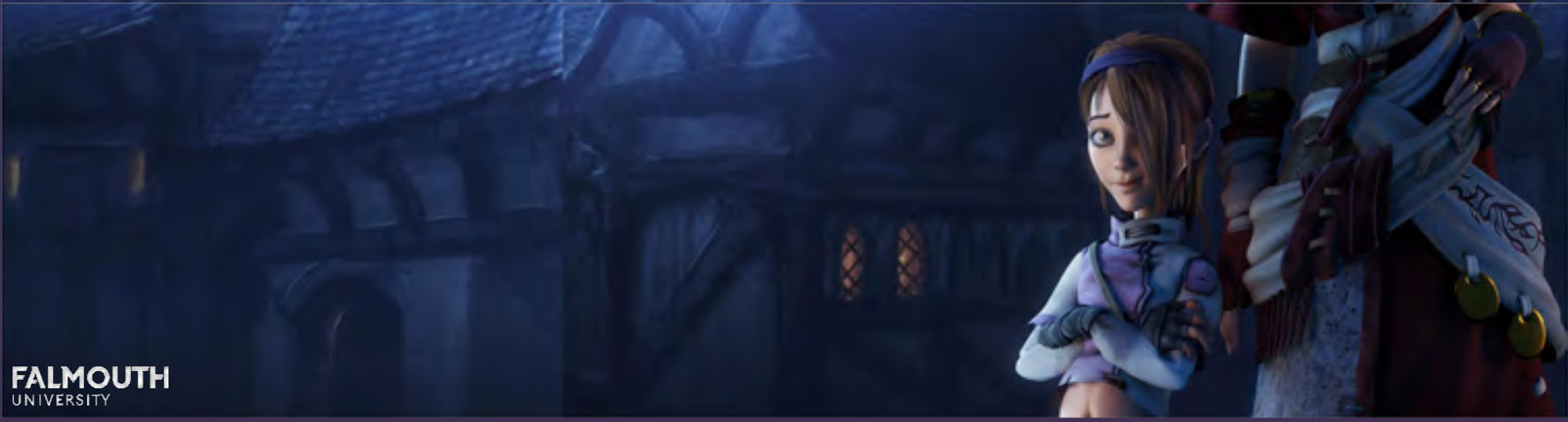

COMP120: Creative Computing

1: Tinkering in C#

Learning Outcomes

- **Outline** the role and basic functions of the IDE
- **Interpret** some basic C# code in Visual Studio
- **Apply** pair programming practices to solve a simple text concatenation problem
- **Explain how** pictures are digitised into raster images by a computer system

FALMOUTH
UNIVERSITY

Basic C# programs

Your first C# program

```
using System;

namespace Test
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

C# Terminology

- **Using** The using directive creates an alias for a namespace or import types defined in other namespaces.
- **nameSpace** A namespace is designed to keep one set of names separate from another. Consequently class names declared in one namespace do not conflict with the same class names declared in another.
- **Class** A class defines the kinds of data and the functionality objects will have. A class enables you to create your custom types by grouping variables of other types, methods, and events.
- **public static void Main** It is the first method which gets invoked whenever an application started and it is present in every C# executable file.

Your second C# program

```
Console.WriteLine("This is a very long line of code which had to be split to fit on the slide, but you should type it as a single line.")
Console.WriteLine("This is the second line of code.")
```

Assigning to variables

```
int a = 10;
Console.Writeline(a);
```

Variable	Value
a	

Remember!

- A program is a **sequence of instructions**
- The C# interpreter executes the **first line** of your program, then the **second line**, and so on
- When it reaches the end of the file, it **stops**

Socrative - FALCOMPMIKE

Login to Socrative!

`https://b.socrative.com/login/student/`

Reassigning variables (1)

```
int a = 10;
int b = 20;
b = a;
Console.WriteLine(a);
Console.WriteLine(b);
```

Variable	Value
a	
b	

Reassigning variables (2)

```
int a = 10;
int b = 20;
a = b;
Console.WriteLine(a);
Console.WriteLine(b);
```

Variable	Value
a	
b	

Reassigning variables (3)

```
int big = 10;
int small = 20;
big = small;
Console.WriteLine(big);
Console.WriteLine(small);
```

Variable	Value
big	
small	

Reassigning variables (4)

```
int a = 10;
int b = 20;
a = b;
b = a;
Console.WriteLine(a);
Console.WriteLine(b);
```

Variable	Value
a	
b	

Reassigning variables (5)

```
int a = 10;
int b = 20;
int c = 30;

a = b;
b = c;

Console.WriteLine(a);
Console.WriteLine(b);
Console.WriteLine(c);
```

Variable	Value
a	
b	
c	

Reading Input

```
Console.WriteLine("Enter your name:");
name = Console.ReadLine();

Console.WriteLine("Enter your age:");
age = Int16.Parse(Console.ReadLine());

Console.WriteLine($"Hello {name}");
Console.WriteLine($"On your next birthday, you will be
{age+1} years old");
```

- `Console.ReadLine()` reads a **string** (a sequence of characters—text) from the command line
- `Int16.Parse(...)` parses(converts) a **string** into an **integer** (a number)

Conditionals (1)

```
int a = Int16.Parse(Console
.ReadLine());

int b = 30;

if (a < 15) {
    b = a;
}

Console.WriteLine(a);
Console.WriteLine(b);
```

Variable	Value
a	
b	

Indentation

- Like many other programming languages, **indentation is not essential but useful** in C#
- C# uses indentation to denote the **block of code** inside a conditional, loop, function etc.
- Microsoft recommends **4 spaces** for indentation
  - Some programmers use a tab character
  - **Never** mix tabs and spaces in the same file!

`https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions`

Conditionals (2)

```
int a = Int16.Parse(Console
.ReadLine());

int b = 0;

if (a < 20){
    b = a + 1;
} else if (a == 20) {
    b = a * 2;
} else {
    a = 20;
    b = 20;
}

Console.WriteLine(a);
Console.WriteLine(b);
```

Variable	Value
a	
b	

Conditionals

An `if` statement can have:

- **Zero or more** `else if` clauses
- **An optional** `else` clause

In that order!

Mathematical operators

- `+` add
- `-` subtract
- `*` multiply
- `/` divide
- `**` power

Order of operations: **BIDMAS**

- Brackets first
- Then indices (powers)
- Then division and multiplication (left to right)
- Then addition and subtraction (left to right)

Comparison operators

- `<` less than
- `<=` less than or equal to
- `>` greater than
- `>=` greater than or equal to
- `==` equal to
- `!=` not equal to

Note the difference between `=` and `==`

- `a = b` means "make `a` be equal to `b`"
- `a == b` means "is `a` equal to `b`?"

For loops and ranges

```
for (int i = 0; i < 5; i++)
{
  Console.WriteLine(i);
}
```

- `for` contains 3 statements: **variable, condition** and **increment**
- Initially the **variable** is set to a value and the **incrementer** increases the value until the **condition** is met
- The `for` loop iterates through the items in a sequence **in order**. As the loop iterates the variable is increased each time: 0, 1, 2, 3, 4
- Note: `i < 5` **does not include** 5 as the condition is met at 4 so the loop stops.

For loops (1)

```
int a = 0;
int b = 0;

for (int i = 0; i < 5; i++)
{
    a = i;
    b = b + i;
}

Console.WriteLine(a);
Console.WriteLine(b);
```

Variable	Value
a	
b	
i	

For loops (2)

```
int a = 0;
int b = 0;

for (int i = 0; i < 5; i++)
{
    if (i < 3 || i > 7)
    {
        a += i;
    }
    else
    {
        b += i;
    }
}
Console.WriteLine(a);
Console.WriteLine(b);
```

Variable	Value
a	
b	
i	

While loops

The `while` loop keeps executing while the condition is **true**

```
int a = 1;

while (a < 100)
{
   a = a * 2;
}

Console.WriteLine(a);
```

Variable	Value
a	

Looping forever

```
int a = 1;

while (true) {
    a = a * 2;
    Console.WriteLine(a);
}
```

Summary

We have seen some basic code constructions in Python

- `Console.WriteLine()` and `Console.ReadLine()` for command-line input and output
- Variable assignment using `=`
- `if` statements for choosing whether or not to execute a block of code
- `for` loops to execute a block of code a specified number of times
- `while` loops to execute a block of code until a condition is no longer true

These are enough to write some simple programs, but you will see several more in coming weeks...

Challenge

- In pairs
- **Implement** the code excerpt
- **Fix** the errors in the code excerpt
- **Modify** the code excerpt to incorporate functions and arguments
- **Post** your solution to the `#comp120` slack channel

You can learn more about functions and arguments at:

`https://docs.python.org/3/tutorial/controlflow.html#defining-functions`

Challenge

The function:

```
public void madlib()
```

Should become:

```
public string madlib(string name, string pet,
string verb, string snack)
```

Challenge

```
public void madlib() {
        string name = "Link";
        string pet = "Spyro";
        string verb = "ate";
        string snack = "doughnuts";
        line1 = "once upon a time," + name + "walked";
        line2 = "with " + pet + ", a trained dragon.";
        line3 = "Suddenly, ' + pet + " announced,";
        line4 = "I really want some " + snack + "!";
        line5 = name + " complained. Where am I going to
        get that?";
        line6 = "Then " + name + "found a wizards wand.";
        line 7 = "With a wave of the wand, ";
        line8 = pet + " got " + snack + ". ";
        line9 = "Perhaps surprisingly, " + pet + " " +
        verb + "  " + snack;
        Console.WriteLine(line1 + line2 + line3 + line4);
        Console.WriteLine(line5 + line6 + line7 + line8
        + line9);
    }
```