

FutureMakers Workshop

Artificial intelligence (AI)
and machine learning (ML)

Part A

Basic programming in Python with CodeSkulptor

Version 1.0.1

This guide will help you complete the FutureMakers workshop.
Follow along at your own pace.

Your name: _____

*Don't forget: programming can be difficult, but it's fun — **you can do it!***

*Don't be afraid to **make mistakes** - mistakes are how we learn.*

*Don't be afraid to **ask questions** - the instructors want to help you.*

Remember: we learn by making mistakes, and it's a good idea to ask questions!

Introduction

Welcome to the FutureMakers workshop. We'll go through a series of activities designed to introduce you to programming, machine learning, and artificial intelligence. Don't worry if you don't have much experience with coding, or even if you've never ever programmed before. This workshop will show you how to code in Python (Part A), how to use exciting tools like face detectors or automatic language translation (Part B), and how to build your own interactive website (Part C).

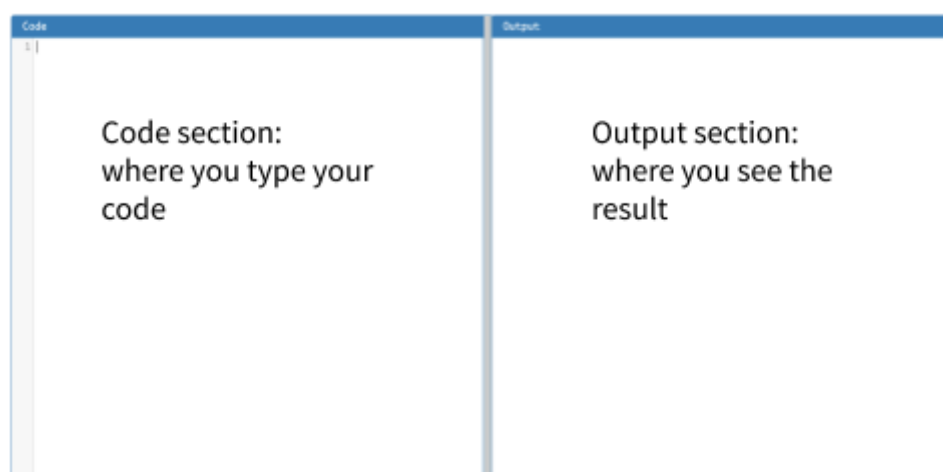
Throughout this workshop, remember that programming can be difficult - but it's also very fun. Don't be afraid to make mistakes and ask questions - making mistakes is how we learn, and even the most professional programmers frequently search on the Web for things they don't understand. Programming isn't about knowing all the answers - it's about knowing how to look for them, often by doing a Web search or asking a friend or colleague.

Part A

Starting to program (in Python)

CodeSkulptor is a website designed for learning to code in Python without having to set anything up on your computer. Open CodeSkulptor: <https://py3.codeskulptor.org/>. Or just search for CodeSkulptor Python 3. **Make sure you are using the Python 3 version, not the Python 2 version.**

CodeSkulptor has two sides:



Start by deleting everything from the code side (on the left), so that it's empty.

Remember: we learn by making mistakes, and it's a good idea to ask questions!

What does it mean? The programmer and the user

The programmer is the person who writes the code.

The user is the person who runs the code.

For example, when you are playing a computer game, you are the user.

When you are testing your own program, you are the programmer *and* the user.

There are three steps to coding:

1. Write your code

Here you enter instructions in the Code box.

A screenshot of a code editor window. The title bar is blue and says "Code". The editor area has a light yellow background. The first line of code is "1 print('Hello World')". The text is color-coded: "1" is blue, "print" is green, and the string is in red quotes. A mouse cursor is pointing at the end of the line.

2. Run your code

You click the Run button:




...and CodeSkulptor will run your Python code and try to follow your instructions.

3. Debug

Did you write the right instructions for the computer to execute? Often, what actually happens is different from what we expect. We might find problems - called bugs - in our code.

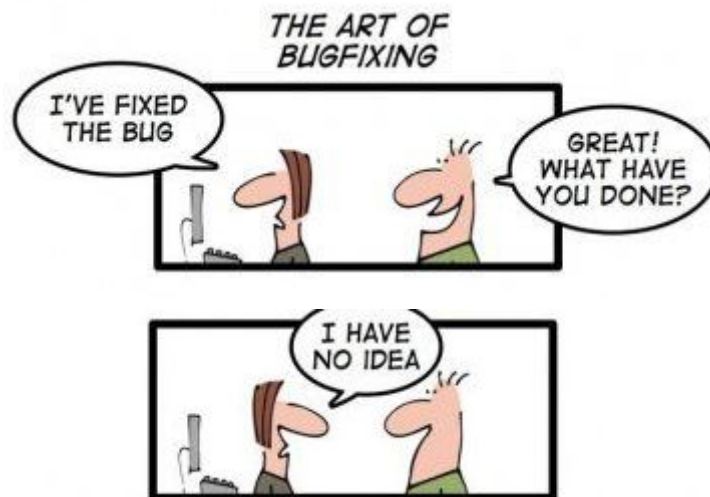
What does it mean? Bugs

Bugs are problems with your code which mean that it doesn't do what you want. When you encounter a bug, an error message will appear (usually in red) giving some information on what is wrong. In this error message, we have tried to use a variable called "greeting", but the variable couldn't be found.

A screenshot of an "Output" window. The title bar is blue and says "Output". The text inside is red and reads "Line 1: NameError: name 'greeting' is not defined".

Remember: we learn by making mistakes, and it's a good idea to ask questions!

Fixing bugs is an important part of programming:



Exercise: Running code

Make sure the Code window is empty, then enter the following program:

```
print('Hello World')
```

Then click Run.

Your turn

Modify the previous program to say something different.

Exercise: Asking the user a question

Enter the following program:

```
name = input('What is your name?')
greeting = "Hello " + name
print(greeting)
```

What does it mean?

A **variable** is a part of your program which can remember things.

Here we have two variables: `name` (which stores the user's name when they input it, such as 'Alex') and `greeting` (which stores the greeting, such as 'Hello Alex'). We used the plus (+) to stick the two parts of the greeting ('Hello ' and 'Alex') together. The plus (+) can be used to join words together as well as to add numbers.

Your turn

Remember: we learn by making mistakes, and it's a good idea to ask questions!

Change the last program to say Goodbye to the user.

Question: why is there a space after "Hello"? What happens if we take this space out? Take it out and run the program again.

Exercise: doing things many times

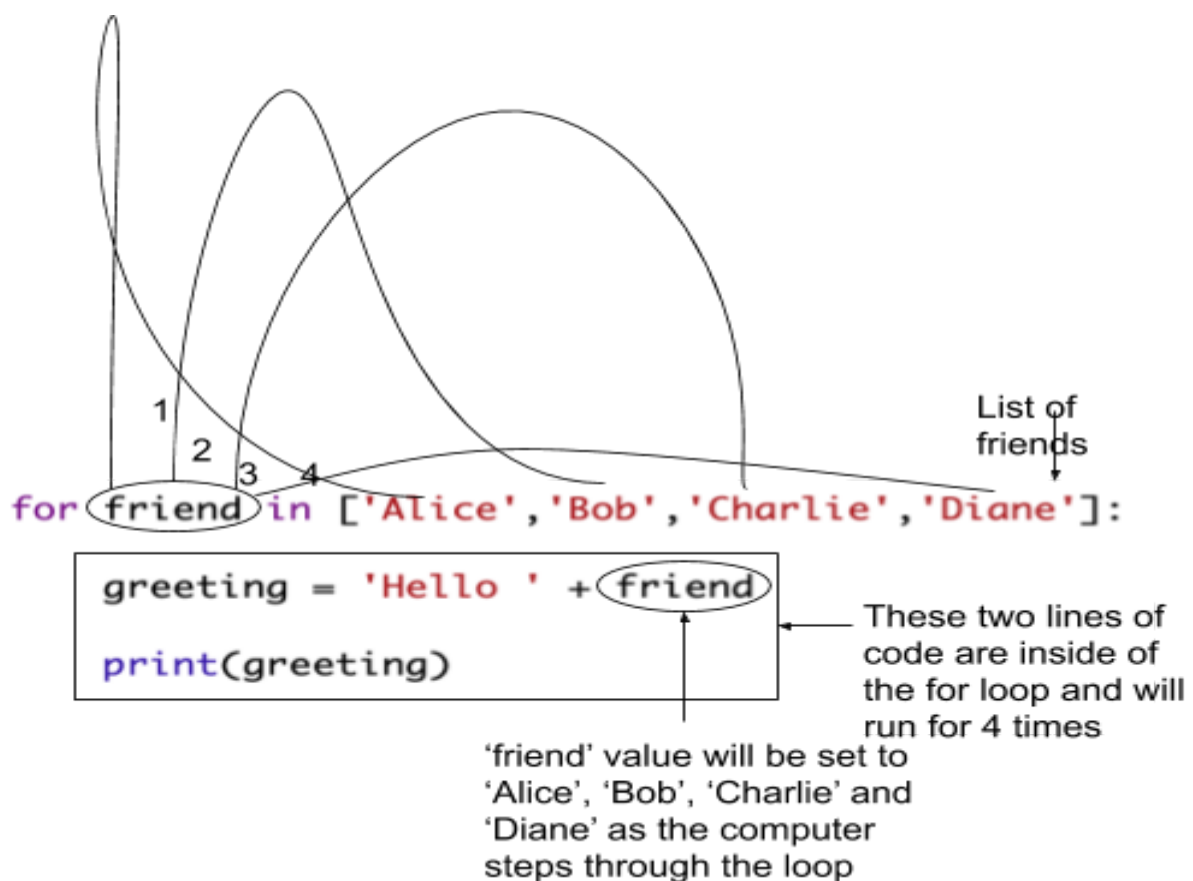
Before you start, make sure the code window is empty. If you want to save your code from the previous exercise, you can open a new tab to get a fresh CodeSkulptor window (make sure to use the Python 3 version).

Type in the following program:

```
for friend in ['Alice', 'Bob', 'Charlie', 'Diane']:
    greeting = 'Hello ' + friend
    print(greeting)
```

Then click Run.

This is called a **loop**. We know it's a loop because it starts with the word **for**. Wherever you see **for**, you know something is going to happen several times.



Remember: we learn by making mistakes, and it's a good idea to ask questions!

Question: Why is there some blank space before some of the lines?

This blank space is called an indent. Every line with an indent is inside a loop.

Sometimes, CodeSkulptor will helpfully insert an indent automatically for you. If you want to make one yourself, press the tab key.

What does it mean?

A **bug** is a problem with your program. Sometimes bugs happen because you made a mistake. Sometimes they happen because another programmer made a mistake.

Bug: what happens if we forget the indent?

Delete the indents so that your program looks like this:

```
for friend in ['Alice', 'Bob', 'Charlie', 'Diane']:
greeting = 'Hello ' + friend
print(greeting)
```

Now run the program. In the output, you will see

```
Line 2: SyntaxError: bad input ('greeting')
```

This means that your program has a bug. Here, the bug is that there is no indent. Because there is a loop, Python is expecting an indented line; it doesn't find one, so it doesn't know what to put in the loop and shows us an error.

Your turn

Let's add some friends to the list.

This part of the program is a list of friends:

```
['Alice', 'Bob', 'Charlie', 'Diane']
```

You can see that each friend's name is in quotes, and they are separated by commas. Add some more names to the list. Don't forget the quotes and the commas. For example:

```
['Alice', 'Bob', 'Charlie', 'Diane', 'Jeremy']
```

Then run the program again.

Tip

Words which you want to print on the screen (like names) always need to be in quotes, like 'London'.

```
print('London')
```

However, variable names (such as greeting) do not need to be in quotes.

Exercise: loops with numbers

Remember: we learn by making mistakes, and it's a good idea to ask questions!

Imagine that you want to say something 10 times. For example:

```
I have 1 banana!
I have 2 bananas!
I have 3 bananas!
I have 4 bananas!
I have 5 bananas!
I have 6 bananas!
I have 7 bananas!
I have 8 bananas!
I have 9 bananas!
I have 10 bananas!
```

We could do this with print:

```
print('I have 1 banana!')
print('I have 2 bananas!')
...and so forth...
```

But this would be boring and repetitive. Try out the following code:

```
for i in range(10):
    print('I have ' + str(i) + ' bananas!')
```

That's much neater.

However, there are still two problems - can you see them? The following code fixes them:

```
for i in range(10):
    j = i + 1
    if j == 1:
        print('I have ' + str(j) + ' banana!')
    else:
        print('I have ' + str(j) + ' bananas!')
```

Can you see how?

Tip: range(x) produces the numbers starting at 0 and continuing until x-1 is produced. For example, range(3) is [0, 1, 2].

Exercise: doing something once, then some things repeatedly, then something once

Enter the following code:

```
name = input('What is your name?')

for thing_to_say in ['Hello', 'Have a nice day', 'Goodbye']:
```

Remember: we learn by making mistakes, and it's a good idea to ask questions!

```
print(thing_to_say + name)

print('Have fun!')
```

Tip: if you have typed the loop line (the line starting with **for**) properly by ending it with a colon, CodeSkulptor will automatically indent the next line. If it doesn't, you can put the indent yourself by pressing the tab key.

What's happening here? The first line, which gets the user's name, is being run once. Then the line inside the loop, which says something to the user, is being run several times. Finally, the last line, which says 'Have fun!' is being run once.

Tip: a line which is indented might be in a loop, so might be run several times. A line which is *not* indented is not in a loop, so will only be run once.

Your turn

Make your own loop and run it. It can do anything you want! You can get input from the user first if you want to, but you don't have to. You can even get input from the user *inside* the loop - this means you will get input several times.

Exercise: doing some maths

Before you start, make sure the code window is empty. If you want to save your code from the previous exercise, you can open a new tab to get a fresh CodeSkulptor window (make sure to use the Python 3 version).

There are many things we can do with Python. One of the simplest is to use it as a calculator.

- When we are programming, we use the plus (+) symbol to add numbers and the minus (-) symbol to subtract numbers.
- However, we don't use the times (×) symbol to multiply numbers, we use the star (*) instead.
- And we don't use the division (÷) symbol to divide numbers, we use the slash (/) instead.

Enter and run the following code:

```
print(14 * 21)
```

You can see that the answer is printed.

Your turn: enter two really big numbers (with 6 or 7 digits each) and multiply them. Python is much, much more powerful than your calculator!

Exercise: maths with variable names

Remember: we learn by making mistakes, and it's a good idea to ask questions!

We can use variables to store numbers with a particular name. Enter and run the following code:

```
number = 5
other_number = 10

total = number + other_number
print(total)
```

Bug: what happens if we try to glue a word to a number?

Enter the following code:

```
number = 5
other_number = 10

total = number + other_number
print('The total is ' + total)
```

Now try and run it. You will see an error:

Line 5: TypeError: cannot concatenate 'str' and 'int' objects

What does it mean? "Concatenate" just means "glue" or "stick together". Python is confused because we are asking it to glue a string (word - **str** is short for string) to an integer (a whole number).

Python is quite clever, but sometimes it doesn't know how to do obvious things. To fix this error, we just need to convert our integer variable into a string first.

Exercise: convert an integer to a string

Before you start, make sure the code window is empty. If you want to save your code from the previous exercise, you can open a new tab to get a fresh CodeSkulptor window (make sure to use the Python 3 version).

Enter and run the following code:

```
my_integer = 7
my_string = '7'

print('This is the integer:')
print(my_integer)

print('This is the string:')
print(my_string)

print('This is the integer converted to a string:')
```

Remember: we learn by making mistakes, and it's a good idea to ask questions!

```
print(str(my_integer))

print('They all look the same!')
```

Exercise: if statements

We've seen how to use a loop to do something several times: this is one of the most important techniques in programming. Another key technique is the **if** statement, which is a way to make a choice: to do one thing under certain circumstances, otherwise to do something else.

Enter the following program in the code panel, and click Run.

```
temperature = float(input('What is the temperature? '))
if temperature > 30:
    print("It's not too hot.")
else:
    print("It's too hot!")
```

Once you enter a temperature, the program makes a choice and decides what to do.

Exercise: running some examples from CodeSkulptor

There is a button on the left side of CodeSkulptor which shows lots of examples written by other people. Have a look around, open one of the examples, have a read (ask an instructor if you have any questions), and click Run.

Then, have a look at some more examples, read them, and run them.