Contents

Exercise 3 – Key Internet Technologies

Objective

To understand the role and purpose of internet technologies.

Task

In a small group, work through the following list between you and write down the purpose of the following pieces of technology, in particular

- Whether they are relevant to clients (web browsers)
- Servers
- Or both

Cover at least one item from every set and search the web for any you don't know.

The purpose of this exercise is to introduce you to some terms and technologies that will crop up on the course. You are not expected to have perfect answers just yet!

Apache, MySQL, nGinX, IIS, Flask

HTML, CSS, Javascript, Python, Java

Chrome, Firefox, Microsoft Edge, Lynx

DNS, HTTP, TCP, HTTP Request Methods

Exercise 4 – HTML

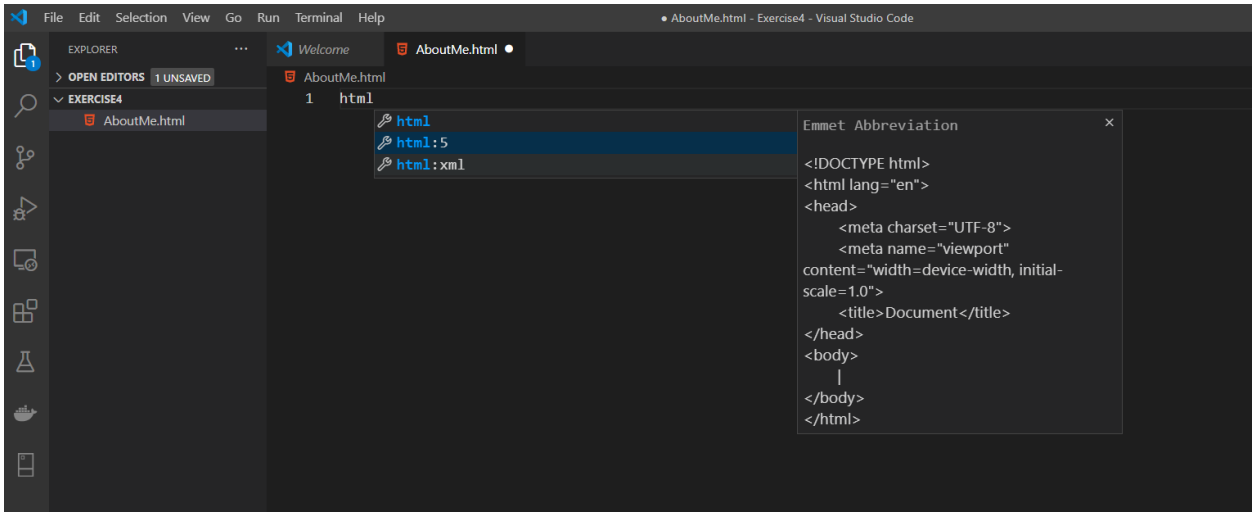
Objective

- To create HTML pages within VS Code
- To use basic tags such as <h1> and <p> tags
- To practise with hyperlinks, images, lists and tables
- To use the HTML5 structural tags
- To annotate your page with HTML comments
- To use character entities

Tasks

Task A: Create a File in VS Code

1. Create a new folder on your machine called **Exercise4** (no spaces).
2. Open VS Code and open the **Exercise4** folder.
3. Create an html file in VS Code within this folder called **AboutMe.html**
4. In the new empty file, type html to see a list of abbreviations. Select the *html:5* abbreviation and hit the *tab* key to generate a HTML 5 page.



5. Change the title to 'About Me' in the <head> section.
6. Use the HTML5 structural tags (section, header, article, footer) plus heading and paragraph tags as appropriate, and write a small piece of content about yourself.
7. Add a folder to the Exercise4 folder called **images**. Put some image files into this folder and include at least one image in the AboutMe.html page.
8. Test the file as you work by opening the file in a browser.

You have not styled your page so it may not be aesthetically pleasing. You will style it in a later exercise. Right now, it is the content and its semantic meaning that is important.

Task B: Create a HTML Table

1. Add a second page to your project called **Favourites.html**
2. Create an HTML **table** in this page and list your top five favourite things, such as your favourite book, band or food. Add at least one **link** to an external resource (e.g. to a Wikipedia page for one of your favourite items). Investigate how you can get this page to open in a separate browser tab.
3. Add a **footer** to this page and add some **copyright** information to it.
4. Put some **comments** in your HTML to explain what the table tags mean.

Task C: Create navigation links

Use a **nav** tag combined with an unordered **list** and **anchor** tags to create a navigation section to link your two pages together.

Exercise 5A – CSS

Objective

- To style HTML pages with CSS
- To use an external style sheet plus internal styles

Tasks

Task A: Create a Folder and copy the starter files

1. Create a new folder called **Exercise5**.
2. In File Explorer / Finder, copy the starter file **AboutLisa.html** and the **images** folder into your **Exercise5** folder.
3. Open the **Exercise5** folder in *VS Code* and examine the contents of the **images** folder and the **AboutLisa.html** file.
4. Open the **AboutLisa.html** file in a *browser* and familiarise yourself with how it looks.
5. In VS Code, add a Cascading Style Sheet file to your folder called **styles.css**.
6. Specify the following style rule within your file:

```
body {
    font: verdana, sans-serif;
    font-size: 20px;
    background-color: lightgrey;
    padding: 20px;
    color: navy;
    text-align: center;
}
```

7. Add a link in the <head> section of your HTML page to your **styles.css** file:

```
    <link href="styles.css" rel="stylesheet" type="text/css" />
```

8. Save all of your files and refresh your browser to view the web page and note any changes.

Task B: Create more style rules

1. To improve the **navigation** section of the page, create a rule in **styles.css** to left align the text:

```
nav {text-align: left;}
```

2. To emphasize the **footer**, create a style rule that contains the following:

```
background-color: dimgray;
            color: white;
            font-size: 16px;
```

3. Save your files and refresh the page in your browser to test.

Task C: Create internal styles

1. You want the <h3> heading of "Who Am I?" to have its own specific style. Add an **id** attribute to the heading with a value of **MainTitle**.
2. Create an internal style block within the <head> section of the page. Create a style rule for the tag with an id of MainTitle that contains the following:

```
color: maroon;
line-height: 2;
font-size: 30px;
```

3. You want to make the building images on this page smaller than the viewport width so create an internal style rule for all <**img**> tags. This rule should specify that images are block-level display elements (they are normally inline), plus automatic margins so the browser balances the margins either side and finally, a width of 60%:

```
display: block;
margin: auto;
width: 60%;
```

4. Save your file and refresh your web page in the browser. The building images should look good; however, the image of Lisa is now too big.
5. As you will have some images that need a smaller percentage size you decide to create a style class rule called **smallimg**. Create this style rule and set the **width** property to **20%.**
6. Add the **class** attribute to the Lisa image and set its value to **smallimg**.
7. Save your files and refresh the page to test.

Task D: Add CSS to your Exercise4 folder

Practice creating styles for the **AboutMe** page in your **Exercise4** folder.

Exercise 5B – CSS

Objective

- To style HTML pages with Bootstrap
- To import a Google Font
- To understand responsive web design

Tasks

Task A: Style a page using Bootstrap

1. In VS Code, open your **Exercise4** folder.
2. Add all the necessary declarations, meta tags and links for Bootstrap to your **Favourites** web page.
3. Research how to style a table using Bootstrap and implement what you learnt into your favourites table. **Tip**: Don't forget the container.

Task B: Import a Font

1. Browse Google Fonts and choose a font for your Favourites page.
2. Create an external style sheet and import the font.
3. Use the new font within your style rules e.g. the body{…} style rule.
4. Link to the style sheet from your web page.

Task C: Examine a responsive application

1. In File Explorer / Finder add **Ex5B_Lisa_Bootstrap.html** and **Lisa_styles.css** to your **Exercise5** folder.
2. Browse to the file in **Chrome** to test.
3. Open the Chrome Developer tools (**F12**).
4. Toggle the Device Bar (**Ctrl + Shift + M**). Adjust the size of the screen to see the effects on the layout of the web page. Pay particular attention to the navigation menu at the top and how the images are stacked or horizontal.
5. In VS Code, explore the markup and Bootstrap classes that have been applied to the document. Notice the different **containers** in the document. Investigate the **navigation** section and notice there are two versions of the navigation area: one for when the viewport is wide enough to show the menu as text and another 'hamburger' button. Also investigate the **search** button and how it was styled with an **icon** plus the **rounded corners** of the image.
6. Edit **Lisa_styles.css** to customise the page to your own individual style. If possible, use a colour palette chooser to help you find complimentary colours.

Exercise 8 – Introduction to Python 3

Objectives

- To become familiar with PyCharm
- To declare and output variables

Tasks

Part 1: Practice declaring variables

1. Create a new PyCharm project: **Exercise8**.
2. Add a new Python file to the project called **say_my_name.py**
3. Within this file, create two variables: one containing your first name and another containing your last name.
4. Use the print function to output your full name with a space between your first and last name.

Part 2: Practice with a more complex script

1. Create a new PyCharm file in the same project called **anatomy.py**.
2. Type the code from the slide titled "Anatomy of a Python Program" into this file
3. Run the file using the PyCharm run button
4. Run the file from the PyCharm terminal and pass different arguments to the script

Exercise 9 – Variables

Objectives

- To practice with variables and types
- To use the input function
- To become familiar with some built-in functions and methods

Tasks

Part 1: Practice declaring variables and using different types

1. Create a new PyCharm project: **Exercise9**.
2. Add a new Python file to the project called **types.py**
3. Within this file, create two variables: one containing your first name and another containing your last name.
4. Use the print function to output your full name with a space between your first and last name.
5. Now transfer these variable values into a *list* and display the list.
6. Take the variables and now store the values in a *dictionary*, using the keys 'first' and 'last'. Display the dictionary values.

Part 2: Practice with methods and functions

1. Create a new PyCharm file in the same project called **operations.py**.
2. Type the following code:

```
var = input("Please enter a value: ")
```

This is an easy way of outputting a prompt to the console and getting a reply. The variable **`var`** is a reference to that reply, which is a *string*.

3. Now print the following:

a) The value of `var` as upper case.

b) The number of characters in `var` (this does not require a method).

c) Does it contain numeric characters? (try the `isdecimal()` method).

Part 3: Practice with complex maths

1. Create a new PyCharm file in the same project called **projectile.py**.

The height of a projectile (y) from a gun (ignoring air resistance) is given as:

$$y = y_0 + x \tan\theta - \frac{gx^2}{2(v_0 \cos\theta)^2}$$

where:

g : Acceleration due to gravity: 9.81 m/s squared

v_0 : the initial velocity m/s

θ : (theta) elevation angle in radians

x : the horizontal distance travelled

y_0 : height of the barrel (m)

Write a Python program to answer the following question:

At a barrel height of 1m, after a horizontal distance of 0.5m, an elevation of 80 degrees, and an initial velocity of 44 m/s, what is the height of the projectile?

To convert degrees (`deg`) to radians use:

```
theta = deg * (pi/180)
```

You will need to import some math methods:

```
from math import pi, tan, cos
```

Exercise 10 – Conditionals

Objectives

- To practice with conditional statements and loop constructs
- To import modules and use methods

Question 1:

1. Create a new PyCharm project: **Exercise10**.
2. Copy the file **ex10_starter.py** into this project
3. Get the directory name from the environment using os.environ: HOMEPATH on Windows, HOME on Linux (we have done that part for you, notice the test of `system.platform`).
4. Construct a portable wildcard pattern using `os.path.join` (we have done that part for you as well)
5. Use the `glob.glob()` function to obtain the list of filenames
6. Use `os.path.getsize()` to find each file's size
7. Add a test to only display files that are not zero length
8. Use `os.path.basename()` to remove the leading directory name(s) from each filename before you print it.

Question 2:

Write a Python program that emulates the high-street bank mechanism for checking a PIN. Keep taking input from the keyboard (see below) until it is identical to a password number which is hard-coded by you in the program.

To output a prompt and read from the keyboard:

```
supplied_pin = input("Enter your PIN: ")
```

Restrict the number of attempts to 3 (be sure to use a variable for that, we may wish to change it later), and output a suitable message for success and failure. Be sure to include the number of attempts in the message.

Question 3 (Optional):

Passwords, and PINs, would not normally be displayed (echoed) to the screen, for security reasons. So now we will add the functionality to hide the characters typed. That

could be a lot of work, but one of the advantages of using a language like Python is that "there's a module for it".

You will need to import a module called **getpass**, which is part of the standard library.

Instead of input use **getpass.getpass**, in the same place in the program, with the same parameters.

Exercise 11 – String Handling

Objectives

- To manipulate strings using functions and methods

Question 1:

1. Create a new PyCharm project: **Exercise11**.
2. Copy the file **ex11_starter.py** into this project
3. You will see a string defined called 'Belgium'. Add code to print:
    a) A line of hyphens the same length as the Belgium string, followed by
    b) The string with the comma separators replaced by colons ':'., followed by
    c) The population of Belgium (the second field) plus the population of the capital city (the fourth field). **Hint**: the answer should be 11183818.

Exercise 12 – Collections

Objectives

- To understand the syntax of lists, tuples and dictionaries

Question 1:

1. Create a new PyCharm project: **Exercise12**.
2. Create a new python file in the project for each of the following questions.

What is wrong with this?

```
cheese = ['Cheddar', 'Stilton', 'Cornish Yarg']
cheese += 'Oke'
```

How should 'Oke' be added to the end of the cheese list?
How would you add two new cheeses to the list with a single command?

Question 2:

What is going on here? Can you explain the output?

```
tup = 'Hello'
print(len(tup))
```

Prints 5

```
tup = 'Hello',
print(len(tup))
```

Prints 1

Question 3:

Write a Python script called **lotto.py** that will generate and display 6 unique random lottery numbers between 1 and 50. Think about which Python data structure is best suited to store the numbers. Use the Python help() function to find out which function to use from the python standard library called **random**.

Exercise 13 – File Handling

Objectives

- To read and write to files

Question 1:

1. Create a new PyCharm project: **Exercise13**.
2. Create a new python file in the project called **write_file.py**
3. Write a line of code to create a file handle to open and append to a file called **pelican.txt**
4. Append the following line to the file using the **write** method of the file handle:

   "A wonderful bird is the pelican,"
5. Append the following second line using the **write** method:

   "His bill holds more than his belican."
6. Create a list that contains the following lines:

```
lines = ["He can take in his beak,\n", "Enough food for a week,\n",
"But I'm damned if I see how the helican.\n"]
```

7. Append this list to the file using the **writelines** method.

Why are the "\n" required?

8. Run the script and confirm a new file called pelican.txt was created and that it contains the limerick as expected.

Question 2:

1. Create a new python file in the project called **read_file.py**
2. Use the **open** and **read** methods to *slurp* the entire contents of your **pelican.txt** file
3. Display the type of the returned data and print the contents of the returned data.
4. What data type is the output?
5. Now, write some code that will read the pelican.txt file into a *list* and then output the number of items within the list.
6. Now use a *loop* to iterate over and print the contents of the file. Be sure not to include any blank lines in the output.

Exercise 14 – Functions

Objective

- To practise creating functions
- To revisit conditional structures
- To use built-in functions

Task A: Create a Game: Rock, Paper, Scissors!

You are going to create a command line version of the game: Rock…Paper…Scissors!

The user will play against the computer at this game. You should design a program that does the following:

- Prompts the user to enter a value: R, P or S
- The program should convert this value into Rock, Paper, or Scissors respectively
- Asks the computer to generate a random value between 0 and 2
- Convert the computer's choice. 0 becomes Rock; 1 becomes Paper; 2 becomes Scissors
- Compare the user's choice with the computer's choice to display a message indicating whether the user won, lost or drew against the computer
- Showcase what you have learned about conditional statements and create your own **functions**

Note:

Rock smashes Scissors, Paper wraps Rock, Scissors cut Paper!

Task B: Compare programs

Compare your solution to other class members'.

- Have you achieved this differently?
- Does anyone's solution stand out?
- How would you structure your code differently?
- Would you consider changing the names of your variables or functions?

Exercise 16 – Object Orientation

Objective

- To experiment with classes and objects
- To design an inheritance hierarchy
- To learn about encapsulation and polymorphism

Task A: Create a Person inheritance hierarchy

Design at least 3 classes: **Person**, **Employee** and **Customer**

Consider appropriate constructors, properties and methods for these classes

Demonstrate your understanding of encapsulation, inheritance and polymorphism

Create a client script and instantiate objects based on the above classes and call their methods and set their properties to demonstrate working functionality.

Task B: Create an Account inheritance hierarchy

As above, design a separate hierarchy for Bank Account types and a client script to demonstrate functionality.

NOTE: Organise your code into different files and use meaningful names for all attributes and methods.

Exercise 17 – Exception Handling

Objective

- To practice raising exceptions
- To practice catching exceptions
- To design your own exception

Task A: Create an InsufficientFunds exception

Create a new exception class called **InsufficientFundsException** for use with your Bank Account classes

Make sure your bank accounts have a withdraw method

Within this method, write the logic to ensure that your exception is raised if the withdrawal would result in the account going overdrawn

Task B: Handle the raised exception

In your client script, catch and handle the exception when it is thrown

Exercise 18 – MySQL Part 1

Objective

- To understand database design
- To use data definition language statements

Tasks:

Draft a list of user stories for a **library** application:

- *As a <type of user>,*
- *I want <some goal>*
- *So that <some reason>*

This collection of user stories should provide a good breadth of the functionality required for a library and identify the different roles within the business.

From these user stories extract a list of candidate table, column and role names for a database design. Create some sample data for each candidate table to help validate your design. Identify any data redundancy and consider further normalization if necessary.

Sketch out your database design to include datatypes, primary and foreign keys, constraints such as default and check constraints and nullability. Document the relationships to denote whether they are one-to-one or one-to-many. Ensure you use meaningful identifiers throughout your design.

Create this database within MySQL.

Exercise 19 – MySQL Part 2

Objective

- To utilise data control language and data manipulation language statements
- To create stored procedures
- To create roles and assign privileges
- To test the functionality of the database with scripts

Tasks:

Take your collection of user stories from the previous exercise and create the necessary SELECT, INSERT, UPDATE and DELETE statements required to fulfil the story activities.

Research stored procedures and create one or two procedures within the database.

Take the library roles identified within the stories and create database roles for each of these. Assign the appropriate privileges for these roles to the different tables and procedures.

Create test scripts to demonstrate the use of the statements and procedures created within the database.