



THE UNIVERSITY OF
WESTERN
AUSTRALIA

Lecture 3

How to write code in Python

Revision: Getting Started with Python

Start with single statements

```
>>> 2+3
```

```
5
```

```
>>> 22/7
```

```
3.142857142857143
```

```
>>> 3**2
```

```
9
```

```
>>> print("Hello world")
```

```
Hello world
```

```
>>> print("2+3=", 2+3)
```

```
2+3=5
```

Functions Group Multiple Statements

- To solve a problem, we generally need to execute more than one statements.
- One way to do this is to use a *file*
- Another way to do this is to use a **function**

```
>>> def hello():  
    print("Hello")  
    print("Computers are Fun")
```

```
>>>
```

Defining Functions in Python

```
>>> def hello():  
    print("Hello")  
    print("Computers are Fun")
```

```
>>>
```

- The first line tells Python we are defining a new function called “hello”.
- The following lines are indented to show that they are part of the hello function. **Indent must be uniform**
- The blank line (hit enter/return twice) on shell lets Python know the definition is finished.

Executing, or Invoking, a Function

```
>>> def hello():  
    print("Hello")  
    print("Computers are Fun")
```

```
>>>
```

- Notice that nothing has happened yet! We defined the function, but we haven't told Python to execute the function!
- A function is **invoked** or **executed** by typing its name.

```
>>> hello() ← Brackets are important!  
Hello  
Computers are Fun  
>>>
```

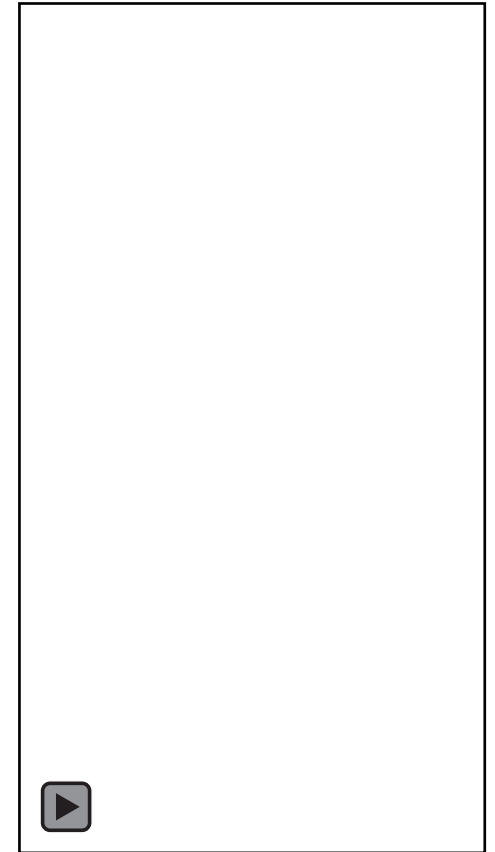
Problems with scripts (files without functions)

- In the scripts we have seen, the entire script consists of just one “block” of statements, executed in order – but this can become unmanageable when your script is thousands of lines long, contained in just one file.
- You may want to use previously written code in other programs. But cutting and pasting bits of code can create inadvertent variable name clashes.
- Scripts are not as flexible as we would like.
- How to address this?

A general problem solving technique is to break down complex problems into smaller, more manageable tasks.

Functions help us to avoid those problems

Functions as black box



Functions can take Inputs (Parameters)

- Functions can have changeable parts called parameters that are placed between the brackets.
- The function “hello” did not need any parameters.
- Here is another function that has one parameter.

```
>>> def greet(person):  
    print("Hello", person)  
    print ("How are you?")
```

```
>>>
```


Invoking a Function that has parameter(s)

- A function that has **parameters** requires **arguments**
- If we try to execute the function `greet()`

```
>>> greet()
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#74>", line 1, in <module>  
    greet()
```

```
TypeError: greet() takes exactly 1 argument (0 given)
```

- It gives us an error because we did not specify a value for the parameter “person”

Passing Parameters to Functions

```
>>> greet("Terry")
Hello Terry
How are you?
>>> greet("Paula")
Hello Paula
How are you?
>>>
```

- When we use parameters, we can customize the output of a function.

Why You need to Use Functions

- Define once, use many times
 - *Replace repeated code sections with a parameterized function*
- Aids problem decomposition
 - *Even if code is used just once, helps break problem into smaller, manageable pieces*
 - *Like sections and paragraphs in a paper, or chapters and paragraphs in a story*
- Defining code as functions allows independent testing/validation of code

Functions in a file

- When we exit the Python interpreter, all functions that we defined will cease to exist.
- How about writing them in a file and saving it.
 - *Saves a **LOT** of retyping*
- A *programming environment* is designed to help programmers write programs and usually include automatic indenting, highlighting, etc.

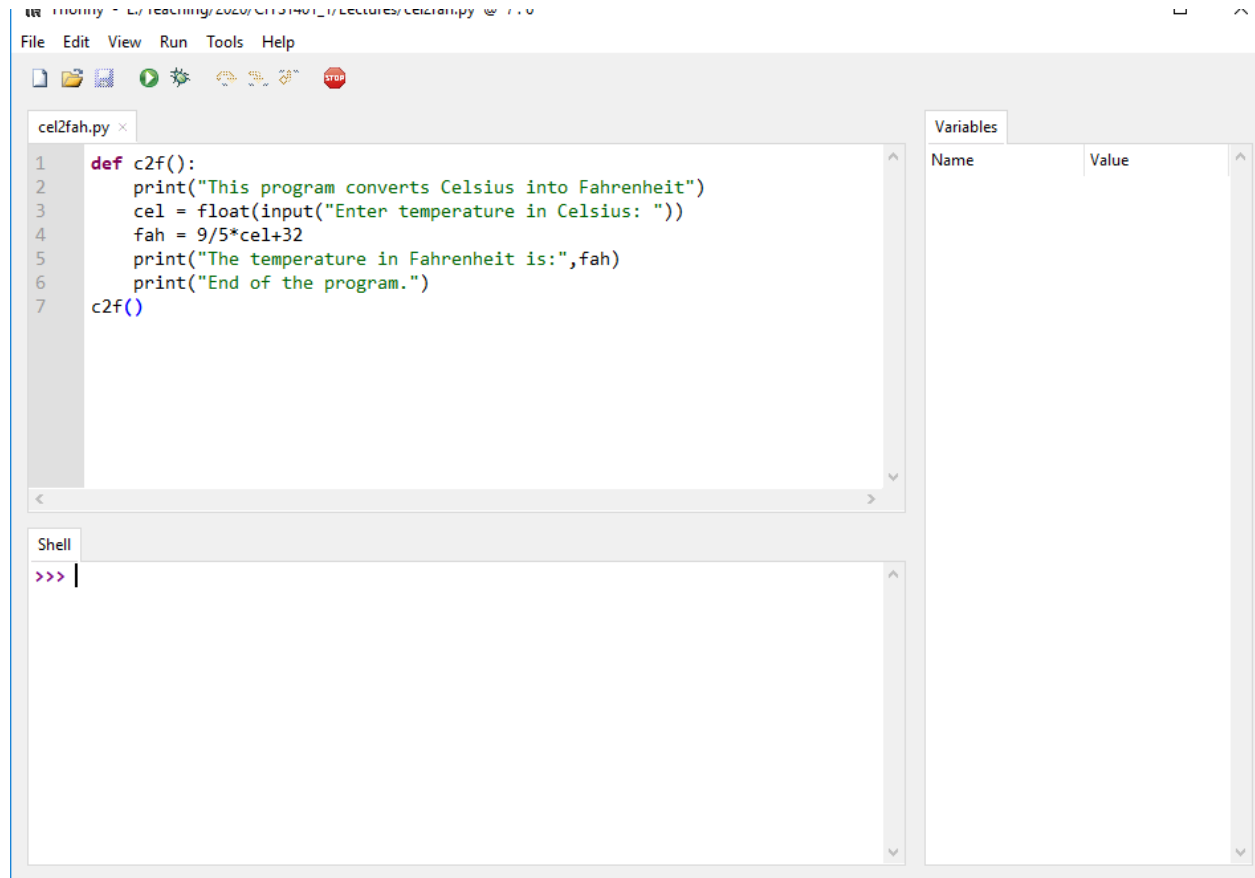
Creating a Module File

```
# File: cel2fah.py
# A simple program is illustrating Celsius to Fahrenheit conversion

def c2f():
    print("This program converts Celsius into Fahrenheit")
    cel = float(input("Enter temperature in Celsius: "))
    fah = 9/5*cel+32
    print("The temperature in Fahrenheit is:", fah)
    print("End of the program.")
c2f()
```

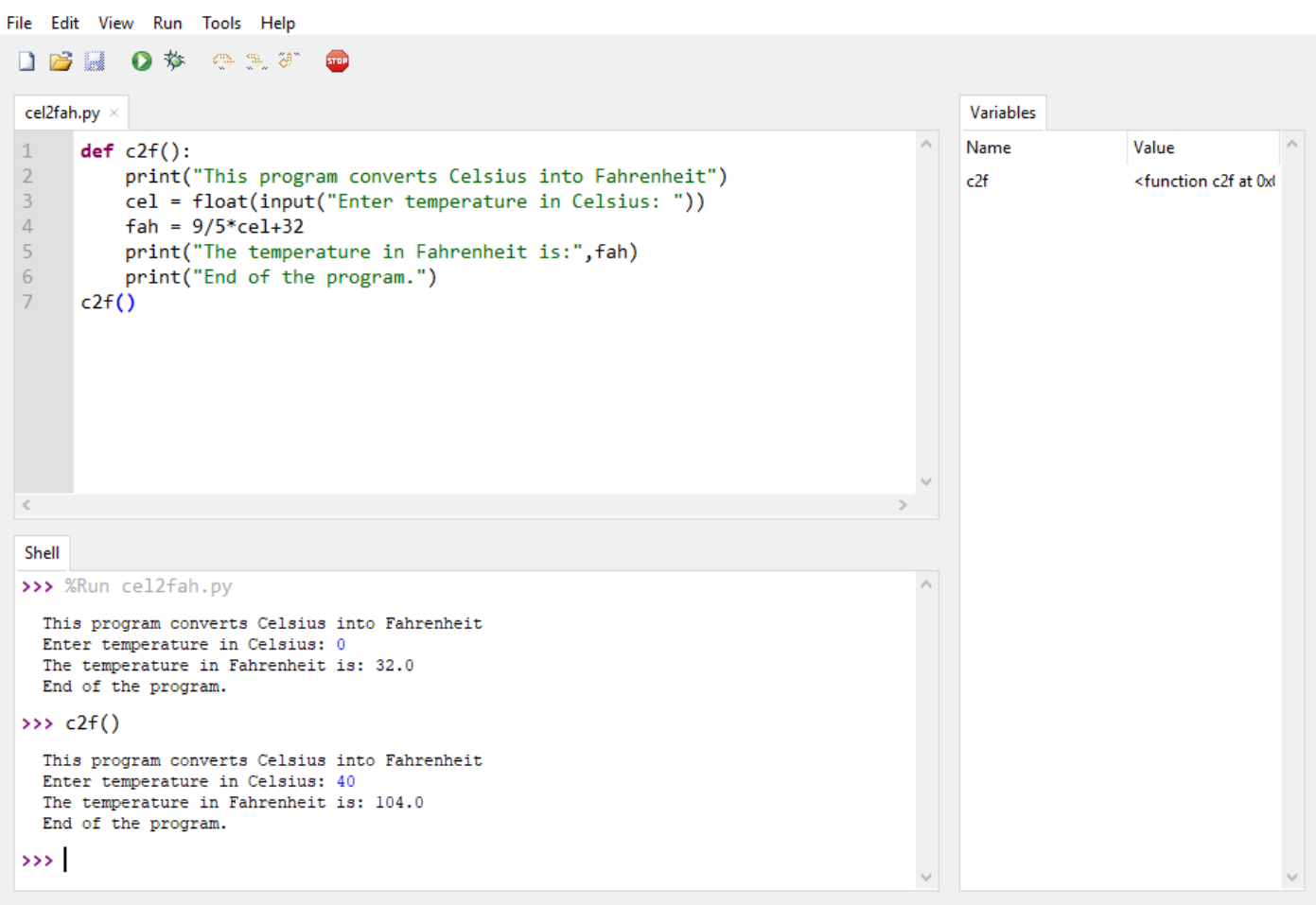
- We use a filename ending in .py when we save our work to indicate it's a Python program.
- Click green button (run) on Thonny to run the program.

cel2fah.py using Thonny IDE



Is the **Run** button, or choose from main menu

Running cel2fah.py using Thonny



The screenshot shows the Thonny Python IDE interface. The main editor window displays the code for `cel2fah.py`:

```
1 def c2f():
2     print("This program converts Celsius into Fahrenheit")
3     cel = float(input("Enter temperature in Celsius: "))
4     fah = 9/5*cel+32
5     print("The temperature in Fahrenheit is:",fah)
6     print("End of the program.")
7 c2f()
```

The Shell window at the bottom shows the execution of the script:

```
>>> %Run cel2fah.py

This program converts Celsius into Fahrenheit
Enter temperature in Celsius: 0
The temperature in Fahrenheit is: 32.0
End of the program.

>>> c2f()

This program converts Celsius into Fahrenheit
Enter temperature in Celsius: 40
The temperature in Fahrenheit is: 104.0
End of the program.

>>> |
```

The Variables pane on the right shows the function object `c2f`:

Name	Value
c2f	<function c2f at 0x...

Inside a Python Program

```
# File: cel2fah.py
```

```
# A simple program is illustrating Celsius to Fahrenheit conversion
```

- Lines that start with `#` are called *comments*. Similar to text enclosed in triple quotes as discussed in earlier lecture
 - *Comments can begin in the middle of lines, too*
- Intended for human readers and ignored by Python
 - **Important**, *so you or other maintainers of that code know what you were intending*
 - *Helps maintainability*
- Python skips text from `#` to end of line

Inside a Python Program

```
def c2f () :
```

- Beginning of the definition of a function called *c2f*
 - *Note the :* is important. It separates header from the function body

Inside a Python Program

```
print(" This program converts Celsius into Fahrenheit")
```

- This line causes Python to print a message introducing the program to the user.
 - *The message is sent to **Standard Output** (usually the computer screen)*
 - ***Standard Input** is usually the keyboard*

Inside a Python Program

```
cel = float(input("Enter temperature in Celsius: "))
```

- `cel` is an example of a *variable*
- A variable is used to assign a name to a memory location so that a value can be stored there and later retrieved.
- Variables come into existence when first assigned to
- The quoted text is displayed. The user enters a number (which is text, i.e. just numerical letters).
- The function `float` converts the string, e.g. “0.5”, into the number 0.5, which is then stored in `cel`.
- Note function call within function call (inner one called first)

Inside a Python Program

```
fah = 9/5 * cel + 32
```

- This is called an *assignment* statement
- The part on the right-hand side (RHS) of the = is a *mathematical expression*
- *, + and / are used to indicate multiplication, addition and division respectively
- Once the value on the RHS is computed, it is stored back into (*assigned*) into fah

```
print("The temperature in Fahrenheit is:", fah)
```

- Prints the calculated temperature fah to standard output

Indenting your Python programs

```
# File: cel2fah.py
# A simple program is illustrating Celsius to Fahrenheit conversion

def c2f():
    print("This program converts Celsius into Fahrenheit")
    cel = float(input("Enter temperature in Celsius: "))
    fah = 9/5*cel+32
    print("The temperature in Fahrenheit is:", fah)
    print("End of the program.")
```

- Indentation is used in Python programs to indicate the different **blocks** of statements. These are executed together, one after the other
- Note the colon highlighted in purple

Inside a Python Program

`c2f ()`

- The interpreter first creates a function definition
- The last line tells Python to *execute* the code in the function `c2f`
 - *No arguments expected so none supplied*

Executing a Python Program from a File

- You can run a program in a file any time you want using one of the following methods:
 1. Using Thonny, the easiest way is to click the green forward arrow or select **Run** from the **Run Module**
 2. On the command line (windows) or terminal (Mac OS), enter `python ./cel2fah.py` (`./` generally not be need if **path variable** has been specified)
 - *Paths are where system looks for files and programs*
 3. You can also double click the `.py` file in Windows to run it

Importing a module

```
>>> import cel2fah
```

 **Note: No .py suffix!**

```
This program converts Celsius into Fahrenheit
```

```
Enter temperature in Celsius: 0
```

```
The temperature in Fahrenheit is: 32.0
```

```
End of the program.
```

```
>>>
```

- This tells Python interpreter to load the file `cel2fah.py` into the main memory.
- Since the last statement of `cel2fah.py` is `c2f()` the function will get executed upon importing the file.
- Importing modules very common (particularly library modules – huge range, performing many useful functions)

Importing a Module

- When Python imports a module, it executes each line.
 - *The `def` causes Python to create the function `c2f`:*
 - *`c2f()` call at the end executes the function*
- Upon first import, Python creates a companion file with `.pyc` extension. This is an intermediate file containing the **byte code** used by the interpreter.
- Modules need to be imported in a session only once.

Modules and Functions

- You can define multiple functions in a module file
- You can call a function by typing
moduleFileName.functionName(...)
 - *E.g.* `>>> cel2fah.c2f()`
`>>> math.sqrt(2)`

A Word about Eval – Eval is Evil

- The second edition of Zelle uses the `eval` function:

```
x = eval(input("Enter a number between 0 and 1: "))
```
- This is very risky and no longer considered best practice
- `eval` evaluates **whatever** you give it, so long as the data is syntactically legal.
- Someone who knows Python could exploit this ability and enter malicious instructions, e.g. capture private information or delete files on the computer.
- This is called a code injection attack, because an attacker is injecting malicious code into the running program.
- When the input is coming from untrusted sources, like users on the Internet, the use of `eval` could be disastrous.

Summary

- Python is an interpreted language. We can execute commands directly in a shell or write a Python file.
- A Python program is a sequence of commands (statements) for the interpreter to execute. It can take input from the user, print output to the screen and run a set of statements.