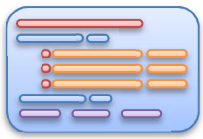
Programming in VB.NET



Iteration in Scratch

Scratch is a visual programming language used to create animations and games. We'll be using it to investigate how to control the sequence of events in a program.

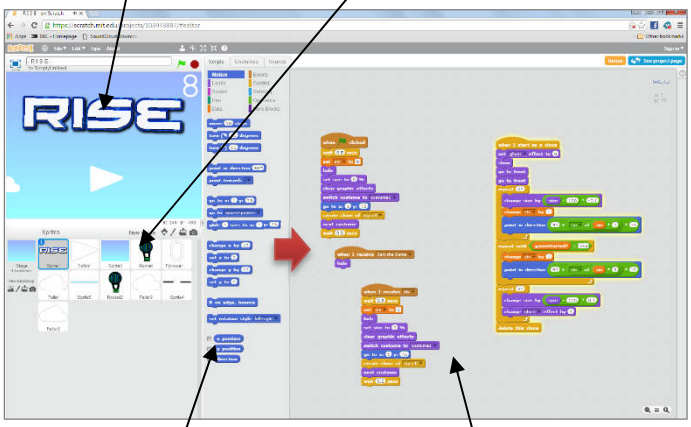
Aim: *Use visual programming in Scratch to investigate iteration.*

Stage

See your project in action.

Sprites

All the things that will be visible in your project.



Script Blocks

Drag these blocks into the script area.

Script Area

Build your scripts here.

Note: *Although it is possible to download and upload projects without an account, it's much easier to store your projects online by signing up.*

A brief guide to getting started

a. Go to https://scratch.mit.edu/ (or type 'Scratch' into Google). Create a login so that your work is saved.

b. Watch the introductory video and have a look at the 'Featured Projects'.

c. Choose a project and click on the 'See Inside' button. Click on the different sprites in the bottom-left section. Each sprite can have different scripts associated with it.

d. From the homepage, click on the 'Create' link. You will start with a cat on the screen which doesn't seem to do anything.

e. Drag a 'move' block into the *Script Area*. Change the number of steps to 20, then click on the blue area of the block. Did the cat move?

Note: *Full instructions included in our first Algorithms resource.*

Task 1 – The Bouncing Ball

a. Start a new project and click on the 'Choose sprite from library icon' as shown. A gallery of ready-made sprites will be displayed. Add a ball of some sort. We have added a basketball.

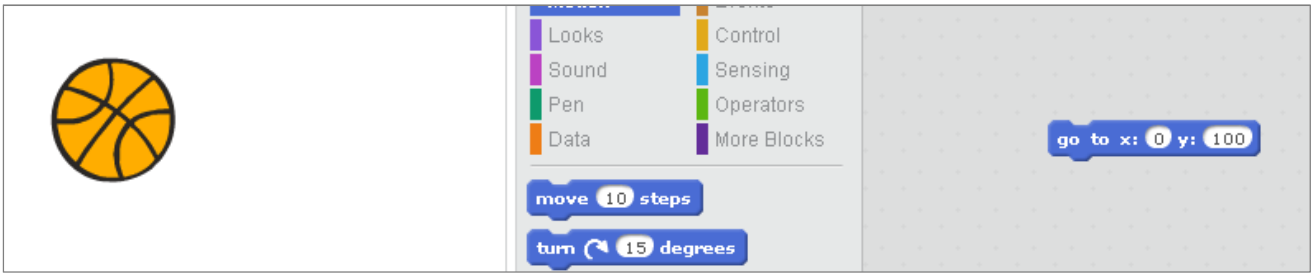
b. Delete the cat by right-clicking on the sprite and selecting 'Delete'. You should now only have a ball in the *Stage* window.

2. Delete the cat by right-clicking and selecting 'Delete'.

1. Add a new sprite by clicking here.

Iteration in Scratch (page 2)

c. Drag the ball and place it near the top of the *Stage*, horizontally centre. You should notice that the coordinates in the blue 'go to' script block change as you move the ball. Fix the values 'x: 0' and 'y: 100' in this script block and drag it into the script area.

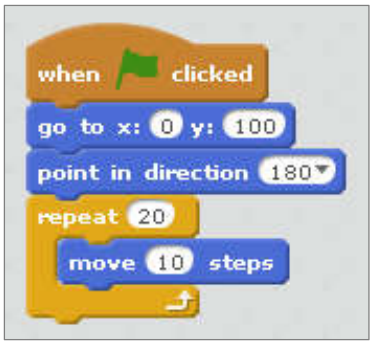


d. Put together the script shown on the right. Clicking the green Start flag above the Stage should now place the ball in the start position, point it downwards (180°) and move it 200 steps.

e. Try adding a 'repeat' loop as shown below-right. Rather than moving 200 steps in one go, we are now moving 10 steps 20 times. A slight delay is built into the loop, so the ball should now move at a more appropriate speed. Test this.

f. Add to the script so that after reaching the bottom, the ball is made to point upwards (0°) and then move back up to its original position.

g. You should now have a ball that bounces once. However, we want our ball to continue bouncing. Add another repeat block so that the whole process occurs 10 times. Your script should look like the one below-right.

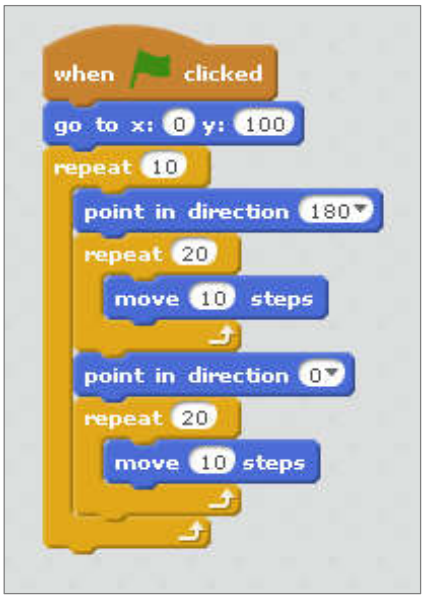


Questions

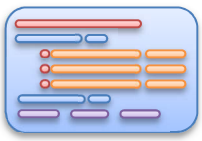
1. How many different loops have now been set up?

2. How many times does the flow of instructions loop back upwards all together?

3. Have a go at writing out this algorithm in pseudocode. You can use the REPEAT-UNTIL method. Try and use counters to make sure that each loop runs the correct number of times. We have started one possible solution below, but you may use a different method of achieving the same result.



SET ball to x:0,y:100
SET *bounceCount* = 0
REPEAT
    SET ball direction to 180°
    SET *downCount* = 0
    REPEAT



As you have learned, there are many programming languages out there. We are going to have a look at some Visual Basic code in the form of VB.NET. The reasons for choosing this language include:

Aim: *Try some visual basic programming.*

1. *It is relatively easy to understand – there are not too many fiddly rules to remember.*
2. *You can create and test some simple scripts using the website www.dotnetfiddle.net.*
3. *You can build fully fledged applications in the free Visual Studio.*
4. *You can write and edit macros that make dynamic things happen in MS Excel and MS Word (this uses a version of Visual Basic called VBA, or Visual Basic for Applications).*
5. *You could learn to set up dynamic web pages, retrieving information from a website and inserting this into the pages before they are presented to the user.*

VB is a very flexible language and a great place to start. Having said this, programming languages have many similarities. Once you've got your head around the basics of programming it's much easier to learn new languages as and when you need to.

Task 1 – Getting Started in .NET Fiddle

You will be using a website called *.NET Fiddle (Dot Net Fiddle*). This site allows you to play around with bits of VB.NET code without the need to install applications. There are limitations to what you can do in *.NET Fiddle*, but it's a great way to get started.

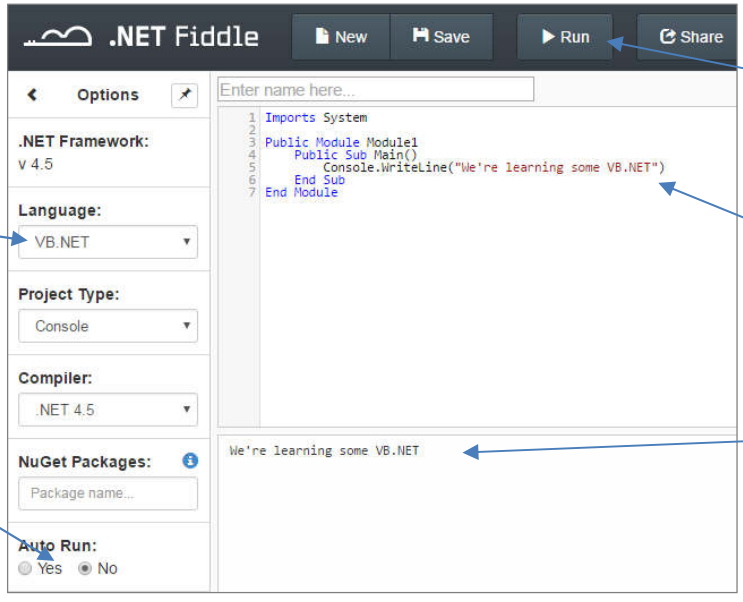
Note: *At the time of writing, .NET Fiddle works best with the Chrome browser.*

a. Navigate to the website www.dotnetfiddle.net. Set up an account so that you can save your work.

b. Using the menu on the left, change the *Language* to 'VB.NET'.

c. Change the red "Hello World" text to something new and wait for the console at the bottom to so show the change. This means that your code has run automatically.

d. Turn off *Auto Run* and use the *Run* button instead to start your programs. This can prevent errors when the application tries to run unfinished code.



1. Change the *Language* to 'VB.NET'
2. Change the *Auto Run* setting to 'No'
3. Edit the red text again
4. Click on the 'Run' button.
5. View the results of your program

Getting Started with Visual Basic (page 2)

Task 2 – Displaying a Name

The programming script below takes a first name and a last name, joins them together and then displays the full name. At this stage, the actual names are fixed in the program.

```
1  Imports System
2  Public Module Module1
3      Public Sub Main()
4
5          Dim FirstName as String
6          Dim LastName as String
7          Dim FullName as String
8
9          FirstName = "Sarah"
10
11         LastName = "Edwards"
12
13         FullName = FirstName & LastName
14
15         Console.WriteLine(FullName)
16
17     End Sub
18 End Module
```

The important bit

This line has to be present when programming in *.NET Fiddle*. Don't worry about what it does quite yet.

We write our code in a Sub (short for subroutine) within a Module. These are set up automatically so just leave them in place. The terms will be discussed later.

3 variables are declared so that we can use them later in our program.

The two names are allocated to two of the variables.

The two variables are added together. We say that the strings are concatenated.

The full name is displayed in the console.

The Sub and Module are closed.

a. Type the program into *.NET Fiddle* and run it. Write down the exact text that appears in the console.

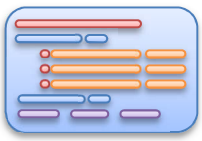
b. We want to add a space between the first name and the last name. We can do this by inserting the code & " " & between the two variables in line 13 in place of the single & symbol. Edit line 13 and rerun the program. Carefully write out the new line 13 below.

Note: *The '&' symbol joins two words (or strings) together. We are therefore now joining the first name, a space and the last name together.*

c. A variable is a space in the computer's memory. Think of it like a box where you can keep information until it's needed. Write down the names of the 3 variables used in this program.

d. Each variable has been declared using the word 'Dim' (short for *Dimension*). This is basically saying '*a variable exists and this is its name*'. On which lines are the 3 variables declared?

Note: *The variables have all been declared as strings. We'll talk about what this means in the following tasks.*



VB.NET allows you to store data in lists. When you create a list, you tell the program what type of data will be involved, the same as you do with a variable. The following line of code will declare a list called *lstNumbers*. The list will contain strings.

Aim: *To set up and edit a list.*

```
Dim lstNumbers as New List(Of String)()
```

Task 1 – Investigating Lists

The program below creates a list of numbers set as text, "One", "Two", "Three" and "Four". When run, the console displays the output shown on the right.

```
List created
Checking list in order: One
Checking list in order: Two
Checking list in order: Three
Checking list in order: Four
The number of items in this list is 4
The 1st item in this list is: One
The 4th item in this list is: Four
One
Two
Three
Four
```

a. Recreate the program in .NET Fiddle, saving as '**12.1 Basic Lists**'. Use the code and the output to work out exactly what each line is doing and add comments to the program to demonstrate your understanding.

Note 1: *Another 'Imports' statement is required for the list functions to work.*
Note 2: *This program uses 'For…Next' loops, which are a simple type of count-controlled loop.*

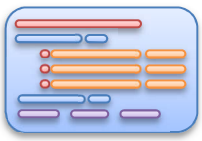
```
1  Imports System
2  Imports System.Collections.Generic
3
4
5  Public Module Module1
6
7      Public Sub Main()
8
9          Dim lstNumbers as New List(Of String)()
10
11         lstNumbers.Add("One")
12         lstNumbers.Add("Two")
13         lstNumbers.Add("Three")
14         lstNumbers.Add("Four")
15         Console.WriteLine("List created")
16
17
18         For Each listItem As String In lstNumbers
19         Console.WriteLine("Checking list in order: " & listItem )
20         Next
21
22
23         Dim numItems as integer
24         numItems = lstNumbers.Count
25         Console.WriteLine("The number of items in this list is " & numItems )
26
27
28         Console.WriteLine("The 1st item in this list is: " & lstNumbers.Item(0) )
29         Console.WriteLine("The 4th item in this list is: " & lstNumbers.Item(3) )
30
31
32         For i = 0 To numItems - 1
33         Console.WriteLine(lstNumbers.Item(i))
34         Next i
35
36     End Sub
37
38
39 End Module
```

Start the variable *i* at 0 and add 1 each loop until you reach 3, which is the index of the last item.

b. Edit the code so that it uses numbers up to "Six".

c. Add a line that writes "The 5th item in this list is…" to the console, as has been done with the 1st and 4th items.

d. Try and explain why line 32 above is "*For i = 0 To numItems – 1*" rather than "*For i = 1 To numItems*".



A data structure is a format for organising and storing data. You will probably have come across **files** plenty of times before. We've used **variables** and **lists** in previous tasks. Other data structures include **arrays**, **records, hash tables, queues** and **trees**.

Aim: *To learn about data structures such as arrays.*

Task 1 – Data Structure Match

Use your experience, common sense or information on the internet to match the data types below to their common use.

	Data Type				Use
1	Variable	•	•	a	Data organised into nodes; a root and then branching structures.
2	List	•	•	b	A structure with keys and values to look up, a bit like a dictionary.
3	Record	•	•	c	A number of items which can easily be changed in length or value.
4	Hash Table	•	•	d	A collection of fields of different type e.g. a 'row' in a database.
5	Tree	•	•	e	An item holding a single value.
6	Queue	•	•	f	A structure often containing a large number of lines.
7	Array	•	•	g	Data kept in order, inserted at one end and removed at the other.
8	File	•	•	h	A fixed number of values in one or more dimensions.

Task 2 – Two-Dimensional Arrays

An array is a simple data structure. Whereas a variable is a single box containing an item of data, an array is like a series of boxes all tied together. A one-dimensional array is similar to the lists we used previously. There are a few differences:

- Arrays must contain only a single data type (strings, integers etc.). Lists can contain a mixture.
- Arrays tend to be more static. You can't insert data or sort an array easily.

An array called *ThreeLetterWords1D* might be assigned the following values:

```
ThreeLetterWords1D(0) = "And"     'As with a list, the 1st index is zero.
ThreeLetterWords1D(3) = "Dab"     'The 4th word is 'Dab'
```

A two-dimensional array can be visualised as a grid. Two indexes are used; the first can be thought of as horizontal and the second, vertical. Use the array on the right to answer the questions.

a. What value is stored at the location *ThreeLetterWords2D(0 , 0)*? _____

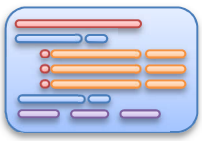
b. What value is stored at the location *ThreeLetterWords2D(3 , 1)*? _____

c. What is the location of the value "Fib"? _____

d. What is the location of the value "Car"? _____

Array named *ThreeLetterWords2D*

	0	1
0	And	Ear
1	Bat	Fib
2	Car	Gas
3	Dab	Hen



VBA, or Visual Basic for Applications, is the programming language used to add extra functionality to Excel spreadsheets, Word documents and other MS Office files. If you've ever recorded a macro in an Office application, then you have already created some VBA code – you probably just didn't realise it.

Aim: *To use our VB skills in MS Office applications.*

Note: *At the time of writing, some aspects of VBA are not quite the same as the VB.NET you've been learning (it's actually VB6). However, for the uses we'll make of it, the differences are minor.*

Task 1 – Recorded Macros in Excel

Macros are essentially subroutines that can be created to perform a set of commands in Excel. For example, you may frequently format cells or charts in a particular way and want to create a shortcut that completes all these tasks. Macros can be recorded and used without looking at any programming code. We'll start by using recorded macros, then delve into the VBA and see what's going on behind the scenes.

Note: *Macros are covered in more detail in our Excel Advanced and Word Advanced resources.*

To Record a Macro

a. Open a new Excel workbook and create the spreadsheet on the right. Make each column 100 pixels wide.

b. Select cell A1. We will create a formatting style for title cells.

c. Click on '**View / Macros / Record Macro**'. Give the macro the name 'Title'.

d. Type the letter 't' in the 'Shortcut Key' box next to 'CTRL +'. The macro will then be called by pressing the 'CTRL' and 'T' keys together.

	A	B	C	D
1	Title			
2				
3		Heading 1	Heading 2	Heading 3
4	Row 1			
5	Row 2			
6	Row 3			
7				

e. The macro can either be stored in a 'Personal Macro Workbook' on the computer or in 'This Workbook' (i.e. this file). So that the Excel application you are using isn't clogged up with macros, select 'This Workbook'. Click 'OK'.

f. ***WARNING****: YOU ARE NOW RECORDING. DO NOT CLICK ANYTHING YET.*

g. Use the icons in the ribbon to carefully format the text. We have used Bold, Red, Size 20, italic. Go slowly - try not to make mistakes or use 'Undo'.

h. Click on 'Macros' in the *View* tab and select 'Stop Recording'.

	A
1	*Title*
2	

To Use the Macro

Click on any other cell containing text and press the 'CTRL' and 'T' keys together. All the formatting changes should take place automatically.

Practice

Try recording some macros to perform other actions. You could format the colours and borders of cells, change the height and width of a cell or carry out other tasks.

Deleting Macros

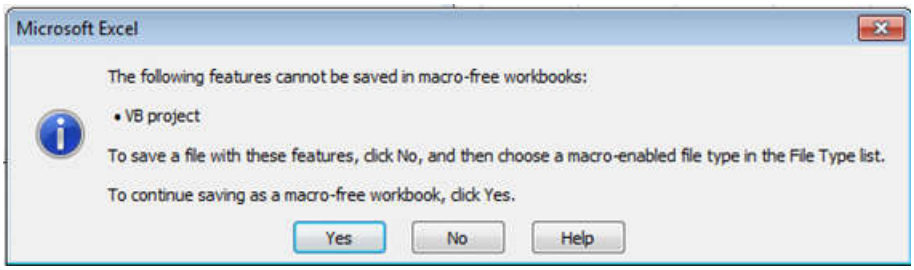
As long as you have saved your macros in 'This Workbook', then they can be deleted easily using the 'Macros' dialogue box. To do this, open the 'View' tab, click on 'Macros' then select 'View Macros'

Task 2 – Storing a Macro

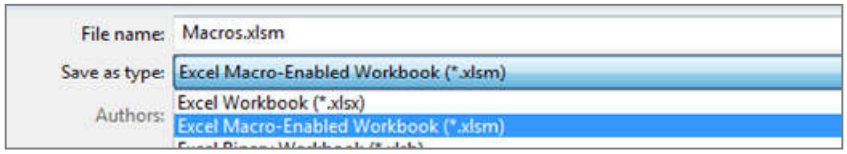
Security Notes

Macros are pieces of computer code, whether recorded using the icons in the ribbon or written yourself. A macro can be created that harms your computer. For this reason, there are security issues with Excel workbooks that you need to understand.

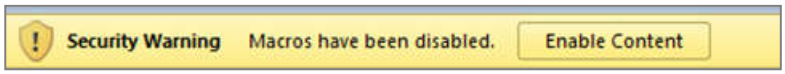
a. Try and save your workbook as the default type (xlsx). The window on the right should open explaining that you will not be able to save the complete document. You have a choice of saving the document without the macro (it will be lost) or cancelling. Click 'No' to cancel.



b. Try to save the file again, this time selecting 'Excel Macro-Enabled Workbook' (.xlsm).



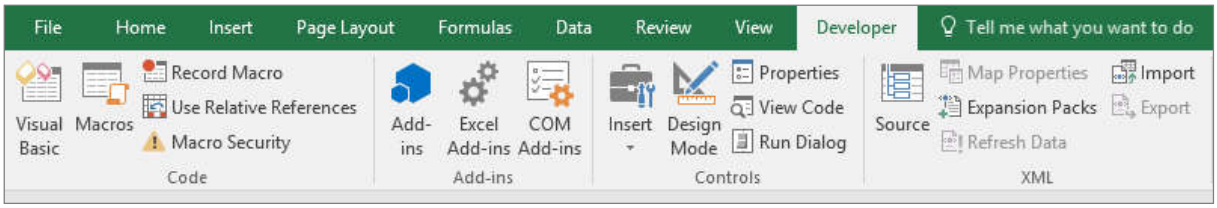
c. Close and then reopen your file. Test your macro to see if it still works; it probably won't. In order to use the macro, you must first enable it. At the top of the screen (below the ribbon) you should see the following message.



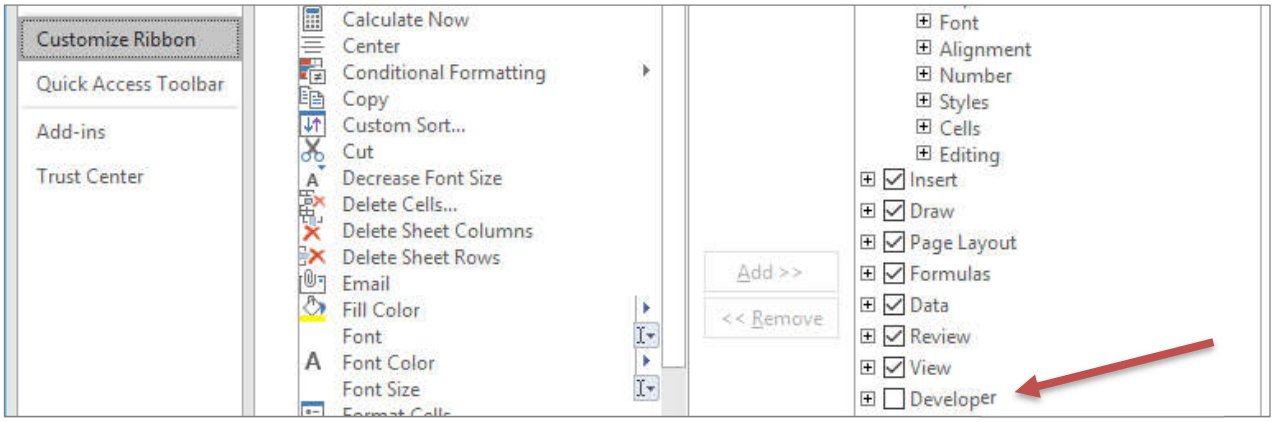
d. Click 'Enable Content' so that the warning disappears. Test your macro again. You should now be able to use it.

Task 3 – Enabling the Developer Options

Although it's not necessary for this task, more coding options are opened up if you enable the *Developer* tab on the ribbon.



To do this, select '**File / Options / Customize Ribbon**' then tick the '**Developer**' box on the right-hand side. If using a school computer, you may be asked by your teacher to hide this tab again when you are finished. You can get by without it for now.



Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. An IDE provides a range of tools for programmers to build their applications, such as editors, automation tools that help write the code, debuggers and visual tools. Visual Studio is used to develop computer programs for Microsoft Windows, as well as websites, web applications and web services.

Aim: *To introduce students to an integrated development environment.*

Note: *If you completed the previous task on VBA then you were using an IDE to edit your macros. However, although using VBA is fun and good practice for beginner programmers, it's not thought to have much of a future in the IT world. Visual Studio allows you to program in VB.NET, C++, F# and offers support for Python, Ruby, JavaScript and other languages.*

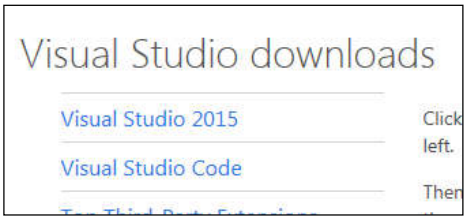
ORB Education VB.NET Projects

This resource is an introduction to Visual Studio. It will help you set up the application, understand the environment and complete a simple project. ORB Education offers sets of VB.NET projects which offer a range of further challenges for you to investigate.

Download the Free Software (Visual Studio Express)

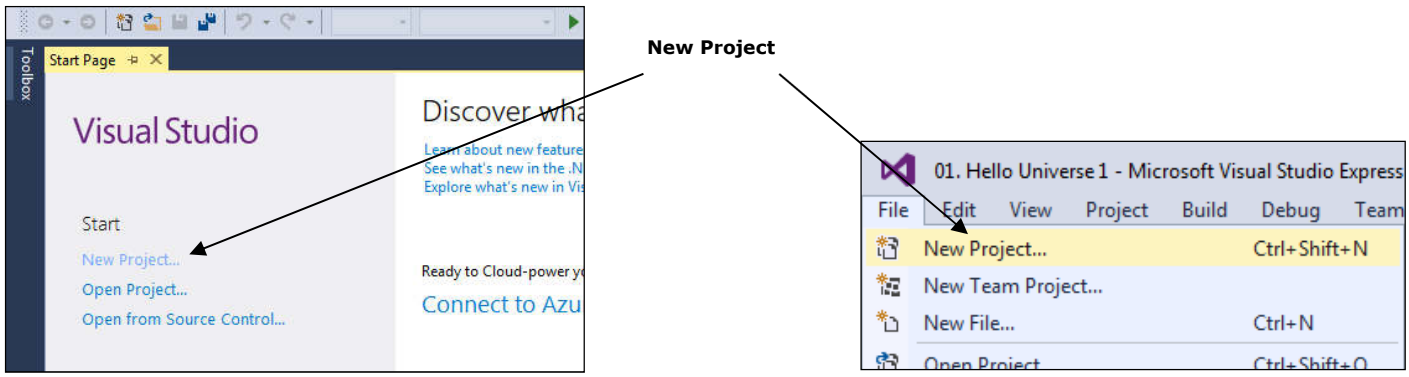
If you are working at school, Visual Studio will probably have been installed on the computers for you. However, if you have a Windows computer or emulator at home, you may also download the free version of Visual Studio Express and work on projects there. Follow the instructions below to download and install the application. Remember though, that Visual Studio is a serious addition to your computer; make sure that you have permission from the computer's owner. You will need full administrator rights, a decent internet connection for the download and a fair bit of space.

a. Go to the website www.visualstudio.com.
b. Click on 'Downloads' in the menu and select 'All Downloads'.
c. Select 'Visual Studio 2015' from the menu then 'Express 2015 for Desktop'.
d. Follow the prompts, run the installer and install the application.



Starting a New Project

a. After starting the application, click on 'New Project' on the start page. Alternatively, from within the application, click '**File / New Project**'.



b. Select 'Visual Basic' from the menu on the left, then 'Windows Forms Application' from the main window.