



**UCT Department of Computer
Science Computer Science 1015F**

Python Basics



Alan Berman

*(thanks to Hussein Suleman hussain@cs.uct.ac.za
and Brian DeRenzi <bderenzi@cs.uct.ac.za>)*

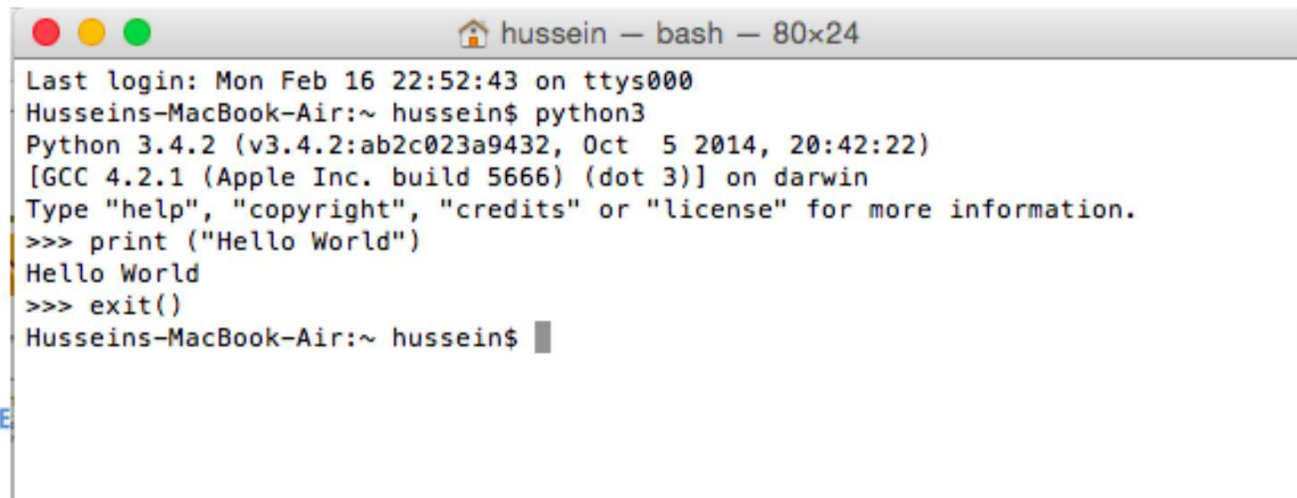
February 2018

Problem 1 Introduction

- Write a program to print out the lyrics of the Drake song “God’s Plan”.

Python Interpreter

- ❑ Python programs are **run** one instruction at a time by the interpreter.
- ❑ We can run the interpreter (usually “python3”) and then enter instructions one at a time.
- ❑ Python will execute each instruction then **prompt** the user for the next one.
- ❑ Enter `exit()` to end.

A screenshot of a macOS terminal window titled 'hussein — bash — 80x24'. The window shows the execution of the Python 3.4.2 interpreter. The user runs 'python3', which displays version information and a prompt 'Type "help", "copyright", "credits" or "license" for more information.'. The user then enters '>>> print("Hello World")', which outputs 'Hello World'. Finally, the user enters '>>> exit()', which returns to the shell prompt 'Husseins-MacBook-Air:~ hussein\$'.

```
hussein — bash — 80x24
Last login: Mon Feb 16 22:52:43 on ttys000
Husseins-MacBook-Air:~ hussein$ python3
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  5 2014, 20:42:22)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>> exit()
Husseins-MacBook-Air:~ hussein$
```

Integrated Development Environment (IDE)

- ❑ Graphical interface with menus and windows, much like a word processor.
- ❑ We recommend WingIDE 101
 - free, scaled-down Python IDE designed for use in teaching introductory programming classes.
- ❑ You are welcome to use any other tools – e.g. IDLE
 - even a separate text editor and interpreter if you wish to simply do things the hard way.



Python program files

- ❑ The interpreter is useful for testing code snippets and exploring functions and modules.
- ❑ However, to save a program permanently we need to write it into a file and save the file.
- ❑ Python files are commonly given the suffix “.py”
 - e.g. `HelloWorld.py`
- ❑ Always save your programs while you are working!

Skeleton Python Program

```
# what the program does  
# author of program  
# date
```

```
PythonInstruction1  
PythonInstruction2  
PythonInstruction3  
PythonInstruction4
```

Example Program

helloworld.py:

```
# first program ever  
# Noah Buddy  
# 20 february 2018  
  
print ("Hello World")
```

output:

```
Hello World
```

What does it mean?

```
print ("Hello World")
```

- ❑ **print** is the name of a function.
- ❑ “Hello World” is the argument or value sent to this function.
- ❑ A function is a common task that we can reuse.
 - Printing on the screen is quite complicated.
 - So we reuse Python's built-in function to make it easier.
- ❑ In a few weeks we will also create our own functions.

Problem 1

- Write a program to print out the lyrics of the Drake song “God’s Plan”.

Problem 2 Introduction

- Write a program to print out your initials in ASCII art.

Program Syntax and Style

- Generally, every statement starts on a new line.

- Case-sensitivity

 - STUFF vs stuff vs STuff vs stUFF

- Everything after # is a (*for humans*) comment.

 - # a simple program

 - print ("Hi") # write Hi on screen

Comments

- ❑ Brief description, author, date at top of program.
- ❑ Brief description of purpose of each function (if more than one).
- ❑ Short explanation of non-obvious parts of code.

```
# test program to print to screen
```

```
# Alan Berman
```

```
# 20 february 2018
```

```
print ("Hello World")
```

```
...
```

Syntax and Logic Errors

- ❑ Syntax errors are when your program does not conform to the structure required.
 - e.g., print spelt incorrectly
 - The program will not start at all.

- ❑ Logic errors are when your program runs but does not work as expected.
 - You **MUST** test your program.

Escape sequences

- ❑ Escape sequences are special characters that cannot be represented easily in the program.
 - For example ... tabs, newlines, quotes

\a - bell (beep)

\b – backspace

\n – newline

\t – tab

\' – single quote

\” – double quote

\\ - \ character



Problem 2

- Write a program to print out your initials in ASCII art.

Problem 3 Introduction

- Write a program to calculate the number of precious seconds you spend at lectures in a semester, assuming you have 5 lectures a day, lectures on 4 days a week, and there are 12 weeks in a semester.
- $A \text{ second/minute} * B \text{ minute/lecture} * C \text{ lecture/day} * D \text{ day/week} * E \text{ week/semester}$
- $= (A*B*C*D*E) \text{ seconds/semester}$

Numeric Data Types

For numbers we have two primitive types:

- integer:

- whole number with no fractional part

- floating point number:

- number with fractional part
- WARNING: stores only an *approximation* to real number
- there is a limit to the precision, or accuracy, of the stored values
 - e.g. $10/3$



Integers: Literals

- Literals are actual data values written into a program.
- Numerical literals can be output just like text, but after sensible conversions:
 - `print (12)`
 - 12
 - `print (12+13)`
 - 25
 - `print (2+2/2)`
 - 3.0

Integers: Expressions

- Common operations
 - + (plus), - (minus), / (divide), * (times), % (mod)
 - // (integer division)
 - ** (a to the power b)
- $11 + 11 // 2 = 16$... how ?
 - precedence of operators:
 - high: ()
 - middle: * / %
 - low: + -
 - left associative if equal precedence.
 - integer operations when both “operands” are integers.

Integers: Quick Quiz

□ What is the value of each expression:

■ $(34 + 12)$

■ $(4 - 3) / (2 + 1)$

■ $5 \% 2 + 2 \% 5$

■ $1 // 1 // 2 // 3$

■ $((1 // 2) // 3) // 4$

Problem 3

- Write a program to calculate the number of precious seconds you spend at lectures in a semester, assuming you have 5 lectures a day, lectures on 4 days a week, and there are 12 weeks in a semester.

Problem 4 Introduction

- Write a program to calculate your subminima and final mark for CSC1015F. Initialize variables for each component mark at the top of the main method so the marks can be changed relatively easily.

Identifiers

- ❑ In source file, *print* is an **identifier**.
- ❑ Identifiers are used to name parts of the program.
 - start with `_` or letter, and followed by zero or more of `_`, letter or digit
 - preferred style: `hello_world`, `my_first_program`
- ❑ **Reserved words:**
 - `if`, `for`, `else`, ...
 - These identifiers have special meaning and cannot be used by programmers for other purposes.

Identifiers: Quick Quiz

- ❑ Which are valid identifiers:
 - 12345
 - JanetandJustin
 - lots_of_money
 - “Hello world”
 - J456
 - cc:123

- ❑ Which are good Python identifiers?

Variables

- ❑ Variables are sections of memory where data can be stored.
- ❑ Most variables have names (identifiers) by which they can be referred.
 - e.g., `a_value`, `the_total`
- ❑ Variables are defined by assigning an initial value to them.
 - `a_value = 10`
 - `the_total = 12.5`

Assignment and Output (I/O)

▣ Putting a value into a variable:

`a = 1`

`b = a + 5`

`c = 1`

- LHS is usually a variable, RHS is an expression

▣ Output values of variables just like literals

- Can do both on one line

- `print ("The value is ", a)`

Problem 4

- ❑ Write a program to calculate your subminima and final mark for CSC1015F. Initialize variables for each component mark at the top of the main method so the marks can be changed relatively easily.
- ❑ $\text{Final} = 0.6 * \text{exam} + 0.15 * \text{tests} + 0.15 * \text{practicals} + 0.10 * \text{practests}$
- ❑ $\text{Theory subminimum} = 0.80 * \text{exams} + 0.20 * \text{tests}$
- ❑ $\text{Practical subminimum} = 0.60 * \text{practicals} + 0.40 * \text{practests}$

Problem 5 Introduction

- Rewrite the previous program to get the user to input the marks instead of just storing them in variables.

Problem 6 Introduction

- ❑ Write a program to add two numbers entered by the user and print the result on the screen.

Output: `print`

- `print` is a built-in function with a precise set of rules for how it works.

- The general format is one of the 2 options:

```
print(<expr1>, <expr2>, ..., <exprn>)
```

```
print()
```

- By default, `print` displays the value of each expression, separated by blank spaces and followed by a carriage return (moving to the next line).



Options for `print`

- ▣ To print a series of literals or values of variables, separated by spaces:

```
print("The values are", x, y, z)
```

- ▣ To suppress or change the line ending, use the `end=ending` keyword argument. e.g.

```
print("The values are", x, y, z, end='')
```

- ▣ To change the separator character between items, use the `sep=sepchr` keyword argument. e.g.

```
print("The values are", x, y, z, sep='*')
```



Quiz: What is the exact output?

```
x=20
```

```
y=30
```

```
z=40
```

```
print(x)
```

```
print(y, '\n', z)
```



Quiz: What is the exact output?

```
x=20
```

```
y=30
```

```
z=40
```

```
print("The values are", x, y,  
      z, end='!!!', sep='!***')
```

input

- ❑ The purpose of the Python input statement is to get a text string from the user; this can be stored in a variable.
- ❑ We get input using the input function.
 - `input (<prompt>)`
 - prompt is usually some text asking the user for input

- ❑ For example:

```
weather = input("How is the weather today?")  
print ("Weather:", weather)
```

Quiz: What is the exact output?

```
name = input("What is your name? ")  
print("Hellooooo", name, end='!!!', sep='...')
```



Inputting numbers

- ❑ The input function always gives us a string (text).
- ❑ This must be converted into a number if you are going to do math.
 - text (strings) and numbers are handled differently by the computer – more about this later!
- ❑ `eval()` is a Python function that converts a string into a number.
 - `eval("20") -> 20`
- ❑ `int()` and `float()` also work if we know what kind of number it is.

Sample program

Useful example using eval...

```
# algorithm to calculate BMI, using input
height = input("Type in your height: ")
weight = input("Type in your weight: ")
height = eval(height) # change a string into a number
weight = eval(weight) # change a string into a number
BMI = weight/(height*height) #calculates BMI
print("Body mass index = ",BMI)
```



Problem 6

- ❑ Write a program to add two numbers entered by the user and print the result on the screen.

Problem 5

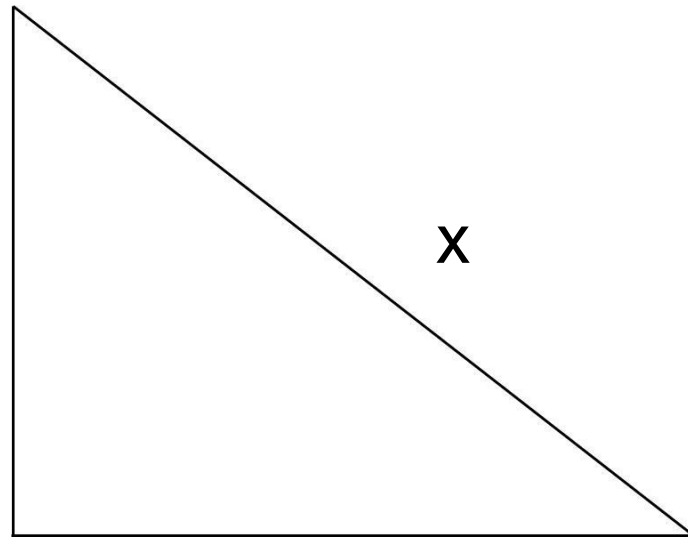
- ▣ Rewrite the program to calculate marks and subminima to get the user to input the marks instead of just storing them in variables.

Problem 7

- ❑ Suppose there is a *min* function that returns the minimum of 2 numbers.
- ❑ Write a program to calculate the minimum of 3 integer values using such a function.

Problem 8 Introduction

- ❑ Use the **math library** for this question.
- ❑ Write a program to calculate "x", the length of the hypotenuse on a right-angle triangle.



Other Useful Numerical Syntax

- ❑ Increment/decrement operators
- ❑ Implicit type conversions
- ❑ Explicit type conversions
- ❑ math Module

Increment / Decrement

□ $a += 3$

■ equivalent to $a = a + 3$

□ Also ...

■ $a -= b$

■ $a *= b$

■ $a /= b$

Implicit Conversions

- ❑ If there is a type mismatch, the narrower range value is promoted up
 - `a=1`
 - `b=2.0`
 - `print (a+b)`

- ❑ Cannot automatically convert down
 - i.e., floats do not become ints

Explicit Conversions

- ❑ Typecast methods cast (convert) a value to another type.

```
a = int (1.234)
```

```
b = float (1)
```

- ❑ Use *math.ceil*, *math.floor*, *round* methods for greater control on floating-point numbers.

- ❑ Use *eval ()*, *int()*, *float()* to convert strings to numbers.

- `eval ("1")`

- `int ("1")`

- `float ("1")`

math Module

- ❑ Python has collections of useful functions in **modules**.

- ❑ To use a module:
 - `import math`

- ❑ Then, to use its functions:
 - `math.sqrt (a)`

- ❑ Alternatively, to avoid saying "math.", use:
 - `from math import sqrt`
 - `sqrt (a)`

Sample math functions

`math.pow(x, y)`

Return `x` raised to the power `y`. Exceptional cases follow Annex 'F' of the C99 standard as far as possible. In particular, `pow(1.0, x)` and `pow(x, 0.0)` always return `1.0`, even when `x` is a zero or a NaN. If both `x` and `y` are finite, `x` is negative, and `y` is not an integer then `pow(x, y)` is undefined, and raises `ValueError`.

Unlike the built-in `**` operator, `math.pow()` converts both its arguments to type `float`. Use `**` or the built-in `pow()` function for computing exact integer powers.

`math.sqrt(x)`

Return the square root of `x`.

9.2.3. Trigonometric functions

`math.acos(x)`

Return the arc cosine of `x`, in radians.

`math.asin(x)`

Return the arc sine of `x`, in radians.

`math.atan(x)`

Return the arc tangent of `x`, in radians.

`math.atan2(y, x)`

Return `atan(y / x)`, in radians. The result is between `-pi` and `pi`. The vector in the plane from the origin to point `(x, y)` makes this angle with the positive X axis. The point of `atan2()` is that the signs of both inputs are known to it, so it can compute the correct quadrant for the angle. For example, `atan(1)` and `atan2(1, 1)` are both `pi/4`, but `atan2(-1, -1)` is `-3*pi/4`.

`math.cos(x)`

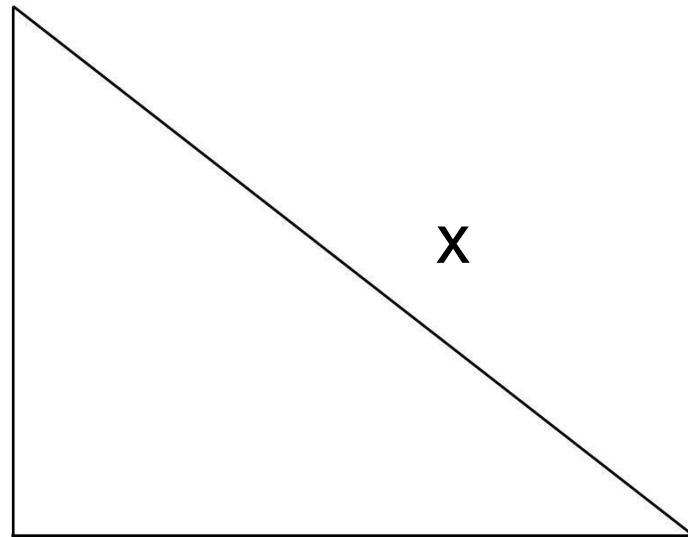
Return the cosine of `x` radians.

`math.hypot(x, y)`

Return the Euclidean norm, `sqrt(x*x + y*y)`. This is the length of the vector from the origin to point `(x, y)`.

Problem 8

- ❑ Use the **math library** for this question.
- ❑ Write a program to calculate "x", the length of the hypotenuse on a right-angle triangle.



Problem 9

- ❑ Write a program to convert your dog's age into human years. Your program must ask for a dog years number and then output the human years equivalent.
 - The formula is: 10.5 human years per dog year for the first 2 years, then 4 human years per dog year for each year after.
 - ❑ [source:
<http://www.onlineconversion.com/dogyears.htm>]
 - Now do it the other way around ... human->dog

Problem 10

- Write a program to calculate compound interest, according to the formula:

$$A(t) = A_0 \left(1 + \frac{r}{n}\right)^{n \cdot t}$$

- A_0 = Initial sum, R = Annual interest rate, N = Number of periods per year, T = Number of years
- Use floating point numbers.

Problem 11

- ❑ Write a program to simulate Eliza, the artificial conversationalist. For example:
 - What is your name?
 - Brian
 - Tell me, Brian, what is your problem today?
 - My students are bored.
 - You said your students are bored. Why do you say that?
 - They have that bored look.
 - Are you sure they have that bored look?
 - Yes
 - . . .