**Analyse 2: Foundation of Modeling:**

**A small study Guide**

**By**

**Nathan Samijo**

**0961962**

Contents

[Lesson 1: Lists 2](#_Toc29846264)

[Questions Lesson 1: 7](#_Toc29846265)

[Answers Lesson 1: 8](#_Toc29846266)

[Lesson 2: Functions 8](#_Toc29846267)

[Questions Lesson 2: 14](#_Toc29846268)

[Answers Lesson 2: 15](#_Toc29846269)

[Lesson 3: Python functions Part 2: 16](#_Toc29846270)

[Questions Lesson 4: 20](#_Toc29846271)

[Answers Lesson 4: 21](#_Toc29846272)

[Lesson 5: Higher Order Functions 21](#_Toc29846273)

[Questions lesson 5: 23](#_Toc29846274)

[Answer lesson 5: 23](#_Toc29846275)

[Lesson 6: Higher Order Functions 2 24](#_Toc29846276)

[Questions Lesson 6: 25](#_Toc29846277)

[Answers Lesson 6: 25](#_Toc29846278)

[Lesson 7: Combinatorics 26](#_Toc29846279)

# Lesson 1: Lists

Under Lists we understand:

1. A data structure in Python
2. Mutable (changeable) ordered sequence of elements:
   1. This means that all elements that are in a list or will be put in a list will have a position
   2. Every element can be changed or removed
3. Values that are within are called items or members

Defining a list: (typically) listing values separated by commas , between square brackets **[]**

**Example:** list\_name = [ 1, 2, 3]

Usage of Lists: when working with many related values.

**Example:** names, birthdays, months, etc

Advantages of using a list:

1. Keep data together
2. Condense code => no need to have 1000 variables, you get the idea
3. Perform same methods and operations on multiple values at once

Operator Shortcuts:

Used to improve code clarity. All of the shortcuts (in Python):

A screenshot of a social media post

Description automatically generated

List Comprehension:

(In Python) List Comprehension provides a concise (in Dutch: kort en bondig) way to create lists.

It consist of:

1. Expression
2. For clause
3. (Optional) Additional for clause
4. (Optional) If clause(s)

Flowchart 1:

A screenshot of a cell phone

Description automatically generated

Flowchart 2:



Nested Lists:

Nested list are list that are an element of another list. You can nest as many list inside of other lists as needed.

Picture 1: nested list: list within a list within a list

A close up of a map

Description automatically generated

Picture 2: nested list: accessing data: keep good track of the nested list:

A screenshot of a cell phone

Description automatically generated

Example of nested lists: Matrix, etc

**Assignment vs Copy:**

In Python, every list is an object. An object can be bound to an object with the assignment operator (**=**).

**NOTE:** *If two or more variables are bound to the same object, any change using one will impact all the others.*

Shallow Copy:

1. Creates new compound object. Then inserts references into it, to the objects in the original => this object is not the same as the original
   1. The new object however contains only references to the elements in the original. This means that if you change a mutable item that was copied, this change will also happen to the original
2. Changes to the list do not affect the original:
   1. Adding items
   2. Removing items

A screenshot of a cell phone

Description automatically generated

Deep Copy:

1. Creates a new compound object. And then into it copies of the object in the original:
   1. Changing a value in the copied list will ***NOT*** affect the original in any way,
2. To use deep copy, import the **copy** moduleA screenshot of a cell phone

   Description automatically generated

## Questions Lesson 1:

1. What is a list in Python?
2. What is a list member?
3. How can you create list in Python?
4. What are the main benefits of using lists in your program?
5. How do you access list members?
6. How do you count the number of elements in a list?
7. Which operator is used to test list membership?
8. How do you add elements to a list?
9. How do you remove elements from a list?
10. What is list slicing in Python?
11. What do operators **+** and **\*** do when used with lists?
12. What is a list comprehension in Python?
13. What are the required and what are the optional parts of list comprehension syntax?
14. What is a matrix? How can you represent a matrix in Python?
15. What is the main difference between list assignment and list copy?

## Answers Lesson 1:

1. List is a mutable ordered sequence of elements.
2. Each value contained inside a list is called an item or a member.
3. List can be created as an empty list using **list()** command, or by listing all values, separated by commas between square brackets.
4. The main benefits of using lists are: keeping data together, ability to perform same methods and operations on multiple values at once, and condensed code.
5. Elements in list are accessed using indices, either positive (starting from the beginning) or negative (starting from the end).
6. Elements in list are counted using **len()** function.
7. Operator **in** is used to test membership of an element in a list.
8. The methods to add element to a list are **.append()** and **.insert()**. Append will add the element to the end, while insert will insert the element at the given position.
9. There are three options to remove an element from a list. Command **del()**can be used, or methods **.remove()** or **.pop()**.
10. Slicing is a quick way of retrieving a subset of items from a list. Inside of curly brackets, you define start and end index separated by colon, and optionally step value.
11. Operator **+** is used for list concatenation, while operator **\*** is used for list repetition.
12. A List comprehension is a syntactic construct in Python that allows to create a list in a single line of code.
13. The required part of list comprehension is the expression, followed by for clause. Optionally, there can be additional for clause(s) and one or more if clause(s) at the end.
14. Matrix is a two-dimensional data structure where elements are arranged in rows and columns. In standard Python it can be represented as nested lists.
15. The main difference between assignment and copying lists in Python is that by assigning multiple variables to the same list, makes the change in one reflect on all the others. Copying, in general sense, creates a new object, so that the changes in the copy do not influence the original.

# Lesson 2: Functions

**NOTE:** *A mathematical relation is an association of two objects, based on some property possessed by them.*

Relations:

A relation is a relationship between sets of values.

In mathematics, relations between two sets can be expressed in:

* + Roster form,
  + Set builder form,
  + Arrow diagram.

**Roster Form:**

* 1. This relation is represented as a set of ordered pairs
  2. The pairs are made based on the relation

Example:

* Let A = {1, 5} and B = {2, 5, 10}
* Let R be the relation between sets A and B such that “**a is less than b**” :

**R = { (1,2), (1,5), (1,10), (5,10) }**

* Note: A x B = {(1,2),(1,5),(1,10), (5,2),(5,5),(5,10)}, hence R ⊆ A x B.

**Set Builder Form:**

1. This is basically another way to write a relation

**Using the same example:**

* **Let A = {1, 5} and B = {2, 5, 10}**
* **Then the relation “a is less than b” can be written as:**

**R = {(a, b) : a ∈ A, b ∈ B, a is less than b}, or**

**R = {(a, b) : a ∈ A, b ∈ B, a < b}**

**Functions:**

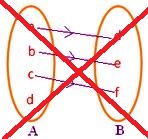
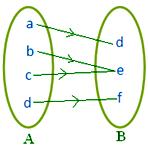
A function can be seen as a special kind of relation and consists of three parts:

1. Input
2. Relationship
3. Output

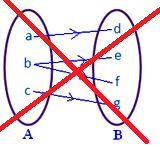
A relation is a function when:

1. Every element from the element from the input set, must have a image in the output set
2. No element from the input set has more than 1 image
3. Different elements from the input set can have the same image but not the same

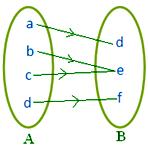
**Rule 1: Relation is a function:**

****

**Rule 2: Relation is a function:**

****

**Rule 3: Relation is a function:**

****

**NOTE:** *all functions are relations but not all relations are functions*

**Picture: Not all relations are functions:**

****

Functions have:

1. Domain => what can go into the function
2. Codomain => what may possibly come out of a function
3. Range => what actually comes out of a function

*Functions are single valued. This means that for every input there is exactly one output.*

**NOTE:** *I have not included solving functions as I trust that you do this by yourself.*

**Toolkit Functions:**

**A close up of a clock

Description automatically generated**

**A screenshot of text

Description automatically generated**

**Python Functions:**

A function is a block of statements with a name that does a specific task. Functions are used to:

1. Avoid code repetition
2. Make abstractions => making a program easier to read, debug and maintain

Python Functions:

1. Consists of functions definitions => here the functionality of a functions is defined
2. Is used via function calls => here the function is executed

**Picture: Function Definition:**

**A screenshot of a cell phone

Description automatically generated**

**Scope:**

Namespace:

A namespace is a system to have a unique name for each and every object in Python.

**NOTE:** *A object might be a variable or a method*

**Picture: Type of namespaces:**

****

**NOTE:** *the scope of a variable determines it’s lifetime and where in the program It is accessible.*

Variable scope:

1. Global variable
2. Local variable

Global Variable:

1. Declared outside of functions
2. Available from the moment of definition and accessible to any part of the program
3. Lifetime => from moment of declaration till the termination of program

Local Variable:

1. Declared inside a function or passed to a function as an argument
2. Only accessible from inside the function
3. Lifetime => from declaration inside the function till the termination of that function

## Questions Lesson 2:

1. What is a relation?
2. How can mathematical relations be expressed?
3. What is a function?
4. When is a relation also a function?
5. What is function domain, codomain and range?
6. What is function in programming?
7. What does a Python function consist of?
8. What is a namespace?
9. What is variable scope?
10. What kind of variables do we distinguish in Python (according to their scope)?
11. What is a global variable?
12. What is a local variable?

## Answers Lesson 2:

1. A relation is relationship between sets of values. Mathematical relation is an association of two objects or two variables, based on some property possessed by them.
2. In mathematics, relations between two sets can be expressed in: a) roster form, b) set builder form, c) arrow diagram.
3. A function is a mathematical relationship in which the values of a single dependent variable are determined by the values of one or more independent variables. It consists of three parts: input, relationship and output.
4. A relation is a function from A to B if every element of A has an image in B and no element of A has more than one image.
5. The domain is the complete set of possible values of the independent variable. The codomain or target set of a function is a set into which all of the output of the function is constrained to fall. The range of a function is the complete set of all possible resulting values of the dependent variable.
6. A function in programming is a block of statements that performs a specific task. Programming functions allow us to avoid code repetition and make programs easier to read, debug and maintain.
7. A Python function consists of function definition and function call. Function definition is where the functionality of a function is defined. A function is executed via function call.
8. A namespace is a system to have a unique name for each and every object in Python.
9. The scope of a variable determines its lifetime, and where in the program it is accessible.
10. In Python we distinguish between global and local variables.
11. Global variables are available from the moment of declaration, and accessible from any part of the program. They exist from the declaration until the termination of the program.
12. Local variables are declared inside of a function or passed to a function as an argument. They are accessible only inside of the declaring function. The lifetime of a local variable is from the declaration inside of a function, until the termination of that function.

# Lesson 3: Python functions Part 2:

**Parameters or Arguments:**

1. These are used to pass information to Python functions.
2. Listed between the parentheses, (), following the function name
3. The values are passed the moment of a function call
4. Python passes all values by reference

**Pure Functions:**

A pure function is a computational analog of a mathematical function and has the following properties:

1. Same return values for the same arguments
2. Evaluation of pure functions has no side effects => this means that a pure function’s return is based only on it’s input and has no other dependencies or effect on the overall program

**Arguments:**

Besides standard arguments we have:

1. Default arguments:
   1. Can be skipped during a function call, and then the default value is used
2. Keyword arguments:
   1. Can be passed in any order
3. Variable-length arguments:
   1. When the developer is unsure about the amount of arguments that will be used

**Python Modules:**

Python modules are Python libraries that contain pre-written code that can be easily imported our programs. Using this significantly reduces development and debugging time. By default Python libraries are installed in the Lib folder

Importing the Libraries:

1. **Importing a Module:** With the **import** statement, the developer imports entire modules/libraries. => **import** module\_name
2. **Import Specific Functions:** only import a function that you need. => **from** module\_name **import** function\_name
3. **Import modules as object:** the module will be imported as an object. => **import** module\_name **as** object\_name

**Questions Lesson 3:**

1. What is a parametric function (in programming)?
2. What is a pure function?
3. What are default arguments in Python?
4. What are keyword arguments in Python?
5. What are variable-length arguments in Python?
6. How are modules imported in Python?
7. Where are Python modules stored by default?

**Answers Lesson 3:**

1. A parametric function is a function that takes one or more parameters / arguments as inputs.
2. A pure function is a computational analog of a mathematical function, whose return value is based only on its inputs and has no other dependencies or effects on the overall program.
3. Default arguments are arguments that have been assigned a default value, and as such can be omitted during function calls. In that case, the default value is used.
4. Keyword arguments are arguments that are passed according to their names, in any order.
5. Variable-length arguments are used when the number of the needed arguments is not known in advance. Non-keyworded variable-length arguments have one asterisk (\*) before the argument name and keyworded variable-length arguments have two asterisks (\*\*) before the argument name.
6. Python libraries can be imported with import command. Entire modules can be imported, or only specific functions. Also, modules can be imported as objects.
7. By default, all modules are stored in Python Lib directory.

**Lesson 4: Plotting Functions**

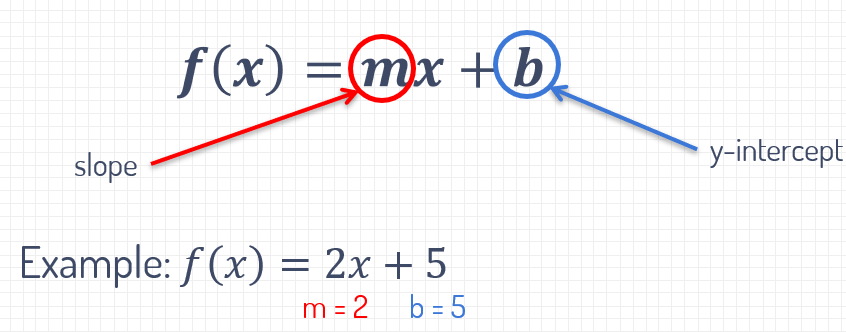
Linear Functions:

1. Have one input variable (one parameter)
2. The input variable always has an exponent 1.
3. The graphs from linear functions are straight lines, hence the name linear

Linear functions can be expressed in:

1. Slope-intercept form: ***y* = m*x* + *b***
2. Point-slope form: **(y – y1) = m(x – x1)**
3. General Form: **A*x* + B*y* + C = 0**

**Slope-intercept form:** is the most common way to express a linear function when plotting



**Point-slope form:** useful to find all points on a line when the makes meets the following requirements:

* The slope is known
* One point on that line is known

Point-form

**General Form:**

For any point (x, y), if the general form is true (equals 0), then this point lies on a certain line

A close up of a sign

Description automatically generated

**Slope:**

This represent the steepness of a line and is usually represented by the letter *m.* Defined as ratio between the rise and the run.

Rise: the vertical change between two points

Run: the horizontal change between two points

**Picture: Calculate the slope:**

A picture containing text

Description automatically generated

**Picture: Run and rise explained**



**Types of slopes:**

There are different types of slopes and these are:

1. A line with a positive slope: rises from left to right
2. A line with a negative slope: falls from left to right
3. A line with a zero scope: the line remains horizontal
4. Parallel lines have the same scope

**Y-intercept:**

This is the point where the line crosses the Y-axis and is usually represented by the letter **b** -> not always.

Can be calculated from**:**

1. **Two points:** first find slope, then substitute one point in the equation
2. **Function equation:** substitute x with 0

**A close up of text on a white background

Description automatically generated**

**Matplotlib:**

This is a Python library used to generate 2d plots, histograms, bar charts, scatterplots and many more. **Pyplot** is a part of **Matplotrib** and us used to plot our functions

**NOTE: *plot()*** *is used to plot points and* ***show()*** *must be used at the end to display the plot*

## Questions Lesson 4:

1. What are linear functions?
2. What is a slope-intercept form?
3. What does a slope represent?
4. What is the behavior of a line with a positive slope? What about negative?
5. When are two lines parallel?
6. What is y-intercept?
7. What is the mathematical expression of a point-slope formula?
8. What is the mathematical expression of the general form for a linear function?
9. How can you import pyplot as an object in Python?
10. What is pyplot function plot() used for?
11. What is pyplot function show() used for?
12. What is specific about show() function, and why is it advised to call it at the end of a program?

## Answers Lesson 4:

1. Linear functions are functions with one variable, and that variable always has an exponent 1. They yield graphs that are straight lines.
2. A slope-intercept form is a mathematical representation of a line (linear function) with an equation . In this equation, m denotes a slope, and b denotes a y-intercept.
3. A Slope represents the steepness of a line. It is defined as ration between rise (the vertical change) and run (the horizontal change) between two points.
4. A line with a positive slope rises from left to right, while a line with a negative slope falls from left to right.
5. Two lines are parallel if they have the same slope.
6. Y-intercept is the point where the line crosses the y-axis.
7. The point-slope formula is expressed by the equation:
8. The general from of a linear function is expressed by the equation:
9. To import pyplot as an object use: import matplotlib.pylpot as plt.
10. Plot() is a versatile pyplot function used to plot points. The most common use is to pass it two sequences of X and Y values. Both sequences must be of the same length.
11. Show() is a pyplot function used to render (display) plotted points on the screen.
12. Show() is a blocking function, meaning that the execution of the program will stop until the plot window is closed. Hence, it is advised to be called only at the end of a program.

# Lesson 5: Higher Order Functions

***God is that you?***

In mathematics we distinguish two kinds of functions:

1. First order functions:
   1. These are functions that can never take functions as an argument and it can’t return a function as it’s result
2. Higher order functions:
   1. Are functions that can take one or more functions as an argument
   2. Or returns a function as its result

**Lambda Calculus (λ):**

1. Lambda (λ) calculus is a abstract mathematical theory of computing.
2. The λ notation is based on function abstraction and application using variable binding and substitution.
   1. In short: this allows you to write a function (code) without cluttering it with needless things (comments, etc)

A close up of text on a white background

Description automatically generated

**Function Composition:**

This is nothing more than giving the result of one function to another. A picture containing text

Description automatically generated

**Lambda Functions in Pyton:**

This is a specific form of a Python function and behaves exactly like one, the only exceptions are:

1. Lambda functions have no name
2. Can have only one expression

A close up of a screen

Description automatically generated

**Argument(s) : what you pass to the function**

**Expression: what the function has to do with the passed arguments**

**NOTE:** *it is possible to have a lambda function as a return value, but this is not a value but a function. Makes sense right?*

## Questions lesson 5:

1. What is the difference between first order and higher order functions?
2. What is lambda calculus?
3. What is function composition?
4. What is lambda function in Python? What is the difference between a lambda function and a regular function?

## Answer lesson 5:

1. First order functions can never take function(s) as an argument and they can’t return function as its result. In contrast to that, for a function to be a higher order function, it must take one or more functions as an argument, and/or return a function as its result.
2. Lambda calculus is a formal system in mathematical logic for expressing computation based on function abstraction and application using variable binding and substitution.
3. In mathematics, function composition is an operation that takes two functions f and g in such a way that the return value of one function (f) is passed as an argument to another function (g).
4. Lambda function is a specific form of Python function. It behaves like any other Python function, but it has no name, and can have only one expression.

# Lesson 6: Higher Order Functions 2

Continuation of lesson 5.

**Functions as Input Argument:**

This is handy when a function only works when it is *two or more independent components* , some of these arguments may be entered as argument functions.

**Example:** the following functions from Python take functions as arguments: **sort, map**. There are more that take functions as arguments, but these are the most used.

**Function as Return Arguments:**

Functions as return value are often used to give it additional properties.

**Example:** a function name without parentheses is contains the reference to the function BUT does **NOT** execute because the parentheses are missing. This makes it return a function without executing it. Ez pz right?

**Decorators:**

Decorator is a ***design pattern*** that allows a user to add ***new functionality*** to an already ***existing*** object (function or class) without modifying it’s structure.

**How a decorator works:**

Now listen, I mean read carefully and good. A decorator, also referred to as a wrapper, is a function that calls other functions from within based on the arguments that is passed to the decorator. Easy right?

**Syntactic sugar:**

Another way to apply a decorator is to write: **@decoratorxname** just above the function definition. This makes your code super sexy and readable, also referred as **syntactic sugar.**

**More on decorators on :** <https://hackernoon.com/decorators-in-python-8fd0dce93c08>

(beware there are a lot of fucking selfies jeez)

**NOTE:** *adding* ***@decorator*** *to your code makes it easier to read and there for other developers like you and call this joy: “****syntactic sugar****”*.

## Questions Lesson 6:

1. What is map function?
2. What is filter function?
3. What is a decorator?
4. What is syntactic sugar?
5. What is a decorator syntax using syntactic sugar?

## Answers Lesson 6:

1. Map is a higher-order function that applies a given function to each item of an iterable (list, tuple etc.) and returns a list of the results.
2. Filter is a higher-order function that processes a data structure (usually a list) in some order to produce a new data structure containing exactly those elements of the original data structure for which a given predicate returns the boolean value true.
3. A decorator is a design pattern in Python that allows a user to add new functionality to an existing object (either a function or a class) without modifying its structure.
4. Syntactic sugar is any aspect of the syntax of a programming language that makes programs easier to read, write, or understand.
5. To use syntactic sugar to decorate a function, you must type @ sign, followed by the decorator name, just above the function definition.

# Lesson 7: Combinatorics

Motivation:

There is a motivation section in the presentation and it’s so fucking depressing to be honest. So I’ll give you some motivation.

A screenshot of a cell phone

Description automatically generated

Small progress is still progress. Everything you do you do for future you. Past you is you in the past no matter how far back. And then we have present you. Next time you do something do it for future you and remember to thank past you for doing it if you did it. Okay too much cheese let’s move on.

**Combinatorics:**

This is an area of mathematics that primarily focused on counting. However also used for generating finite structures such as:

1. Finite sets
2. Finite lists
3. Functions and relations on finite sets
4. There’s too much too list but these are some examples^^

**Addition:**

If a set contains of **disjoint (unique)** subsets, count the items in each subset and add up the results.

**Multiplication rule ( rule of product ):**