Python Lab 1 Summary

Python values, types, expressions, variables and functions

Allen B. Downey, *Think Python: How to Think Like a Computer Scientist (2nd ed)*, 2016.
https://greenteapress.com/wp/think-python-2e/

▶ Study *Think Python* Chapter 1: The way of the program

▶ Study *Think Python* Chapter 2: Variables, expressions and statements

▶ Study *Think Python* Chapter 3: Functions

There is also the Rhodes Local Edition (RLE) which has more exercises:
http://openbookproject.net/thinkcs/python/english3e/

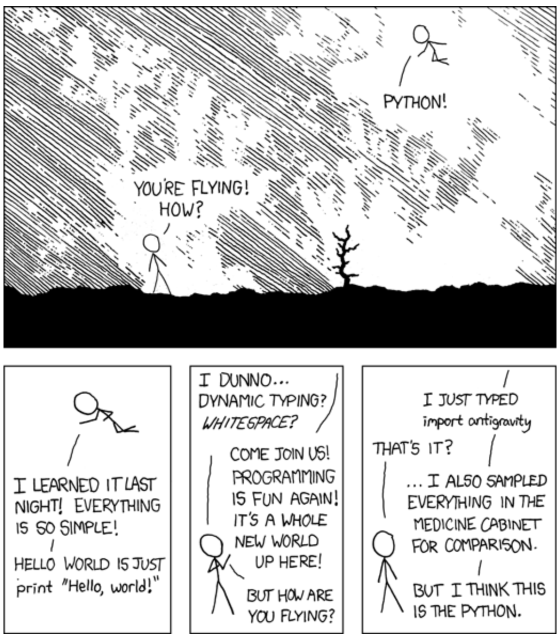
PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
...I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Image from: https://xkcd.com/353/

1. Expressions, Values and Types

An expression is built out of *values* (such as 2, 3 and 5) and *operators* (such as +).

```
>>> 2 - 3/5
1.4
```

Python can *evaluate* an expression to produce a single value.

Every value in Python has a particular *type*.

Values like 4 and 17 have type *int* (short for *integer*).

Values like 2.5 and 17.0 have type *float* (short for floating point, i.e., has a decimal point).

```
>>> type(42)
<class 'int'>
>>> type(3.1415)
<class 'float'>
>>> type('hello')
<class 'str'>
```

2. Variables

A name that refers to a value is called a variable.

Python variable names can use letters, digits, and the underscore symbol (_), but then cannot start with a digit.

You create a new variable by *assigning* it a value.

```
>>> radius = 5.0
```

Variables are called variables because their values can vary as the program executes.

```
>>> radius = 5.0
>>> 3.1415927 * radius ** 2
78.5398175
```

```
>>> radius = 2.7
>>> 3.1415927 * radius ** 2
22.902210783
```

Assigning a value to a variable that already exists doesn't create a second variable. Instead, the existing variable is reused, which means that the variable no longer refers to its old value.

We can create other variables.

```
>>> radius = 2.7
>>> area = 3.1415927 * radius ** 2
```

Warning! – The symbol = does not mean equality. It means assignment.

3. Functions

Python comes with many built-in functions.

```
>>> abs(-9)
9
>>> round(4.7)
5
>>> int(4.7)
4
>>> float(21)
21.0
>>> help(abs)
Help on built-in function abs in module builtins:

abs(x, /)
    Return the absolute value of the argument.
```

Python keeps track of each value in a separate object and each object has a memory address.

You can discover the actual memory address of an object using the built-in function *id*.

```
>>> id(-9)
2100008041808
>>> id('hello')
2100005029808
>>> id(abs)
2099966043784
```

▶ For more mathematical functions we will have to wait until we look at modules (libraries). Most of the useful functions we wish to use are in the math module (library).

Defining our own functions

Rather than use the Python prompt >>> we usually put Python code in a text file and save it with a .py extension. This is easy to do in IDLE and from within the IDLE editor you can run the Python code using F5. Some good advice is NOT to use spaces in filenames (use underscore instead).

The code below is a *function definition* that tells Python what to do when the function is called.

```
def convert_to_celsius(fahrenheit):
    """Convert from degrees Fahrenheit to degrees Celcius"""
    celsius = (fahrenheit - 32) * 5 / 9
    return(celsius)
```

Notice

- The first line is called the *function header* (starts with def and ends with :).
- The function body is indented (four spaces).

- The name of the function is *convert_to_celsius*.
- There is only one parameter which is *fahrenheit* (this is the input to the function). A parameter is a variable in the body of the function.
- Most functions contain a return statement that, when executed, ends the function and produces a value.
- The second line is a *docstring*. This is what is returned if we use the help function on this function.

Summary

- >>> is the Python prompt (or In [1]: in ipython)
- Programs are made up of **statements** (or instructions). These can be simple expressions like 3 + 4 and assignment statements like celsius = 20 (which create new variables or change the value of existing ones). There are many other kinds of statements in Python.
- The following code shows which version of Python you are running.

```
>>> import sys
>>> sys.version
'3.7.9 (default, Aug 31 2020, 17:10:11) [MSC v.1916 64 bit (AMD64)]'
```

- Every **value** in Python has a specific **type**, which determines what operations can be applied to it. The two types used to represent numbers are **int** and **float**. Floating-point numbers are approximations to real numbers (with a decimal point).

```
>>> type(42)
<class 'int'>
```

- Python evaluates an **expression** by applying higher-precedence operators before lower-precedence operators (BEDMAS). You can change that order by putting brackets around subexpressions.

Highest:	**	exponentiation (powers)
	-	negation
	*, /, //, %	multiplication, division, integer division, remainder

Lowest:	+, -	addition, subtraction

- The integer division (or quotient) operator // rounds the result of the division down to an integer value. The remainder (or modulo) operator % gives the value leftover.

- Python stores every value in computer memory. A memory location containing a value is called an **object**.

```
>>> id(42)
140703668545200
```

- Variables are created by executing **assignment** statements (variable = expression). The = is the assignment operator (not equality). If a variable already exists because of a previous assignment statement, Python will use that one instead of creating a new one.

- A variable name must start with a letter or the underscore character.

- A variable name cannot start with a number.

- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).

- Variable names are case-sensitive (age, Age and AGE are three different variables).

- You cannot use certain keywords as variable names. The following code gives a list of Python keywords. So, e.g., you cannot have a variable named continue.

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

- Variables contain memory addresses of values. We say that variables refer to values.

- Variables must be assigned values before they can be used in expressions.

- Syntax error: you type something that is not valid Python code.

- Runtime error: error does not appear until after the program starts running, such as division by zero.
- Semantic error: makes the program do something other than what the programmer intended.
- **Comments** start with # and Python ignores the rest of the line
- Python has a number of builtin functions such as abs(), max(), min(), round(). The following code gives a list including the Python builtin functions.

```
>>> dir(__builtin__)
```

- The help() function provides some help on each function.
- Functions int() and float() convert a number to that type.
- A function definition introduces a new variable that refers to a function object. The function body is indented (use 4 spaces). Leave a blank line after the function body ends.

```
def convert_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5 / 9
```

- The return statement describes the value that will be produced as a result of the function when this function is done being executed. If the function call ends without reaching a return statement, then the function returns the special value None.
- A parameter is a variable that appears between the brackets of a function header.
- A local variable is a variable that is used in a function definition to store an intermediate result in order to make code easier to write and read.
- A function call tells Python to execute a function. A function call is executed by: (1) evaluating each argument one at a time from left to right, (2) passing the resulting values into the function, and (3) executing the function body.

```
>>> print(convert_to_celsius(212))
100.0
```

- An argument is an expression that appears between the brackets of a function call. The value that is produced when Python evaluates the expression is assigned to the corresponding parameter.

- A recipe for designing new functions: (1) create examples, (2) define header, (3) give a description, (4) design the function body, and (5) test.
- Rather than using Python interactively (at the prompt) but Python code in a text file and save it with a .py extension. You can then run your Python code from the commandline.

Exercise.

(a) Write Python code to determine how many days are there in 400 years (beware of leap years). Write further Python code to check whether this number of days is a multiple of 7.

(b) Zeller's algorithm is a formula that can be used to determine the day of the week for a given date (see https://en.wikipedia.org/wiki/Determination_of_the_day_of_the_week#Zeller's_algorithm). Note that the notation $\left\lfloor \frac{a}{b} \right\rfloor$ is found using integer division in Python, i.e., a//b.

Your task is to write Python code to determine the day of the week of 1 January in a given year using Zeller's algorithm. For a particular year, find the day of the week using Python and check the calculation by Python, by hand and also confirm the answer by Google search.

(c) Is there any difference between the calendars for 1971 and 2021?

Exercise.

(a) Write a Python function that converts a distance in miles to a distance in kilometres. Note that one mile is exactly 25146/15625 km).

(b) Write a Python function that takes three arguments (the number of hours since midnight, the number of minutes in the hour, and the number of sections in the minute) and returns the number of seconds since midnight.

(c) Write a Python function that converts from degrees Fahrenheit to degrees Celsius.