start on your own 'Programming Skills' research report:

```
\documentclass[12pt]{article} %formatting articles and reports
\usepackage[utf8]{inputenc} %character encoding
\usepackage[margin=25mm]{geometry} %formatting margins
%\usepackage{graphicx} %formatting and numbering figures
\usepackage{xcolor}
\usepackage[nottoc]{tocbibind} %add bibliography to ToC

\title{Programming Skills}
\author{Your Name, ID:12345678}
\date{November 2021}

\begin{document}

\maketitle

\tableofcontents
\newpage

\section{Introduction}

Use all your own words e.g.: The purpose of this report is to explain some of the basics of computer programming … etc.

\subsection{Operating Systems}

Operating systems are …

The OS command prompt looks like this:

\begin{verbatim}
C:\>  word.exe
\end{verbatim}

Common OS keywords are given in Table \ref{table:commands}:

\begin{table}[htbp]
\centering
\caption{Common OS Keywords}

\begin{tabular}{|l|c|r|}
\hline
Mac OS & Description & Windows OS \\
\hline
A1 & B1 & B3 \\
A2 & B2 & B2 \\
A3 & B3 & B3 \\
\hline
\end{tabular}
\label{table:commands}
\end{table}

\subsection{Programming Languages}
Computer programmers typically learn a number of programming languages e.g. Java, JavaScript, Python, C++, LaTeX and SQL. In addition ....

\begin{verbatim}

Python
print(`Hello, world')

LaTeX
\title{Hello world!}

C++
#include
Int main()
{
    std::cout << "Hello, world";
    return 0;
}

JavaScript
document.write(`Hello, world!');

\end{verbatim}

\subsection{Writing Programs}

Common IDEs include:

\begin{itemize}
\itemsep0em           % Separation distance between items
\item XCode (Apple Mac and iPhone languages)
\item Visual Studio Code (Most languages)
\item Android Studio (Android applications)
\item RubyMine (Ruby language)
\item PyCharm (Python language)
```

\end{itemize}

Running a program is achieved by either compiling the program into an executable (.exe) file or interpreting the source code (e.g. game.py) text file during computer run-time....

\section{Python Programming}

Introduction text explaining how the various keywords, functions in the following program work. The calculate() function … etc.

\begin{verbatim}

```
def calculate():
    operation = input('''
    Please type in the math operation you would like to complete:
    + for addition
    - for subtraction
    * for multiplication
    / for division
    ''')

    num_1 = int(input('Please enter the first number: '))
    num_2 = int(input('Please enter the second number: '))

    if operation == '+':
        print('{} + {} = '.format(num_1, num_2))
        print(num_1 + num_2)

    elif operation == '-':
        print('{} - {} = '.format(num_1, num_2))
        print(num_1 - num_2)

    elif operation == '*':
        print('{} * {} = '.format(num_1, num_2))
        print(num_1 * num_2)

    elif operation == '/':
        print('{} / {} = '.format(num_1, num_2))
        print(num_1 / num_2)

    else:
        print('Not a valid operator, run program again.')

    # Add again() function to calculate() function
    again()

def again():
    calc_again = input('''
    Do you want to calculate again?
    Please type Y for YES or N for NO.
    ''')

    if calc_again.upper() == 'Y':
        calculate()
    elif calc_again.upper() == 'N':
        print('See you later.')
    else:
        again()

calculate()
```

\end{verbatim}

\section{Conclusions}

Summary of your report...

\end{document}