

CEBD 1100 Introduction to Data Analysis and Python

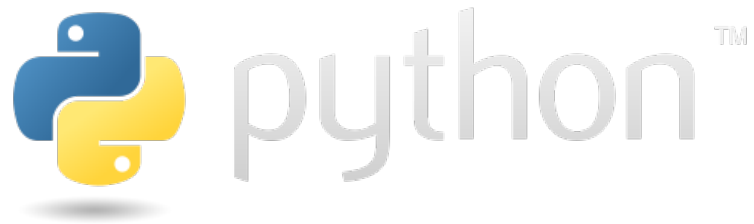
Introduction



What is Python

“Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.”

The Python Tutorial, <http://docs.python.org/tutorial/>



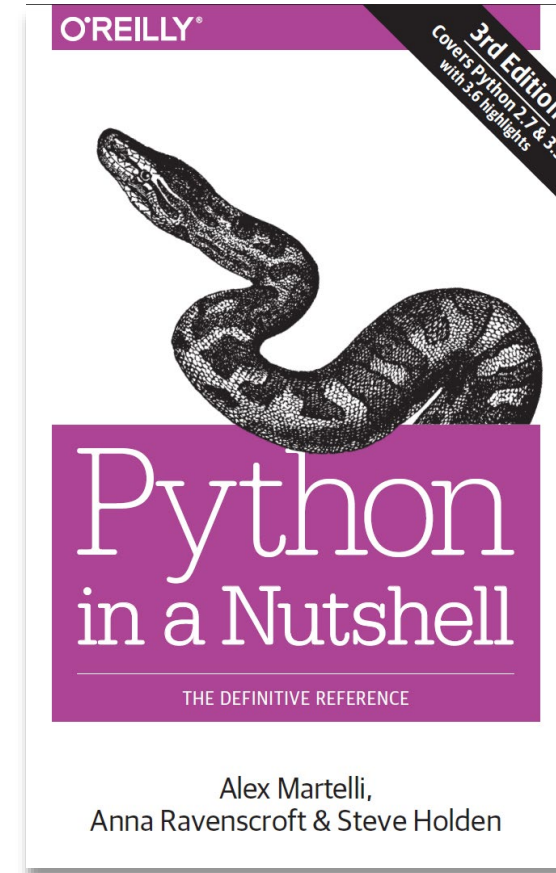
Interesting Fact

Guido van Rossum, the creator of the Python language, named the language after the BBC show "Monty Python's Flying Circus".



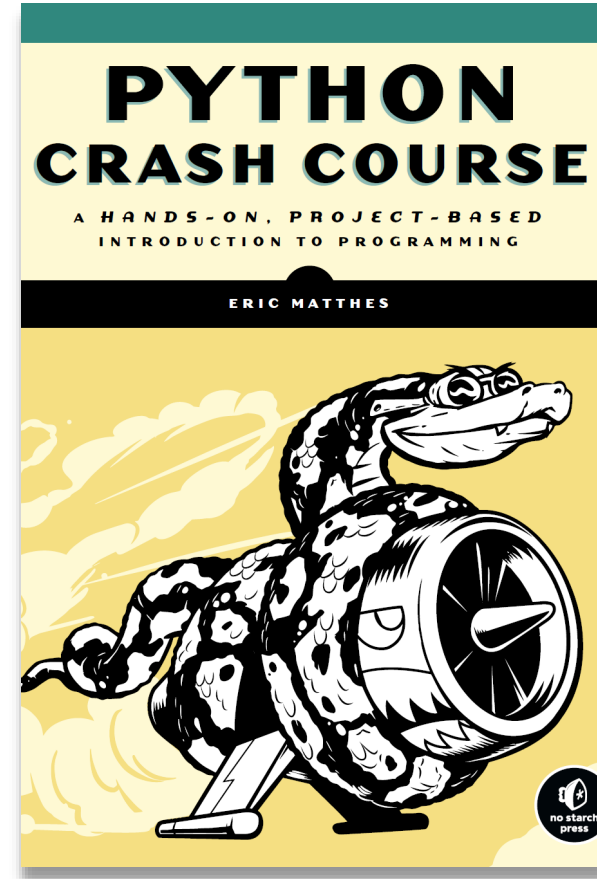
Recommended Book #1

- **Reference**
- You can't read through this casually.
- Refer to this text for specific functions and problems you need, as you study.



Recommended Book #2

- Learning Python
- To further your knowledge of Python, or to study more of what we do in class, this book is known as one of the better & most popular Python books.



Python, Python IDE and GIT - Setup


Downloading and Installing Python

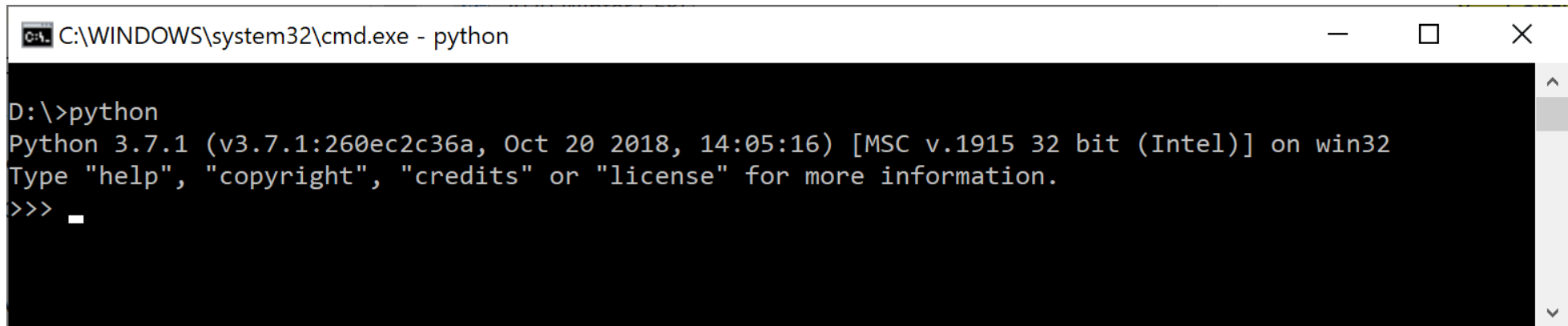
- Download the Python engine
- <https://www.python.org/downloads/>
- For Windows users, just click on the yellow button.
- Note: There are MAC and Linux versions also, which are exactly the same on other operating systems.
- Windows users, it's recommended to download the 32 bit version.

Download the latest version for Windows

Download Python 3.7.1

Command-Line Python

- Start Python via the Start menu. +R, "cmd", ENTER.
- Type the command "python" to start python.
- To quit, type "quit()"
- Running a Python File
 - Type "python filename.py".



```
C:\WINDOWS\system32\cmd.exe - python

D:\>python
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```


Running a Python File

- Run a python file
 - `python name_of_file.py`
- Run a python file with a parameter
 - `python name_of_file.py 1 2 3`

How to Read Parameters

- Use the "print()" function to print out values.
- Include the sys library : `import sys`
- Get the number of parameters : `print(len(sys.argv) - 1)`
- Get the name of the calling program : `print(sys.argv[0])`
- Get the first parameter: `print(sys.argv[1])`
- Get the second parameter: `print(sys.argv[2])`

Exercise

1. In a file, include some print commands like
`print("hello world")`
`print(1+2)`
2. Run the file.
3. Change the file so that it prints out the file name, and the first parameter.



Python Basics



Variables and Constants

Variables / Constants

- Variables are simply names which point to any value or object:
 1. `a_string = "hello, world"`
 2. `an_integer = 12`
 3. `a_float = 3.14`
 4. `a_boolean = True`
 5. `nothing = None`
- Dynamic typing: the value or type of a variable may be changed at any time.



String Values & Console Output

Printing to the Console

- To print to the console, use the "print" command.
- Everything printed must be within parentheses.
 - `print("hello")`
 - `print(12)`
 - `print(variable_name)`
- Printing a list
 - `print("a", "b", 12, variable_name)`

Quotes

- Single and double quotes are equivalent
 - Single Quotes
 - `a = 'abc'`
 - Double Quotes
 - `a = "abc"`
- Triple Quotes (Multiline Text Alternative)
 - Allow you to assign a string over several lines long.
 - `a = """a`
 - `b`
 - `c"""`

Python Backslash Escape Characters

- `\newline`
 - Ignore the backslash and newline (see next slide)
- `\\`
 - Backslash `"\"`
- `\' or \"`
 - Quote (`'`) or double quote (`"`)
- `\f and \n`
 - Form feed and line feed (create a new line)
- `\t`
 - Tab character

Quotes

- Single Quotes

- Useful to include quotes in your string.

```
a = 'Hello "world"'
```

- To escape characters use backslashes.

```
a = 'don\'t'
```

- Double Quotes

- To escape characters use backslashes.

```
a = "this is a quote : \" "
```

```
filePath = "C:\\Windows\\System32" (not "C:\Windows\System32" !!!)
```

Line Continuation

- A line may be continued to the next line by using a backslash character.

```
t = t + \  
"A line of text"
```

String Operators and Operations

- String concatenation

```
string = string1 + " : " +  
string2
```

- String repetition

```
string1 * 3
```

- The string methods can be found at:

- https://www.w3schools.com/python/python_ref_string.asp

Example (Concatination)

```
a = "Hello"  
b = "Concordia"  
c = a + " " + b  
print(c)
```

OUTPUT

```
Hello Concordia
```

Example (Repetition)

```
number_boxes = 5  
bar_of_boxes = "#" *  
number_boxes  
print(bar_of_boxes)
```

OUTPUT

```
#####
```



String Formatting

Print with Format Method

<https://www.geeksforgeeks.org/python-output-formatting/>

Using placeholders

```
print('{} {}'.format('Hello', Everyone))
```

Using positional placeholders

```
print('{1} {0}'.format('Hello', Everyone))
```

Using positional placeholders

```
print('{0} {1}'.format('Hello', Everyone))
```

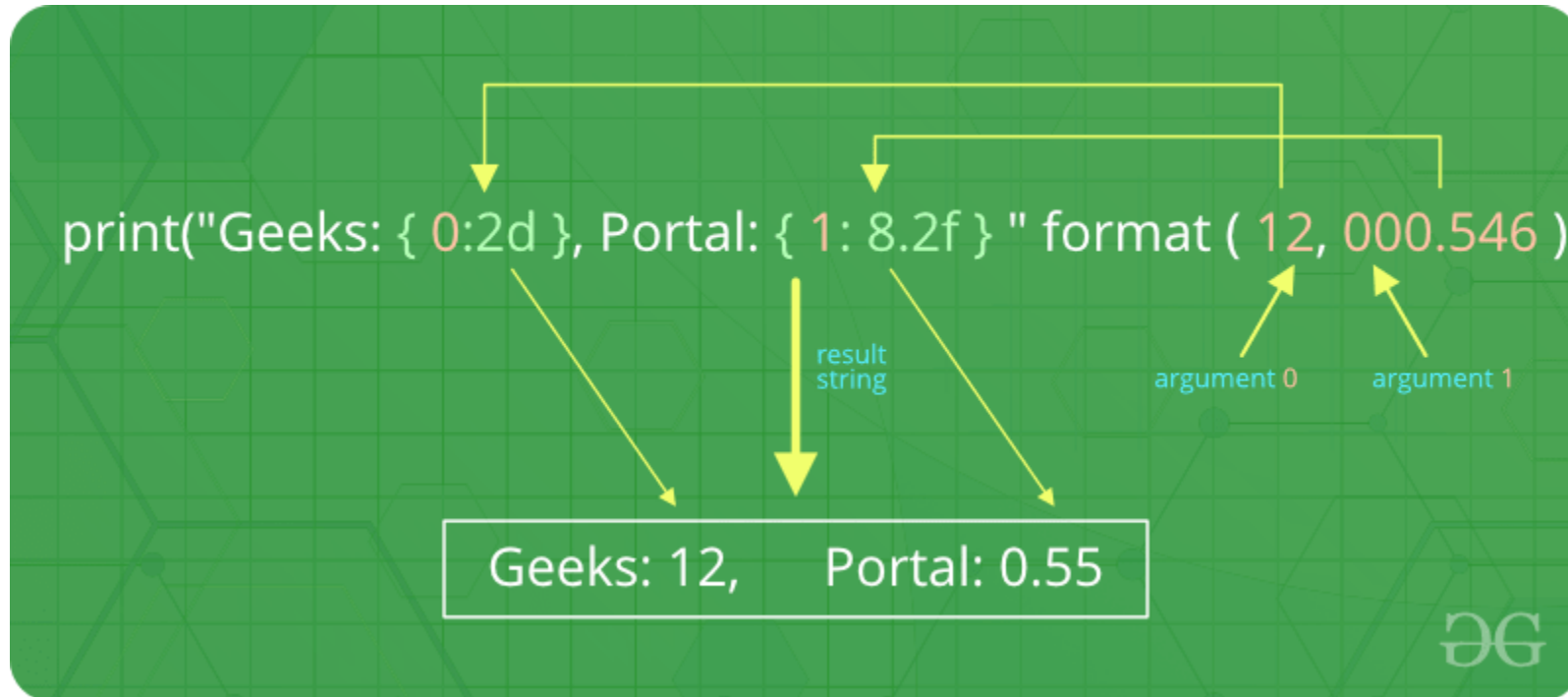
Print with Format Method

Printing numbers

```
print("Gas price is {0:2d}, and cost per litre is  
:{1:8.2f}".format(12, 00.546))
```

- 2d means, two number places, decimal. If one number is supplied, will put a space.
- 8.2f means, 8 numbers, 2 after decimal, type of “float”.

Explained



Obtained from: <https://www.geeksforgeeks.org/python-output-formatting/>

More (new) format method examples - Money

To print money using comma separators for 1000's and 2 decimal places.

Note: This is NOT region friendly. For example, in Québec, the amount \$1500.00 is shown differently as 1500,00\$

We can use other special tools to make the money amounts regional-friendly.

```
formatted_float = "${:,.2f}".format(1500.2)
```

Exercise: String Formatting & Concatenation

Use the supplied python file and produce the expected results using string formatting.



String Related Functions

Converting String and Number

`str(x)`

- Converts most numbers to a string.

`int(x)`

- Converts a string number to a number (int). Must be a whole number.

`float(x)`

- Converts a string number to a float (int).

`decimal.Decimal(x)`

- Converts a string to a decimal. You must import decimal (import decimal).

Common String Functions

`len(string)`

- Length of string

`string = string.upper()`

- Case change (See also lower, title and capitalize).

`string = string.strip()`

- Strip leading and trailing spaces. This is good to filter input from a user, in case they give us extra spaces.

`string =
string.replace('abc',
'xyz')`

- Replace text in the string.

Print on Same Line

- Print on the SAME LINE, across multiple PRINT statements.
- Useful when generating reports, interacting with user via command line.

```
print('*', end="")  
print('*', end="")
```

Output: **

String Slices (Substrings)

Hello

0	1	2	3	4
-5	-4	-3	-2	-1

- The "slice" syntax is a handy way to refer to sub-parts of sequences -- typically strings and lists. The slice `s[start:end]` is the elements beginning at start and extending up to but not including end.

Suppose we have `s = "Hello"`

`s[1:4]` is 'ell' Chars starting at index 1 and extending up to but not including index 4

`s[1:]` is 'ello' Omitting either index defaults to the start or end of the string

`s[:]` is 'Hello' omitting both always gives us a copy of the whole thing (a way to copy a sequence like a string or list)

`s[1:50]` is 'ello' An index that is too big is truncated down to the string length

String Slices (Substrings)

Hello

0	1	2	3	4
-5	-4	-3	-2	-1

- The standard zero-based index numbers give easy access to chars near the start of the string. As an alternative, Python uses negative numbers to give easy access to the chars at the end of the string: `s[-1]` is the last char 'o', `s[-2]` is 'l' the next-to-last char, and so on. Negative index numbers count back from the end of the string:

Suppose we have `s = "Hello"`

`s[-1]` is 'o' last char (1st from the end)

`s[-4]` is 'e' 4th from the end

`s[:-3]` is 'He' going up to but not including the last 3 chars.

`s[-3:]` is 'llo' From the 3rd char from the end and extending to the end of the string.

Working with Money

- To work with money in a more regional way (Internationalization), we can use the babel package and “format_money” function.

```
from babel.numbers import format_currency

print (format_currency(1099.98, 'USD', locale='en_US')) # $1,099.98
print (format_currency(1099.98, 'USD', locale='es_CO')) # 1.099,98 US$
print (format_currency(1099.98, 'EUR', locale='de_DE')) # 1.099,98 €
```

Currency codes: <https://www.xe.com/iso4217.php>

Language codes (PDF doc): <https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-LCID/%5bMS-LCID%5d.pdf>



Math and calculations

Math functions

****** Exponent

+ - / * Plus, Minus, Division and Multiplication.

% Modulus (Remainder of a division)

() Brackets may enforce order of operation.

+= Increment

-= Decrement

***=** Multiplication (change your source to the new value)

/= Multiplication (change your source to the new value)

Statements using simple arithmetic (integer)

```
# Integer arithmetic
```

```
x = 14
```

```
y = 8
```

```
result1 = x + y
```

```
# result1 = 22
```

```
result2 = x - y
```

```
# result2 = 6
```

```
result3 = x * y
```

```
# result3 = 112
```

```
result4 = x / y
```

```
# result4 = 1
```

Statements using simple arithmetic (double)

```
# Double arithmetic
```

```
a = 8.5
```

```
b = 3.4
```

```
result1 = a + b
```

```
# result1 = 11.9
```

```
result2 = a - b
```

```
# result2 = 5.1
```

```
result3 = a * b
```

```
# result3 = 28.9
```

```
result4 = a / b
```

```
# result4 = 2.5
```


Exercise

- Write a program that will print the result of following operations:

```
# Integer arithmetic
x = 14
y = 8
```

```
result1 = x + y      # result1 = 22
result2 = x - y      # result2 = 6
result3 = x * y      # result3 = 112
result4 = x / y      # result4 = 1
```

```
# Double arithmetic
a = 8.5
b = 3.4
```

```
result1 = a + b      # result1 = 11.9
result2 = a - b      # result2 = 5.1
result3 = a * b      # result3 = 28.9
result4 = a / b      # result4 = 2.5
```

Arithmetic Operations

OPERATOR	DESCRIPTION	SYNTAX
+	Addition: adds two operands	$x + y$
-	Subtraction: subtracts two operands	$x - y$
*	Multiplication: multiplies two operands	$x * y$
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	$x // y$
%	Modulus: returns the remainder when first operand is divided by the second	$x \% y$

Exercise

a = 9

b = 4

Addition of numbers

add = a + b

Subtraction of numbers

sub = a - b

Multiplication of number

mul = a * b

Division(float) of number

div1 = a / b

Division(floor) of number

div2 = a // b

Modulo of both number

mod = a % b

print results

Exercise

```
a = 9  
b = 4
```

```
# Addition of numbers  
add = a + b  
# Subtraction of numbers  
sub = a - b  
# Multiplication of number  
mul = a * b  
# Division(float) of number  
div1 = a / b  
# Division(floor) of number  
div2 = a // b  
# Modulo of both number  
mod = a % b  
# print results
```

```
# print results  
print(add)  
print(sub)  
print(mul)  
print(div1)  
print(div2)  
print(mod)
```

```
13  
5  
36  
2.25  
2  
1
```

Exercise

Given the year of manufacture of a car, write a program to calculate how many years have elapsed since production. Print the results in a nice format.

1. Calculate the number of months elapsed since production.
2. How about the number of days?

Exercise – Temperature converter

Write a program to convert Celsius to Fahrenheit and Fahrenheit to Celsius for the following values and print the result on the console:

1) Celsius to Fahrenheit

$$Fahrenheit = \frac{9}{5} \times Celcius + 32$$

2) Fahrenheit to Celsius

$$Celcius = (Fahrenheit - 32) / \left(\frac{9}{5}\right)$$

Exercise 1 – Temperature converter (Samples)

Celsius	Fahrenheit
0	32
10	50
15	59
20	68
30	86
40	104



Getting Input

Getting Input

- Use the "input" function to get input from the user.
- The parameter to the function is the prompt that will be displayed.

```
the_result =  
    input("\nFirst Number: ")
```

Input Practice

- Rental Car: Write a program that asks the user what kind of rental car they would like. Print a message about that car, such as “Let me see if I can find you a Subaru.”

Homework

1. Revisit your temperature convertor to get the temperature from the user before converting it.
2. Write a program to get the user's first name and last name and print the initials.



CONCORDIA.CA

