# Data Analysis and Hypothesis Testing Using the Python ecosystem

## Python Basics

*Stavros Demetriadis*

*sdemetri@csd.auth.gr*
*http://mlab.csd.auth.gr/sdemetri*

- Key ideas
- Data types & I/O
- Flow control

# Python basics: **Key ideas**

- What is Python
  - High-level, general-purpose, high-readability programming language
- What to install
  - Python IDLE, Anaconda (Jupyter notebook)
- Language execution model
  - Interpreted, Python Virtual Machine (PVM)
- Namespaces and Modules
  - Object reference, Import
- Variables and Dynamic typing
  - Assignment, type properties
- Where do types come from?
  - Class hierarchy, type, id, dir

# Python basics: **Basic Data Types and I/O**

- Data types
  - Numerics (Integer, Float, Complex)
    - Integers & Arithmetic operators
    - Complex & Precision limitations
  - Strings
    - Ordered sequence, Zero-indexing, len() function
  - random & math modules
    - 'import random' for generating and managing pseudorandom numbers
    - 'import math' for basic mathematical functions
  - Booleans
    - Comparison & Boolean operators
- I/O commands
  - Print
    - print.format() & backslash
  - Input
    - int(), float, split() methods

# Python basics: **Flow control**

- 'while' loop
  - ‘while/else’ loop, Indentation, Blocks
- 'if' control
  - ‘if/elif/else’ control, Conditional expressions
- Break, continue, pass
  - Examples
- 'for' loop
  - ‘for’ loop, iterator, iterable. Examples with various iterable structures
- range function
  - How range() works. Examples with chr() and ord() functions

# Review exercises

- basic data types, I/O and flow control

# Leap years

- Write a script that takes as input a year number (four digits) and identifies **whether the year is leap or not**.

- *Hint*: A leap year is one that is divisible by 4 except from century years (1800, 1900, etc.) which should be divisible by 400

- *Addition:* Adjust your code to run repetitively. After each repetition the code prompts the user with the following prompt:

  “Enter to continue, ‘q’ to quit”

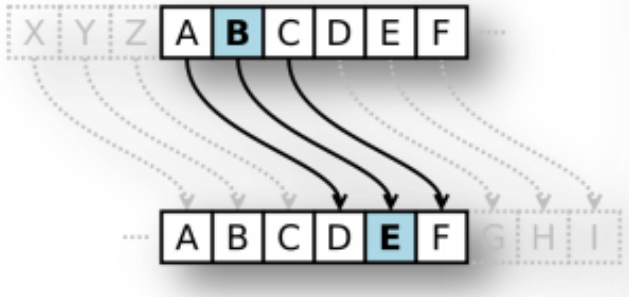
# Multiplication table

- Write a script that prints in the output the **multiplication table** as appears below
- *Hint*: develop two versions of the code; one using the **print.format()** and another with the **'\t'** escape sequence

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 |
| 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |
| 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 | 60 |
| 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | 70 |
| 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |
| 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | 90 |
| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |

# Caesar's cipher

- Write a script that takes as input: (a) a text message, (b) an integer. Then the script **shifts each character** in the message as many positions as the integer indicates (either in the ASCII or in the Unicode table). Finally, the script prints out the encrypted message.




- *Hint*:
  - Use the **chr()** and **ord()** functions to change from numeric code to character and vice versa.
  - See ASCII table here
  - See UNICODE table here

# Lists

# Python basics: **List** 1/2

- The 'list' object
  - List features: ordered, indexed, iterable, mutable, heterogeneous
  - List indexing and assignment, Deleting a list / Item in a list
  - List is 'mutable', List as iterable in a for loop (enumerate), the 'in' operator
  - The 'list' constructor method, List vs. String
- Constructing lists
  - (1) list & range()
  - (2) append()
  - (3) List comprehension
- List slicing
  - Slicing examples
  - The 'step' value, Negative indexes
  - Slicing and value assignment

# Python basics: **List** 2/2

- List of lists
  - Lists with lists as member items
  - 2D lists: square, rectangular, ragged
- List functions & methods
  - *Functions*: len, max, min, sorted, sum
  - *Methods*: append, extend, insert, remove,…

# Review exercises

- Lists

# Pseudorandom numbers

- A) Write a script to construct a list with 100 pseudorandom integers in [1,50] using the **append** method. Next, find the minimum and maximum number in the list writing your own code. Also, find the place (index) in the list where these numbers are located.
- B) Change the method you construct the list and use **list comprehension**. Now, use the built-in list functions to identify the aforementioned numbers and their location in the list (index).

# Clear multiple numbers

- Use list comprehension to construct a list with 20 pseudorandom integers in the [1,2-] interval. Then check for integers with multiple appearance and construct a new list containing each number in the original list but only once. Sort and print out the new list and print also the number of different (unique) numbers.

- *Hints*:
- Use methods **count()** and **sort()**
- See more about these methods here

# Wins in ‘2D’ list

- Write a script that constructs a ‘2D’ list and populates it with pseudorandom integers in [1,3].

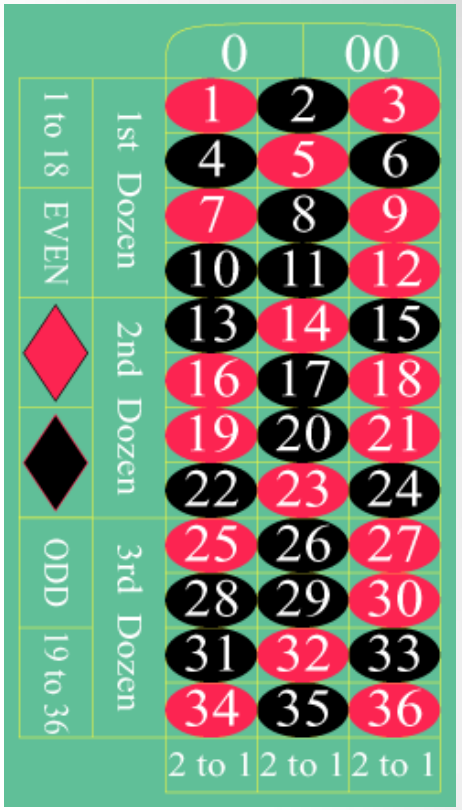
```
[1, 3, 1]
[1, 2, 3]
[2, 2, 2]
Wins:  1
Enter to play / "q" to quit)
```

*Example output*

- Next, the script checks for ‘wins’ in the rows, columns and diagonals of this list representation. As ‘win’ is considered a situation where the same number appears in all positions of a row, column or diagonal.
- The script prints out the list (as in the image), the results of the ‘win’ control  and prompts the user to play or quit (see image above)

# Roulette

- Write a script that simulates the roulette game as follows: the script runs repeatedly and in each repetition a pseudorandom integer number is drawn in [0,36] interval (ignore '00'). The script prints out: the random integer and whether it is:
  - A) 'Low' (<18) or 'High' (>18)
  - B) 'Even' or 'Odd'
  - C) 'Red' or 'Black' (see image on the right)




- *Hints*:
- Use a list with members the 'red' or 'black' numbers and check for membership using the 'in' operator.
- Use a list with four items and place there for each draw: the drawn integer and each of the above features (string items). Print this list onscreen to present the draw result.
- Write your code to run repetitively. After each repetition the code prompts the user with the following prompt:

"Enter to continue, 'q' to quit"

# Dictionary & Tuple

# Python basics: **Dictionary & Tuple** 1/3

- The 'dictionary' object
  - Features: mapping, collection, indexed, unordered, iterable, mutable
  - Dict indexing and assignment
  - Dict as mutable, Dict as iterable
- Constructing dictionaries
  - (1) Assigning pairs
  - (2) Using the dict() constructor method
  - (3) Dictionary comprehension
  - Conditional expressions in dict comprehensions

# Python basics: **Dictionary & Tuple** 2/3

- Combining dictionary and list
  - Dictionary with lists as values
  - Dictionary with dictionaries as values
  - Lists of dictionaries

- Dictionary vs. List
  - Dict - List comparison
  - Dict - List use
  - Dictionary as a hash table
  - Efficiency of the 'in' operator

# Python basics: **Dictionary & Tuple** 3/3

- Dictionary functions and methods
  - *Functions*: len, max, min, sorted, sum
  - *Methods*: keys, values, get, items,…
- The 'tuple' object
  - Tuple features
  - Tuple constructor
  - Use of tuple

# Review exercises

- Dictionary

# Book library -1

- Write a script to construct an empty dictionary named ‘booklib’ and then apply the **assigned pairs** technique to enter the following data into the dictionary
  - Don Quixote, Miguel de Cervantes
  - Moby Dick, Herman Melville
  - Hamlet, William Shakespeare
  - War and Peace, Leo Tolstoy
- Your script should display all the key-value pairs in the dictionary, one per line by iterating over the dictionary’s keys

# Book library -2

- Explain what the following code does. Then run the code and explain the output.
- Finally see the documentation for copy() function and explain how to use it to get the desired outcome.

```
import pprint
titles = ['Don Quixote','Moby Dick','Hamlet','War and Peace']
authors = ['Miguel de Cervantes','Herman Melville','William Shakespeare','Leo Tolstoy']

book = {'Title':'', 'Author':''}
booklib = [book for i in range(4)]

for i in range(len(booklib)):
    booklib[i]['Title'] = titles[i]
    booklib[i]['Author'] = authors[i]

pprint.pprint(booklib)
```

# Functions

# Python basics: **Functions**

- Functions defined
  - The 'def‘ command. Examples
  - How does Python execute a function?
- Arguments
  - Arguments pass by 'object reference'
  - Flexible argument passing
    - (a) Default parameter values
    - (b) Keyword Arguments
  - Unpacking return objects in .format()
- Variable Scope
  - What is "variable scope"
  - Rules

# Review exercises

- Functions

# Statistics

- Write a function that takes as input argument a list of scalars (integers or floats) and returns the mean value and the sample standard deviation

- Hints:
- See the statistics built-in library for statistics functions to use in your function algorithm

# Primes

- Write a function **isprime()** that takes an integer as input argument and returns True or False depending on whether the integer is prime (‘True’) or not (‘False’)

- *Hint*: All prime numbers, other than 2 & 3, are of the form 6n+/-1

- Addition: Use & explain the code:

```
if __name__ == '__main__':
    main()
```

# Upper – lower case

- Write a function that takes as input a string and an appropriate parameter. Depending on the parameter value the function returns the transformed string either in UPPER case (capital letters) or in lower case.

- *Hint*:
- Use the upper() and lower() methods to transform in upper or lower case respectively

# A strong password

- Write a function that takes a string as input (supposed to be a proposed password) and returns also a string including an explanation as to whether the password is strong or not.
- A password is strong if it satisfies the following criteria:
  - (1) Has at least 8 characters
  - (2) Includes capital AND lower case characters
  - (3) Contains at least one 0-9 digit (see isdigit function)
  - (4) Contains at least one of the special characters: ! @ # $ % &
- Write your function as an enclosing function with 4 enclosed functions (one for each of the above criteria). Each of these four functions returns 'True' or 'False' depending on whether the proposed password satisfies the relevant criterion.