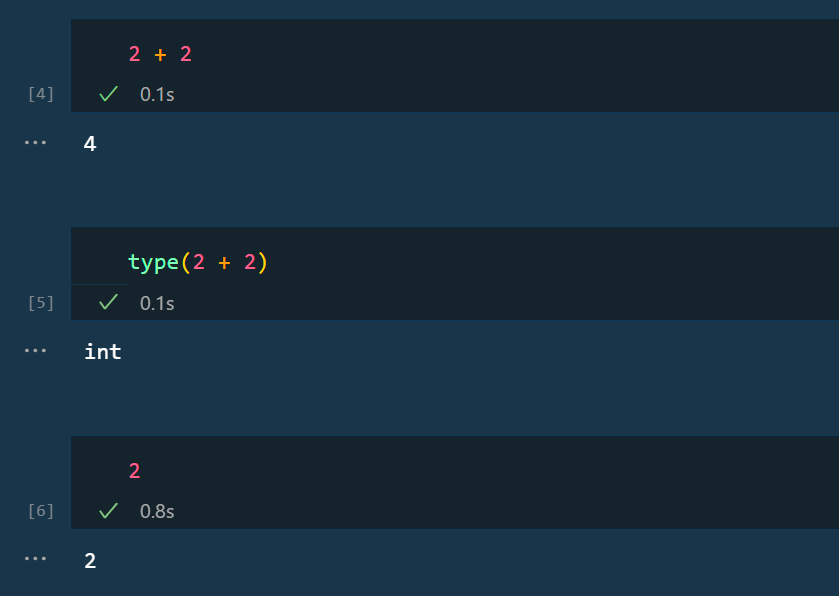
# Python Core Concepts

## Expressions

1. An **expression** is the most basic instruction in any programming language.
2. An expression consists of ***values*** and ***operators****.*
3. They will always evaluate to a single result.



Here, 2 + 2 is called as an expression.

2, 2 are values and + is the operator.

## Variable

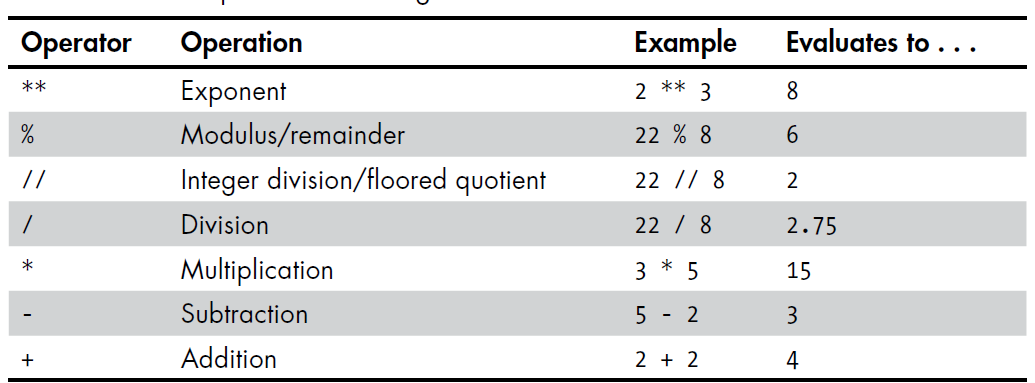
Variable

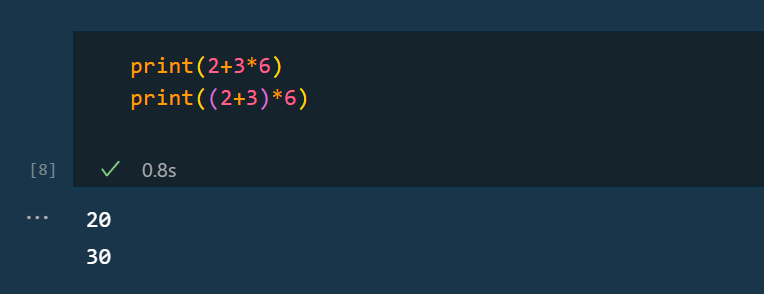
Variable overwriting

Global variables

## Operators

1. Arithmetic operators
2. Logical operators
3. Boolean operators

Math operators

* The order is from highest precedence to lowest precedence.
* Using parentheses, you can override the expression

### Order of Precedence

Python follows the same order of precedence as mathematics i.e.

* () Brackets
* **\*\*** Exponential
* **%** Modulus
* **//** Integer division
* **/** Division
* **\*** Multiplication
* **-** Subtraction
* **+** Addition
* All the expression values evaluate down to single value.
* Each Single value is assigned with a datatype.

### Walrus operator

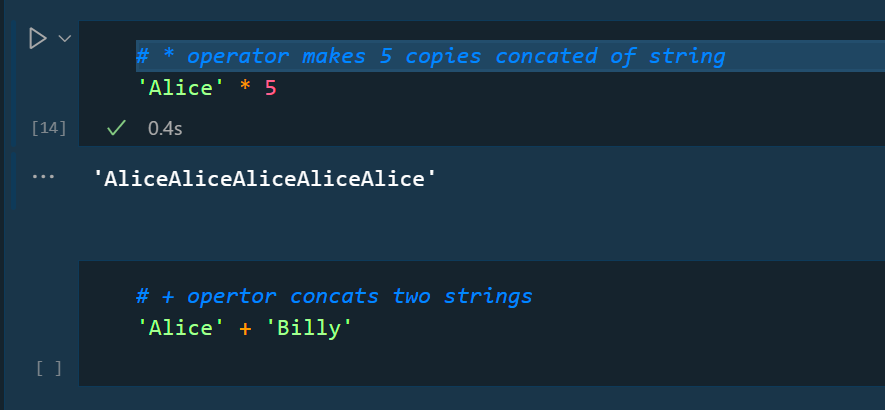
* This is basically used to reduce the time

### Bitwise Operators

## Datatypes in Python

1. Integer
2. Floating pointer
3. String type

Different datatypes with operators behave in different manner.

Ex-

### Strings in Python

#### Concatenating, Indexing and Slicing

#### String methods

Strings are immutable. You can’t change them once they are created.

string.lower()

string.strip()

* This function is used to remove any whitespaces, blanks, tabs etc. especially in case of input has to come from user.

string.lstrip()

* Used to remove spaces from the left side of strings.

string.rstrip()

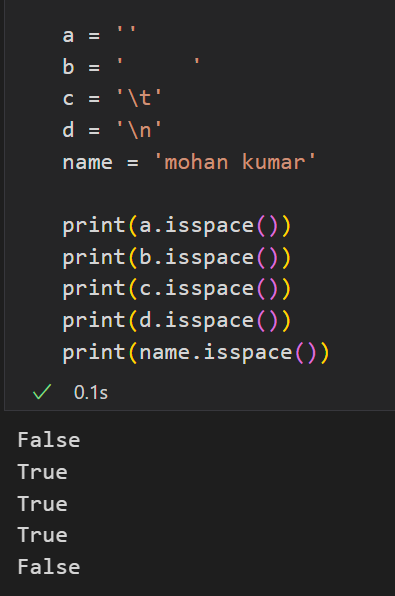
* Used to remove spaces from the right side of strings.

string.mstrip()

* Used to remove spaces from middle of string.

string.replace()

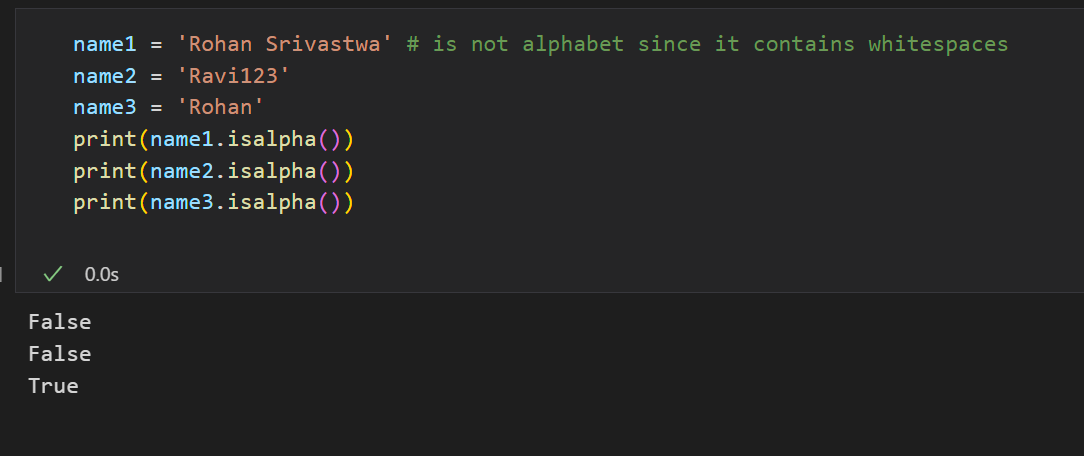
string.find()

string.isspace()

* The isspace() method returns **True** if there are only whitespace characters in the string.
* If not returns **False**. Ex- b, c, d
* isspace() methods doesn’t take any parameters.

string.isalpha()

* The isalpha() method returns **True** if the string has only alphabets
* If not then it returns **False.**
* Isalpha() methods doesn’t take any parameters.



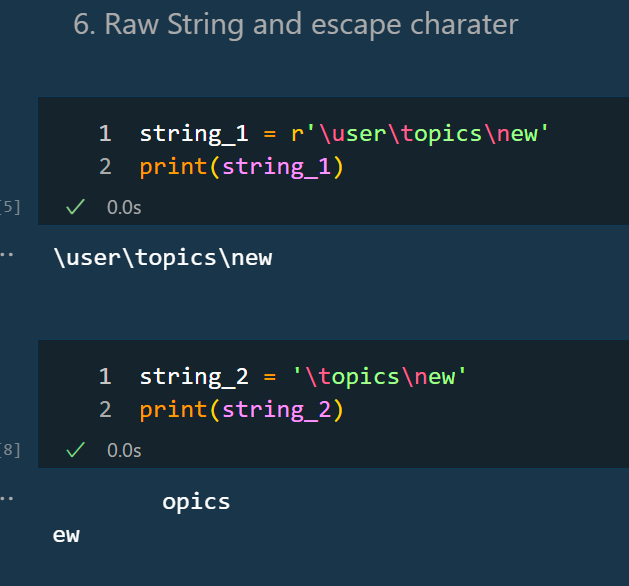
#### f strings in python

Replit

How to get substring - <https://realpython.com/python-string-contains-substring/?utm_source=notification_summary&utm_medium=email&utm_campaign=2022-08-16>

#### Raw String – r’

* In Python, the **‘r’** prefix before a string literal denotes a “raw string”.
* By default, python interpreter treats ‘\t’, ‘\n’ as tab space and new line respectively. So, we provide these in any string variable this will be treated differently in python. You can see in tsring\_2 there is tab space in output and a new line also



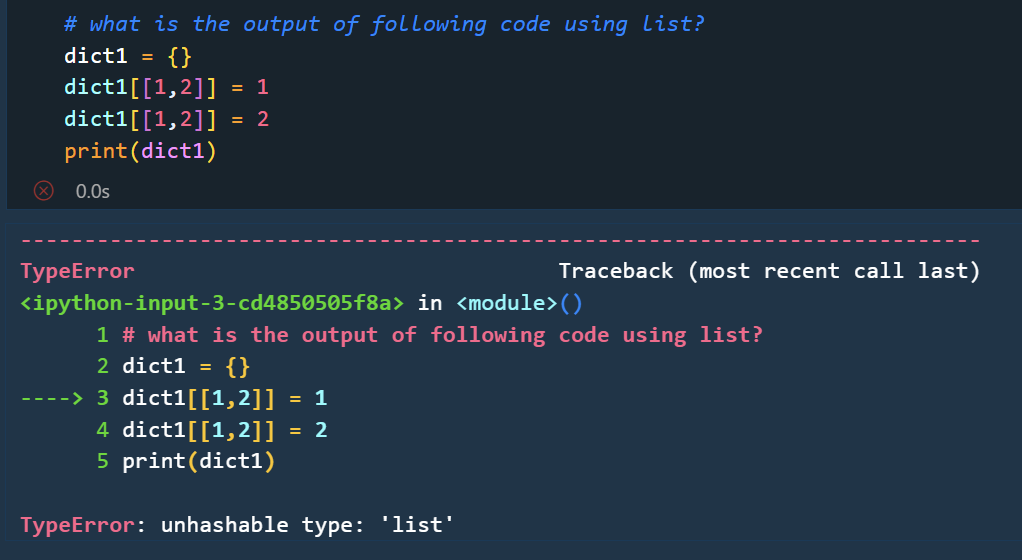
#### Special characters

Escape character

## Immutable and mutable in python

**Immutable**

* If the state of an object cannot be changed or data contained cannot be changed, then it is called as immutable object.
* Immutable objects are common in functional programming.
* Immutable objects have hashes created for them and they don’t change throughout the lifecycle of program.
* Immutable objects – strings, tuples, frozensets, integers, floats

****

**Mutable**

* If the state of the object can be changed and the data contained can be changed, then it is called as mutable object.
* Mutable objects are used in object-oriented programming.

## Objects, values and types

Any Object in python has three things – id, type & value.

**Id**

* An object **‘id’** can never be changed once created for an object.
* ‘is’ operator is used to compare identity of two objects
* id(x) function returns an integer representing an object’s identity.
* id(x) refers to the memory location where x is stored.



**Type**

* An Object **‘type’** determines the type of operator that the object will support and possible values of object of that type.
* type() function is used to determine the type of an object.
* Similar to object id, its type cannot also be changed.



**Value**

* Object **‘value’** can be changed for some objects.

Immutable – Objects whose value cannot change are called immutable. Ex- string, int, tuples.

Mutable – Objects whose value can change are called mutable. Ex- list, dictionary.

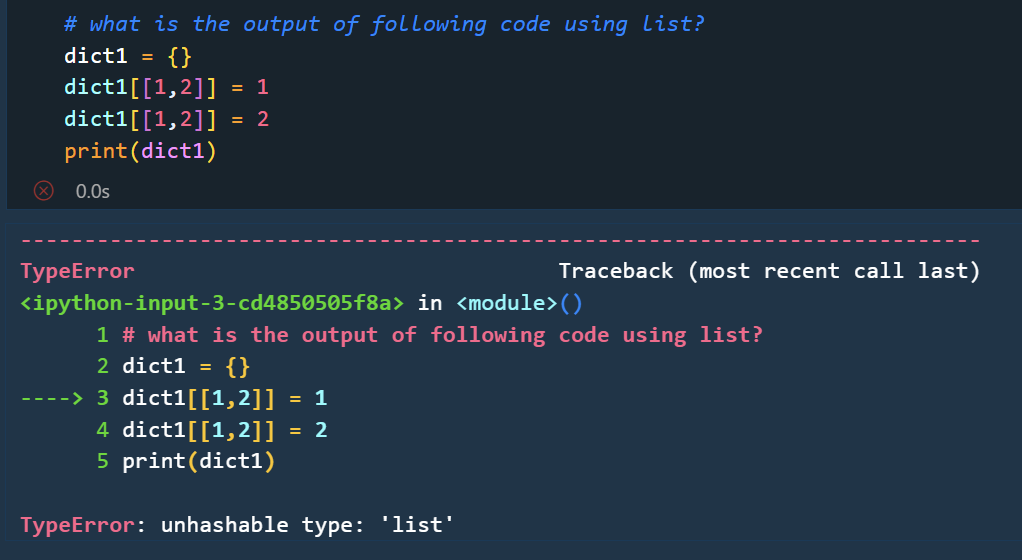
## Advanced datatypes

## Hash tables

<https://realpython.com/python-hash-table/>

Also knows as **hash maps** are concepts way back to 50 years, it is **classic data structure** and fundamental to programming. Python’s own hash table is dict.

### Dictionary

* Dictionaries were inherently unordered.
* A python dictionary is an implementation of the **hash table 🡪 which is an unordered data structure.**
* A dictionary is a **set** of [ key: value] pair and **sets** are unordered.
* Dictionaries also does not have much reordering functionality unlike list where you can insert elements at any position.
* ****Dictionary keys are hashable objects which means they must be immutable objects like string, integers or tuples.

#### dict\_items class

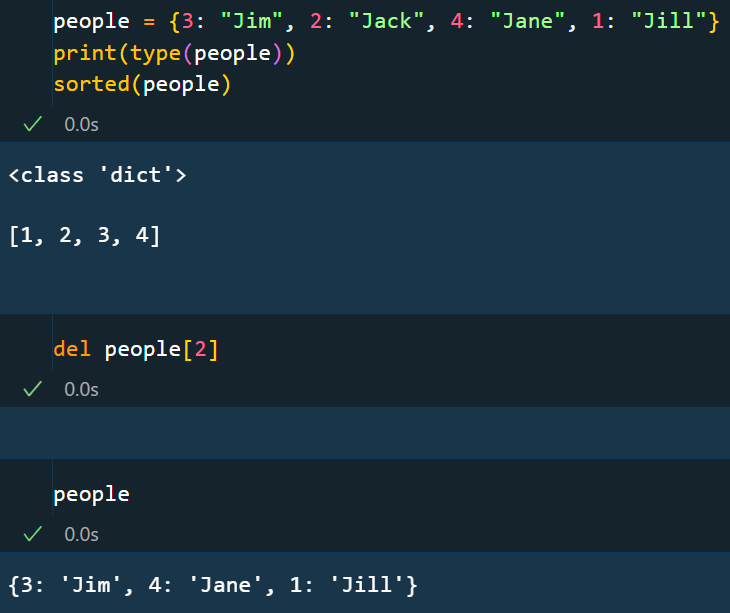
* It is an object type that represents a view of the items (key-value pairs) in a dictionary.

#### defaultdict

* defaultdict in Python is a subclass of the build-in dictionary type.
* It works like a regular dictionary, but with 1 key difference: when we access a key that does not exist in the dictionary, a defaultdict will automatically create that key and assign it a default value specified by a factory function.
* By using defaultdict, we don’t need to explicitly check whether a key exits in the dictionary before accessing it. This can simplify our code and reduce the chance of errors.

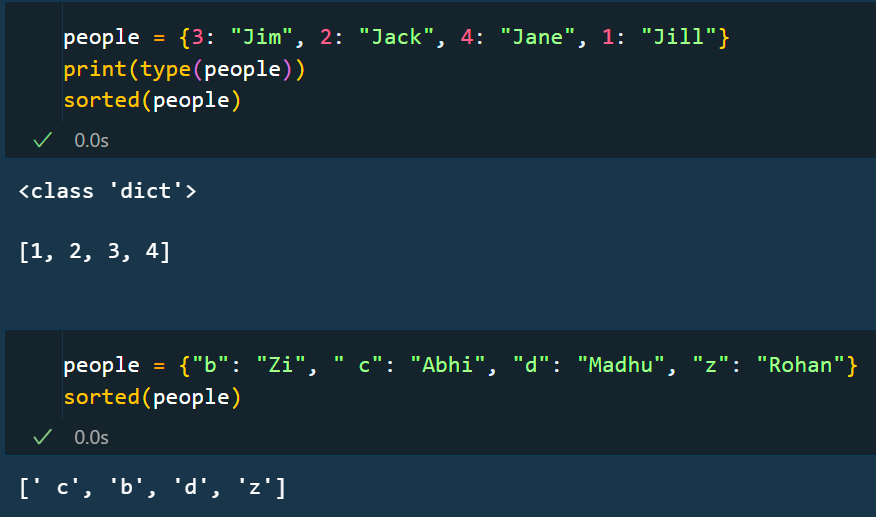
#### Dictionary operations

* + - 1. Iterating over a dictionary
      2. Deleting item from dictionary



* + - 1. Sorting a dictionary
* sorted() for dictionary
  + <https://www.freecodecamp.org/news/sort-dictionary-by-value-in-python/>
  + sorted() function is not suitable for sorting dictionary since it gives sorted “keys” of dictionary .

Ex- for int keys – ascending order for string keys – alphabetical order



1. Using OrderedDict class
2. callback function
3. list.sort() vs sorted()
4. sorted(reverse=True)
5. reversered()
6. sorted(dictionary)

#### Dictionary methods

* + - 1. dict.items()
      2. dict.keys()
      3. dict.values()
      4. dict.fromkeys()
      5. dict.pop()
      6. dict.popitem()
      7. dict.upate()
      8. dict.copy()
      9. dict.clear()
      10. dict.get()
      11. dict.setdefault()

## Collections in Python

### Sets

## Sequence in Python

### Strings

### List

Define a list

Index (negative index)[inclusive] and slice of a list(inclusive:exclusive)

Iterating through a list

List vs tuple

Methods of list

list.append()

list.extend()

list.sort()

list.reverse()

list.count()

list.clear()

list.pop()

List comprehension

Build-in functions for list

#### Zipped

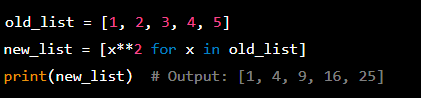
* Zip is a built-in function used to combines two or more datasets and zip them together.
* This returns a object containing pair of items derived from the datasets.
* The grouping happens in the order of indexes.

#### List Comprehension in Python

* There are two ways to generate a list comprehension – using for loop and list comprehension.
* List comprehension is more efficient than for loop for generating lists.
* In Python, there are two types of list comprehensions: Basic and Nested.
  + - 1. Basic list comprehension:

A basic list comprehension is a concise way of creating a new list by transforming or filtering an existing list. It has the following syntax:

* **expression** is a transformation or computation to be performed on each element of the iterable
* **item** is an element of the iterable
* **iterable** is the existing list or sequence
* **condition** is an optional condition that filters the elements of the iterable based on a boolean expression

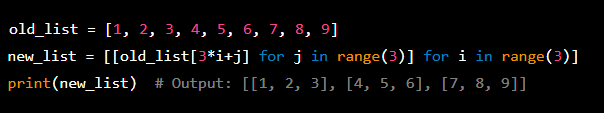
For example, the following code uses a basic list comprehension to create a new list of squared values from an existing list of integers:

1. Nested list comprehension:

A nested list comprehension is a way of creating a list of lists (or a multi-dimensional list) by applying a list comprehension to each element of an existing list. It has the following syntax:



* **expression** is a transformation or computation to be performed on each element of the inner iterable
* **item** is an element of the inner iterable
* **iterable2** is the inner iterable
* **item2** is an element of the outer iterable
* **iterable1** is the outer iterable
* **condition** is an optional condition that filters the elements of the outer iterable based on a boolean expression

For example, the following code uses a nested list comprehension to create a 3x3 matrix (a list of lists) from an existing list of integers:

In this example, the outer iterable is a range from 0 to 2 (inclusive), and the inner iterable is a range from 0 to 2 (inclusive). The expression is the indexing operation that extracts the appropriate element from the old\_list.

### Tuples

### range() objects

### Byte arrays

### Byte sequences

## Dates and times

* Epoch time
* Olson time zone
* Daylight saving time (DST)
* UTC
  + Stands for coordinated universal time.
  + Based on International atomic time (TAI)
  + Also called as Zulu time and is used in fields where precise timing are important like navigation, aviation.
  + It is not affected by Daylight saving time (DST).

## Decimals

## Enums or Enumeration

<https://realpython.com/python-enum/?__s=trcte8voapvgbb66hrgs>

Enumerations are useful in following scenarios

* + - 1. Day of a week, months and years
      2. Earth Cardinal direction – NSEW
      3. HTTPS status code
      4. Colours of traffic lights

## Containers

* Containers are data structures holding elements and that supports membership tests. These data structures live in memory and hold all values in memory too.
* All the container types are iterables i.e., each element in the container can be visited once.
* <https://cvw.cac.cornell.edu/python-performance/faster-python/python-containers>
* Python provides several build-in container types :

Lists

Tuples

Sets

Dictionaries

Strings

Collections

Numpy arrays

User-defined classes

# Control Flow

## If statement

## If-elif-else statement

## Loops in Python

### While loop

### For loops

## Loop control statements

* There are situations in looping when you want to exit the loop, skip an iteration or ignore the condition. These can be done by Loop control statement.
* break, continue, pass statements are loop control statements.

### break statement

* break statement is used to terminate the loop or statement in which it is present. After this the control will move to statement next to break statement if present in the loop.

### continue statement

* continue statement is used to skip the current iterations and move to next iteration in a loop (for and while loop). It does not break the loop but just skips the existing loop iteration.

### pass statement

* pass statement is a null operation, it does nothing when executed.
* It simply serves as a placeholder for something in that block of code which be modified at later stages.

<https://nvie.com/posts/iterators-vs-generators/>

## Iterators and Iterables

## Iter() in Python

Iter() function is used to return iterator for a given object.

## Scopes in Python

1. Scopes refers to the region in program where a particular variable or function

Types of scopes

Build-in

Global

1. A variable defined outside a function is a global variable.
2. A global variable can be accessed outside or inside a function.

Local

1. A variable defined inside a function is considered local scope to that function.
2. These variable are accessible only with the function they are defined and they are not visible or accessible from the outside function.

Nonlocal

1. This keyword denotes the variable assigned this variable is not local to the scoped it is defined to but refers to nearest enclosing scope

# Functional programming

Python is a multiparadigm language since it supports both functional programming and object-oriented programming.

Functional programming

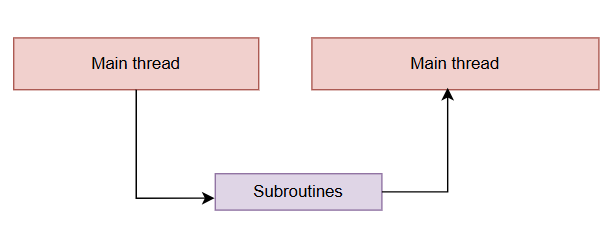
1. Programs are written using purely functions.

Object-oriented programming

* Functional programming a paradigm in which the primary method of computation is evaluation of functions.
* In functional programming you deal with pure functions which do not have any [side effects](#_Functions).
* <https://realpython.com/python-functional-programming/>

## Subroutines in Python

* Subroutines are block of code in Python designed to perform specific tasks.
* When a subroutine is called from main thread, the call is transferred to subroutine which completes its task and the call is given back to main thread.
* Subroutines cannot be paused and resumed. They run until they complete.



* They are mainly of two kinds in python :
  + - 1. Functions
* Functions are independent block(not related to any object or classes) of code defined using “def” keyword.
* They can accept arguments when called and return value. Functions enhance code modularity, readability and reusability.
  + - 1. Methods
* Methods are functions associated with object and classes. They operate with data or attributes within a class. They are assessed using dot notation on instances.
* In Python, a method is a function that can be used on an object because of object type.

Example – if you create a python list, the ‘append’ method can be used with that list. All lists have ‘append’ method because they are lists.

Similarly, if you create a string, the ‘upper’ method can be used on that string because it’s a string.

To be clear, my\_list does not have ‘upper’ method and my\_string does not have ‘append’ method. That is because methods only exists for a particular object if they have been defined for that type of object.

Calling a method : **object\_name.method\_name()** and arguments with in the parenthesis.

## Coroutines in Python

* Coroutines are special types of functions in python used for asynchronous programming. They allow cooperative multitasking and non-blocking I/O operations. Coroutines are defined using the **async** and **await** keywords and are managed by event loop.
* Coroutines can be paused and resumed because they maintain their state.
* Applications of Coroutines :

Databases queries

I/O operations

HTTP request

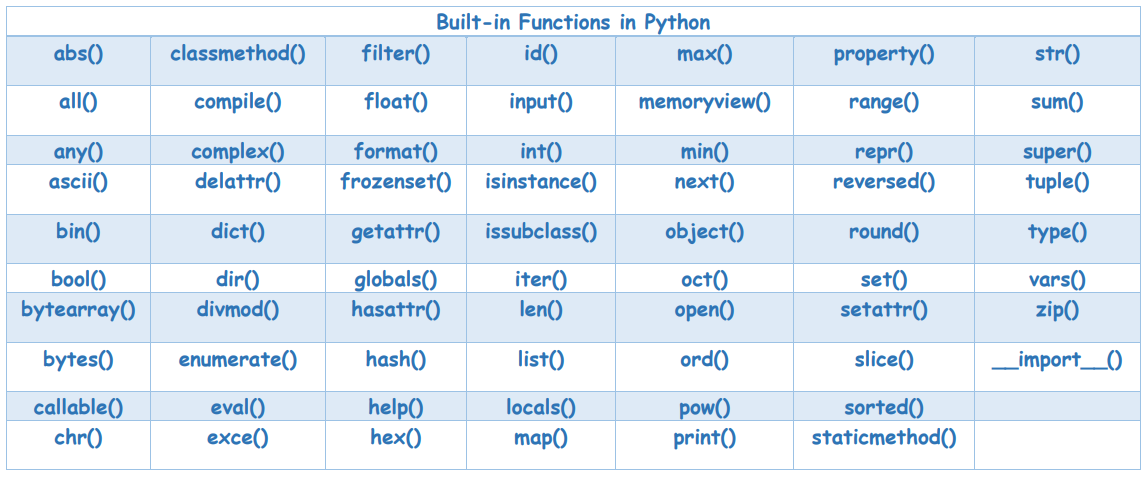
## Concurrency in Python

### Synchronous and Asynchronous programs

* In a synchronous programming, tasks are performed one at a time in the order they are called. Each task must finish before the next task starts.
* In an asynchronous programming, different tasks can start, process and finish in any order. Ex- http requests between a server and client. This non-linear approach saves the most valuable thing – **time.**

### AsyncIO in python

## Built-In Functions

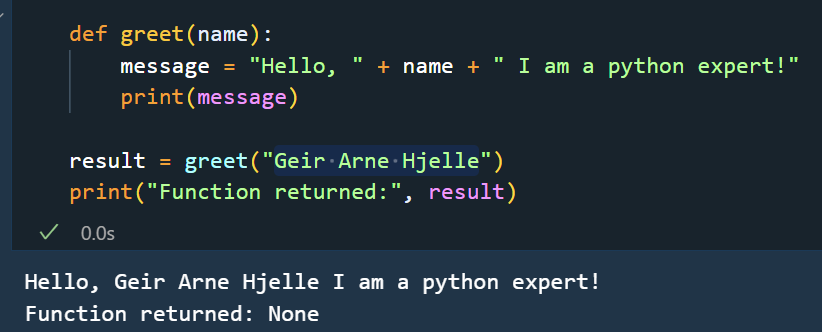


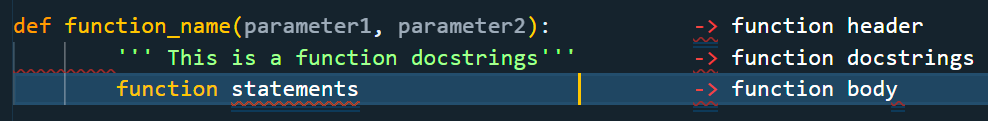
* There are around 68 built-in functions in Python as of 2024.

## Functions

* Functions is a sequence of instructions packed as a unit to perform certain task bundled as a single unit.
* Functions returns can have side effects rather than just converting an input (arguments) into output (values).

Ex- print() function outputs the message to console, even though there is no return statement but “None“ is still returned.



-> *function definition*

*-> function call*

function\_name(argument1, argument2,…)

->  *Docstrings : These are enclosed in* “’ ‘” *- triple quotes and used to given details about the function. They have similar role as comments.*

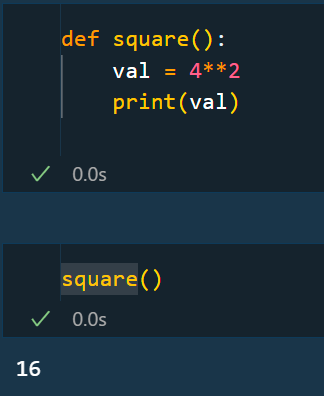
Note :

Function parameter is a variable which is defined inside a function definition header “()” parenthesis.

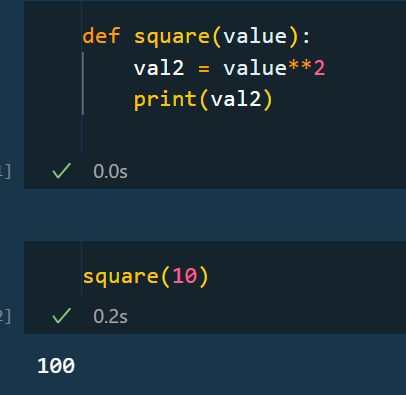
Function argument is a value which is always defined inside a function call “()” parenthesis.

### Types of functions

1. Functions without parameters
2. Functions with 1 parameter
3. Functions with a return value
4. Functions with multiple arguments and return values
5. Functions without parameters:



1. Functions with 1 parameter :



1. Functions with a return value

### \*args and \*\*kwargs in python

* args allows any number of positional arguments.
* The operators \* and \*\* helps you pass variable numbers of parameters to a function.

### High Order Function

* A function that uses another function as an input argument or returns a function (HOF) is known as higher-order function.
* Examples – map and filter.

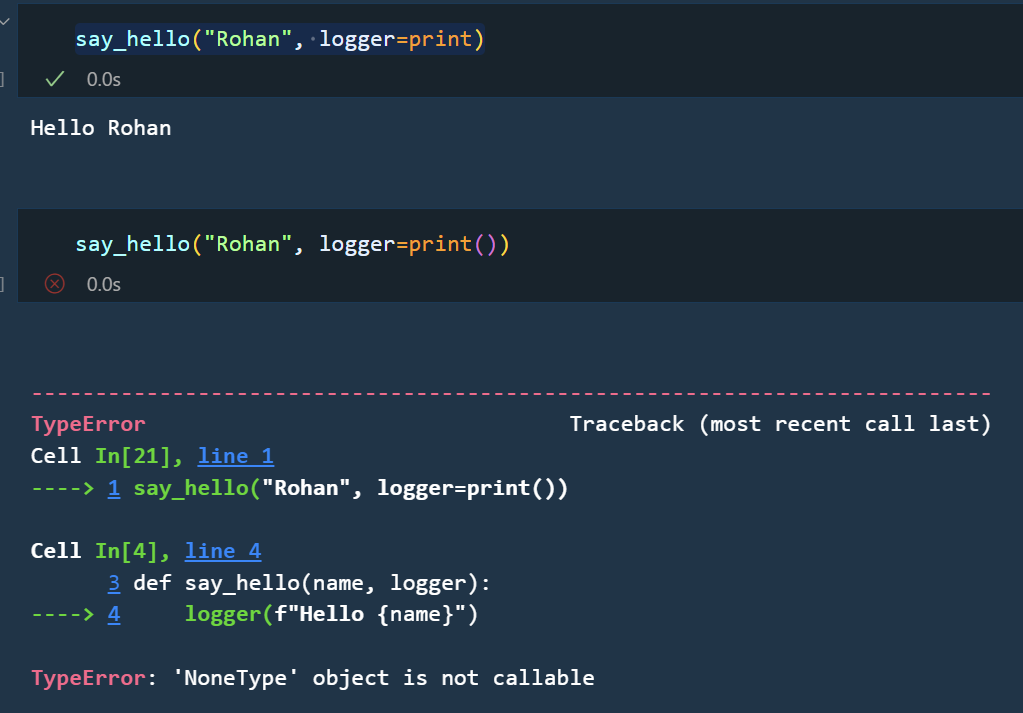
Any function is called HOF if

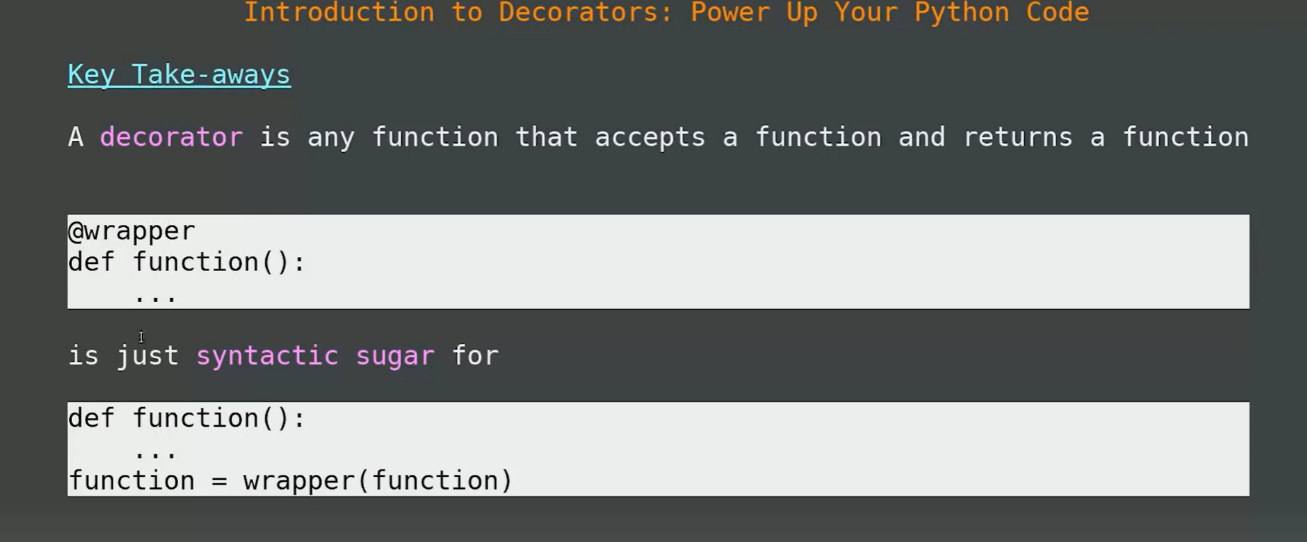
* + - 1. takes 1 or more function as argument
      2. returns a function as result

Example

## Anonymous functions - Lambda functions

## Decorator

* In python, functions are first-class objects. This means that functions can be passed around and used as arguments.
* Decorators is function that takes another function and extends the behaviour of the latter function. It adds extra feature to an existing function.
* When passing functions to another function, we just give the function name but not the function call as shown below :
* When we decorate a function, we create a wrapper around that function and in the wrapper, I add the functionality. Ex-



* It can be used to time functions, create input/output caches, logging purposes

functools.wraps() – it is a decorator function itself that is used to preserve the metadata of original function when creating a wrapper function.

### @staticmethod decorator

* A @staticmethod is a type of method that is bound to just the class and does not need class instance to be created (object creation) to be accessed.
* It can’t access or modify class state.

## Closures

# Comments in Python

# Keywords in Python

## Yield

* Yield keyword is used to create a generator function.
* A generator function is a memory efficient functions and can be used as an iterator object.

## assert statement

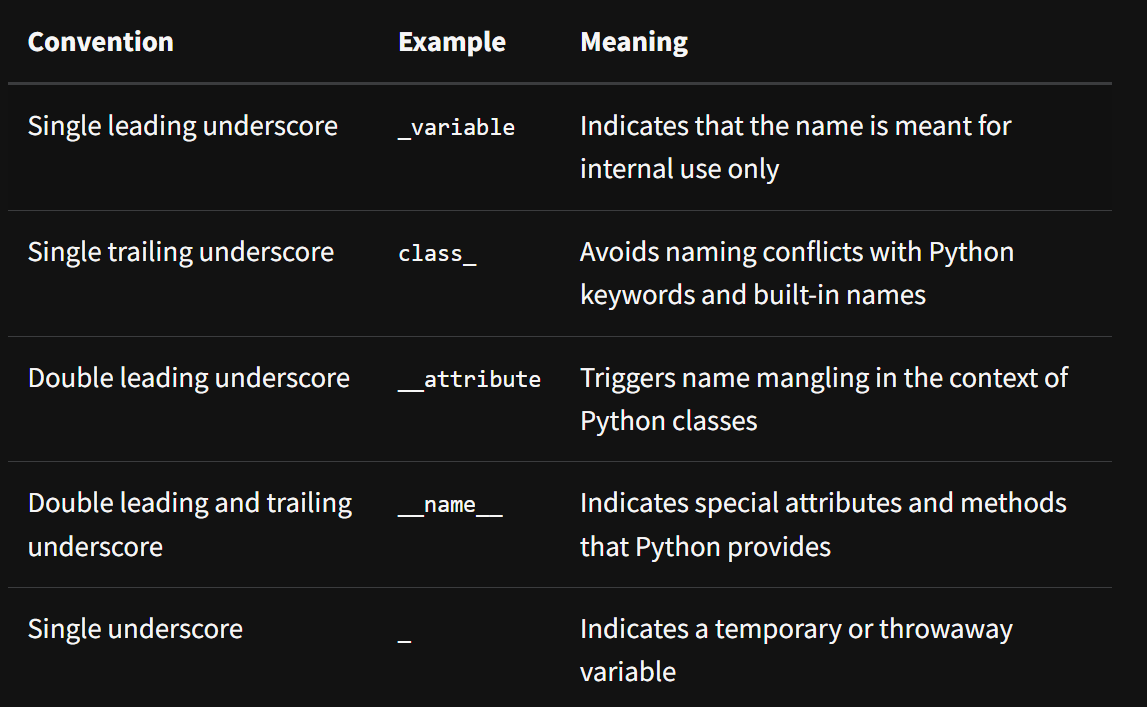
* assert statement is used check if a given condition evaluates **‘True’.** It raises an ‘**AssertionError**’.
* It is primarily used for debugging and testing purposes to validate assumptions about the code.
* Syntax:
  + - assert condition, *message [Optional]*
* Please note: assert is useful for debugging and testing purposes but not in production code. Use try & except for handling exceptions gracefully.

## exit(), sys.exit() & quit() statements

## breakpoint()

* This method is used to stop the execution of line after breakpoint() has been used.
* breakpoint() method is used to debug the code.

# Naming convention



# Regular expression

* Regular expressions are sequence of characters that represent specific patterns that we are trying to find in our text/string or code files.
* In programming these are useful for the following scenarios:
  + - Validate the input
    - Find patterns
    - Replace matches
* Examples
  + - 1. gr(a|e)y matches 🡪 {gray, grey}
      2. g(oog)+le matches 🡪 {google, googoogle, googoogoogle…}
      3. ^code matches words that begin with “code” 🡪 {codenation, codehack}
      4. code$ matches words that end with “code” 🡪 {chefcode, leetcode}

# Advanced Python

## Recursion

* When you can’t work with loops work with recursion
* Recursion is when you call a function to itself.
* Call stack
* Top-down recursion
* Bottom-up recursion
* Tail recursion
* Use of recursion

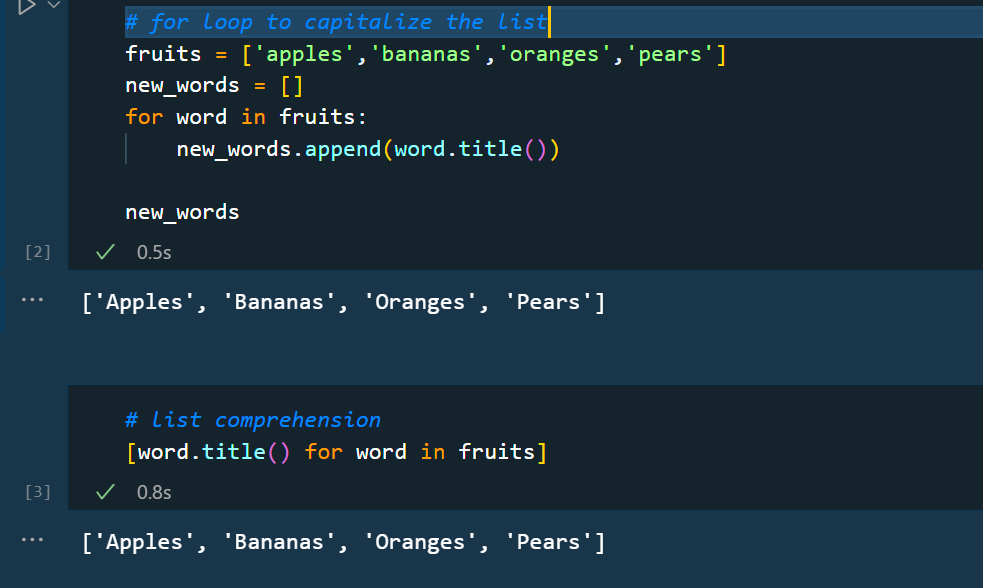
1. Backtracking
2. Tree like structure

## Generators

## Python comprehension

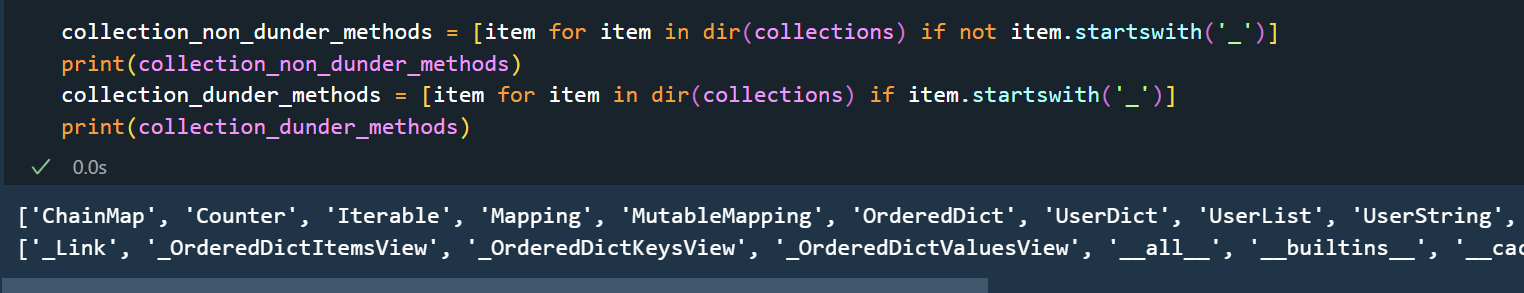
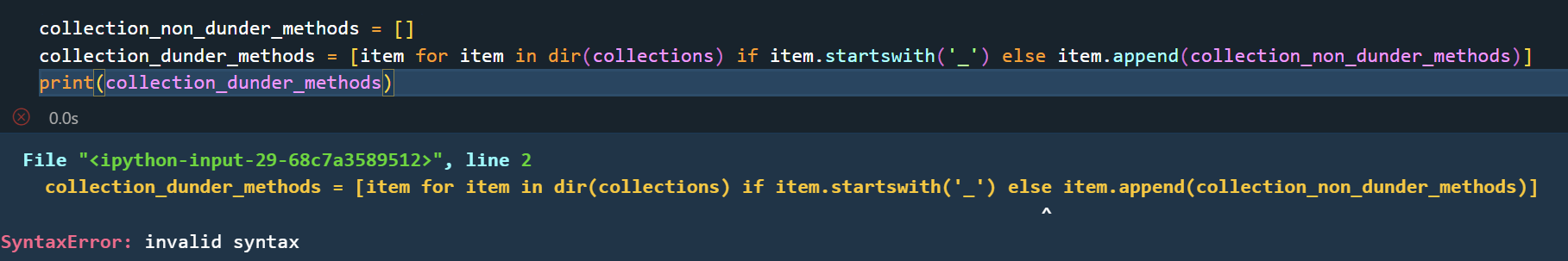
### List comprehension

* List comprehension helps in reducing the lines of codes. Improves memory performance – comprehensions are 90% times faster than for loops.



* List comprehension cannot have both if and else statements in the same comprehension. We can either use if and for else 🡪 if not.

Ex- In the following code we are trying to put each element from another list [dir(collections)] which starts with a dunder\_ and in the else list we are trying to append non-dunder starting elements, but this cannot be achieved using if – else in a list comprehension.

****

Set comprehension

Dictionary comprehension

## Parallelism in python

## Multiprocessing in python

<https://machinelearningmastery.com/multiprocessing-in-python/>

## Multithreading in python

## Context Manager and with statement

* Context managers in python are used to efficiently manage external resources such as
  + Files
  + Databases connections
  + Locks
* Context manager is used to manage

# Exceptions in Python

* Exception was incorporated in programming language in 1960s.
* Exceptions in code are related to errors in code. Error can be mainly be two kinds of :

1. Syntax error
2. Exception

* Syntax error :
* In certain scenarios, even if a statement or expression is syntactically correct, it may cause an error when it is executed. Error detected during execution are called ***exceptions.***
* order of handling exception also matters based on the hierarchy of class

## Exception propagation

* Stack trace
* Unhandled exceptions
* try….except statement
* try…except….finally statement
* try….except…else….finally statement

**try:**

* code that we want to protect
* some potential exception – guarded code

**except <ExceptionType> as ex:**

* code that will run if that specified code has some kind of problem at execution
* <ExceptionType> occurs (or any subclass of exception)

**else:**

* This is an optional block that runs if the code in “try block” executes successfully without raising any exceptions.
* It is used for code that should only run when no error occurs.

**finally:**

* This block always executes regardless of whether exception was raised or not.
* It is used for cleanup actions – closing files, databases connections, releasing resources.

## Python exception handling class hierarchy

## Handling exceptions

## Raising exceptions

### raise statement

## Creating our own exception classes

### Custom hierarchy

### Extending functionality

# Reading and Writing files in python

1. Variables can store data until our program is running
2. Files

## File handling in Python

### Output generation styles

There are several ways to give the output of a program

1. data can be printed in a human-readable form
2. written to a file – for future read

There are 3 ways to do it :-

1. expression statements
2. print() statements
3. write() statements

this standard output file can be referenced as ***sys.stdout***

## Global Interpreter lock

## Data buffer

File Object

### print statement

# Working with Web scraping

The following libraries are useful in web scraping

* + - 1. webbrowser – comes with python and opens a browser to a specific page.
      2. requests – downloads files and webpages from the internet.
      3. urllib -
      4. bs4 – parses HTML, the format that web pages are written in.
      5. selenium – Launches and controls a web browser. This module can fill in forms and simulate mouse clicks in the browser.

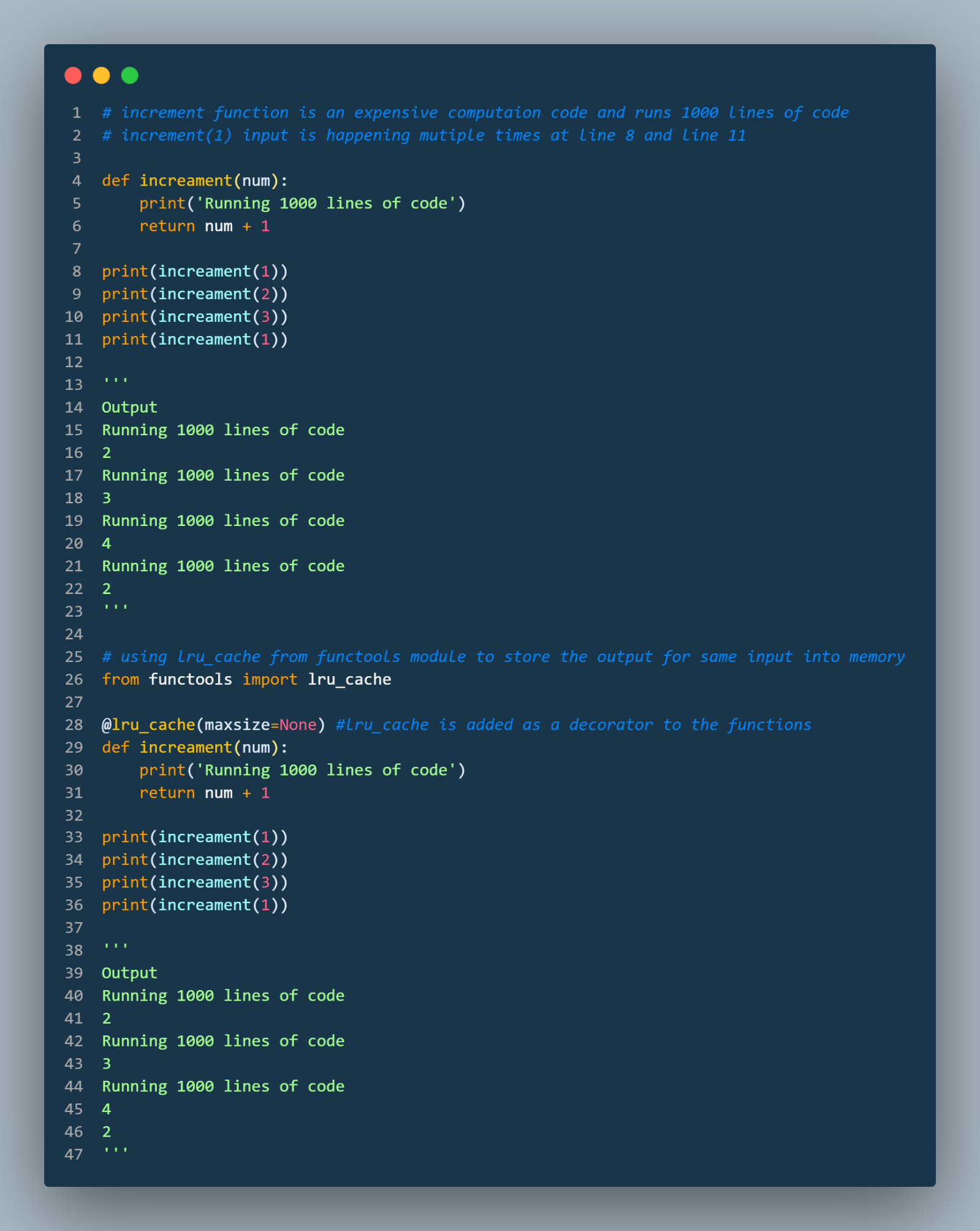
# Design Pattern in Python

# Optimization in Python

## Computational complexity

## Memoization

* It comes under dynamic programming in python.
* It is an optimization technique in computer programming **to reduce the execution time of a function** by **caching** its results for specific inputs.
  + A cache is simply a temporary data store that holds data so that future request of that data can be served faster.
* Memoization is used for functions that are computationally very expensive or involves repetitive calculation.
* <https://towardsdatascience.com/mastering-memoization-in-python-dcdd8b435189>
* Example



# Logging in Python

* Just like debugging, logging is also an essential toolbox for any developer. Logging is a process when you want record relevant information about the execution of your code. These records are called logs which helps us in tracking the flow of code execution.
* Python provides a logging system as part of its standard library. With Python logging :
  + - 1. you can create and configure loggers
      2. set log level
      3. format log messages
* Why to use logging ?
  + Debugging
  + Performance analysis
  + Monitoring usage patterns
  + Logging should be always preferred as standard coding practice over print() because it decrease maintenance burden
* Working with python logging module
  + Logger – Logger is the main component of a logging module. Think of this as a reporter to your code. Logger can decide what to record, at what level of details, where to store or send these records.
  + Root logger
  + Default logger
  + Based on severity level, logger have set up basic logging configuration

## Level of logging

|  |  |  |
| --- | --- | --- |
| Log Level | Function | Description |
| DEBUG | logging.debug() | Provides detailed information that is valuable to you as a developer. |
| INFO | logging.info() | Provided general information about what’s going on with your program. |
| WARNING | logging.warning() |  |
| ERROR | logging.error() |  |
| CRITICAL | logging.critical() |  |

## Log formatter

Log formatter to style your log messages.

## Redirect log records with handlers

StreamHandler

FileHandler

## Define logging rules with filters.

## Links

<https://toptal.com/python/in-depth-python-logging>

# Debugging in Python

# Error Handling in Python

# Code documentation

Essentials for code documentation

* Add comments, API documentation, docstrings
* Make your code easy to read, easy to change and easy to test
* Clear naming and clear structure

Comments should be used to tell the user what is happening at a particular section of code in a crisp and crystal manner. Use better comments extension of VS code.

Type hints

Docstring

ChatGPT - Copy code to chatGPT and ask to put documentation to code.

mkdocs – this can help you to create a website from documentation created.

# Python code quality

<https://realpython.com/python-code-quality/>

Import statements

1. Always keep imports from same module into a single line for better organization.
2. Avoid using “ \* “ i.e. don’t import all functions from a module
3. Keep imports in organised manner – standard library imports first, 3rd party liberaries next & custom/local module imports last.
   * 1. 1st 🡪 Standard modules
     2. 2nd 🡪 3rd party modules
     3. 3rd 🡪 Custom/Local modules

Function optimization

1. While writing functions always keep your arguments minimum.
2. Ideally, it should have maximum 2-3 arguments, if you need more arguments to be provided to your function create a config object which can be passed to multiple functions.
3. Provide annotation to arguments, parameter and function return type.



# Errors in Python

There are 3 types of errors:

1. Compile time error – syntax error. Ex- missing (:) colon, indentation error
2. Logical error – code gets compiled and gives output, but the output is wrong. Ex- 2+3 = 4
3. Runtime error – when inputs are given by user are wrong. Ex- 5/2 = 2.5 but 5/0 will give a zero-division error.

Among these 3, compile time error is the easiest to debug.

Logical error can also be handled with a testing team.

But runtime error is not done by you but by the user. These issues should be handled by developer.

Every statement you write in any language can be of two types

* + Normal statement
  + Critical statement

## Runtime error

1. IndexError: list index out of range

Empty pandas dataframe getting to be converted into spark dataframe with schemas as shown

1. Py4J - <https://www.py4j.org/py4j_java_protocol.html>
2. Struct type - <https://docs.python.org/3/library/struct.html>
3. FileNotFoundError
4. TypeError - <https://www.freecodecamp.org/news/python-typeerror-int-object-not-subscriptable-solved/#:~:text=Why%20the%20%22TypeError%3A%20'int,an%20integer%20as%20an%20array>
5. NameError
   * This error occurs when the interpreter encounters a reference to a variable or name that doesn’t exist in the current scope.
   * Following scenarios are possible
     + 1. Undefined variable
       2. Misspelled variable name
       3. Variable not defined in scope
       4. Using a Name before assignment
6. KeyError
   * This error happens when you try to access a dictionary key that does not exist. This error typically happens when you use the index operator (‘[ ]’) to access a dictionary element, but the specific key is not present in the dictionary.
   * You can use dict.get() method to avoid this kind of errors.
7. SyntaxError
   * Unexpected EOF while parsing – this statement of error tells that python interpreter reached the ‘EOF – End of File’ while **it was still expecting more codes**.
8. ValueError
9. ImportError
10. ImportError: cannot import name 'Thread' from partially initialized module 'threading' (most likely due to a circular import) (D:\Coding\git repository\threading.py)
11. **Circular import -**  Circular import leading to import error and it happens when directly or indirectly module A imports module B and module B import module A
12. Remove any compiled file with extension (.pyc) from the project directory.
13. Rename your “.py” file
14. Verify your import statement.
15. Lastly, restart your python environment.
16. OSError
    * **OSError is a base class** for various operating system related errors in python.
    * This means OSError acts a parent class for many operating system related system error in python like FileNotFoundError, PermissionError, IsADirectoryError, NotADirectoryError, FileExistsError.
    * It is raised when errors like file or directory not found happens, permission is denied or other system related issues happens.
17. AttributeError
18. RecursionError
    * RecursionError –
19. UnboundLocalError
    * It occurs when a local variable is referenced before it has been assigned a value within the local scope.
20. ModuleNotFoundError
21. Error

# Python Runtime

How a python code runs on CPU when we executed it ?

1. The code with ‘.py’ extension must pass through interpreter (CPython or whatever type you are using).
2. The interpreter has 3 main parts :
   1. Compiler
   2. Bytecode
   3. Python Virtual Machine (PVM)

Following process are performed when a python program runs:

1. Source code parsing

* python interpreter reads source code file while is a **‘.py’** file and parses it line by line.

1. Tokenization

* Also known as lexical analysis or lexing
* it is process of breaking down source code to smaller meaningful units called **tokens/lexens.** performs lexical analysis also known as tokenization.
* Identifies keywords, variables, operators etc.
* Keywords: if, for, def, class, while etc are reserved keywords in python.
* Identifiers: Variable names, class names, function names etc.
* Literals:

1. Numeric literals like ‘123’, ‘3.14’
2. String literals like ‘Hello World!’
3. Boolean literals like ‘True’ & ‘False’

* Operators:

1. Arithmetic
2. Comparison
3. Logical

* Punctuations symbols like parenthesis ‘()’, square brackets ‘[]’, curly braces ‘{}’, colons ‘:’, semi-colons ‘;’

1. Parsing

* Also known as syntactic analysis
* AST (Abstract syntax tree) is created in this step.

1. Bytecode generation
2. Bytecode execution
3. Dynamic memory allocation
4. Importing modules & libraries
5. Execution flow
6. I/O operation
7. Exceptional handling
8. Program termination

## [Python interpreter](Useful%20Modules%20in%20Python.docx)

1. It is the program used to run the Python code and scripts.
2. It is a software layer between your program and your computer hardware to get the code running.
3. Depending upon the python implementation, you can have following interpreters:

CPython – program written in C language.

Jython – program written in Java to interact with Java libraries, implementation of Python for JVM.

PyPy – program written in Python itself.

IronPython – program implemented in .NET framework.

MicroPython – program written for microcontroller and embedded systems.

Brython – web browser implementation of python to convert python code to javascript.

# Python Environment

* Python environment is a folder structure that gives you everything you need to run a lightweight yet isolated python environment.
* A Python environment is a context which refers to the specific configuration and set of installed packages and dependencies that you use to run Python programs. It includes the Python interpreter, libraries, modules, and other tools necessary to execute your code.
* When working on a Python project, it's generally a good practice to use virtual environments to manage your dependencies and ensure project isolation. This way, you can avoid conflicts between different projects and easily replicate your environment on other machines.
* Some standard modules to create environment – pyenv

## Python virtual environment

Some commonly used Python environments:

1. System Python

* This refers to the Python interpreter that comes pre-installed with your operating system. It may not have all the packages or the desired version of Python that you need for your project.
* It's generally recommended to avoid modifying the system Python to prevent potential conflicts with the operating system.

1. Virtual Environments

* Virtual environments allow you to create isolated Python environments for different projects.
* They provide a way to install specific packages and dependencies without affecting the system Python or other projects.
* The popular tools for creating virtual environments in Python are venv, virtualenv, and conda.

1. Anaconda
   * + Anaconda is a distribution of Python that includes the Anaconda package manager, conda. It comes with a wide range of pre-installed scientific computing packages and tools.
     + Anaconda simplifies the management of packages and environments, particularly for data science and machine learning projects.
2. IDE-specific environments

* Some integrated development environments (IDEs) like PyCharm, Spyder, and Visual Studio Code have their own mechanisms for managing Python environments.
* They often provide features like virtual environment creation, package management, and project-specific configurations.

1. Containerization

* Docker is a popular tool for creating lightweight, isolated containers that encapsulate your Python environment.
* With Docker, you can package your application, along with all its dependencies, into a portable container that can run on any system with Docker installed.

To create a virtual environment using venv, you can follow these steps:

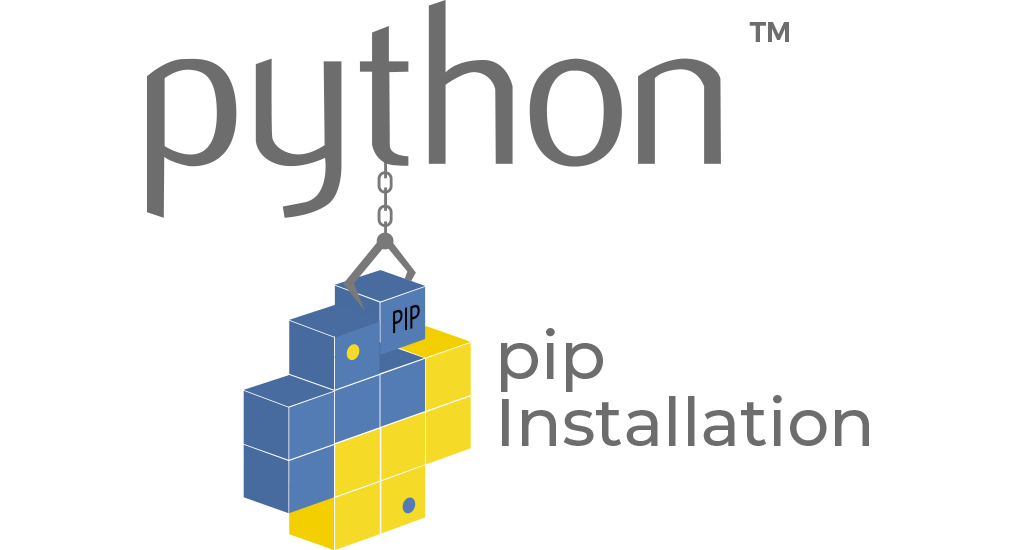
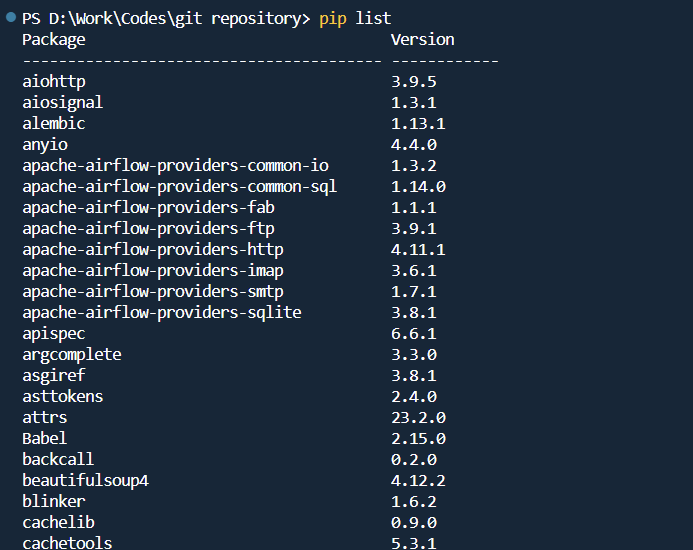
1. Open a terminal or command prompt.
2. Navigate to your project directory.
3. Run the command: python -m venv myenv (Replace myenv with the desired name of your virtual environment).
4. Activate the virtual environment:
5. On Windows: myenv\Scripts\activate.bat. You will see the prompt change to indicate that you are now working within the virtual environment.
6. Install the required packages using pip.
7. Run your Python code within the activated virtual environment.

Remember to activate the virtual environment each time you work on your project to ensure that you are using the correct Python environment.

## Python Versions

Link - <https://realpython.com/intro-to-pyenv/\>

## PIP

* PIP stands for Python Installation package. It is used to manage packages and dependencies that do not comes with python standard modules.
* PIP is used to install, update and uninstall packages.
* Commonly used pip commands
  + - 1. To check PIP version – “pip --version”
      2. To update PIP – “pip install –upgrade pip”
      3. To install a package – “pip install <package\_name>”
      4. To uninstall a package – “pip uninstall <package\_name>”
      5. To update a package – “pip install <package\_name> --upgrade”
      6. To list all the installed packages – “pip list”
      7. To get details about a package – “pip show <package\_name>”



# Python scripts, Modules, Packages and Library

In Python, both packages and modules are organization structure for organizing and managing the code. However, they both serve their own purpose and distinct characteristics.

### Modular programming

* Modular programming refers to the process of breaking a large complex programming task into separate small and more manageable subtasks called **modules.**
* These individual modules are then clubbed together to form a larger application.

### Python script

* A python script is a standalone file with ‘.py’ extension.

### Python Module

* A module is a single python file containing codes.
* A module has codes related to a specific kind of domain or functionality with various object related to it 🡪 functions, classes, variables.
* A module can be imported and used by other python modules/scripts. Example – math.py, datatime.py, os.py are modules used by programmer to build their specific python codes.
* Types of Modules in Python
* Import statement
* Module Search path

### Python Packages

* A package is a collection of various modules kept in a file like hierarchy format.
* A package is identified by presence of \_\_init\_\_.py script inside the package directory.

### Python Library

* A Library is collection of modules and packages. Examples – Numpy, pandas, matplotlib, TensorFlow, and requests.
* There various kind of libraries in python:

1. Standard libraries
2. Third party libraries
3. Data Validation libraries
4. Web development libraries
5. Scientific computation libraries
6. Machine learning libraries
7. Database libraries

# Python Debugging

# Memory Management in Python

1. Although, python is not the fastest programming language, but it has a good memory management. Python memory manager is responsible to manage the memory for you and you can just focus on the code part.
2. Python does not allow you much freedom in managing memory unlike other languages like C++ where you can manually allocate and free memory.
3. Memory management is crucial to avoid performance issues in code in python.
4. Python build-in tools can also be used to monitor your python code.
5. <https://www.geeksforgeeks.org/memory-management-in-python/>

## Garbage collection

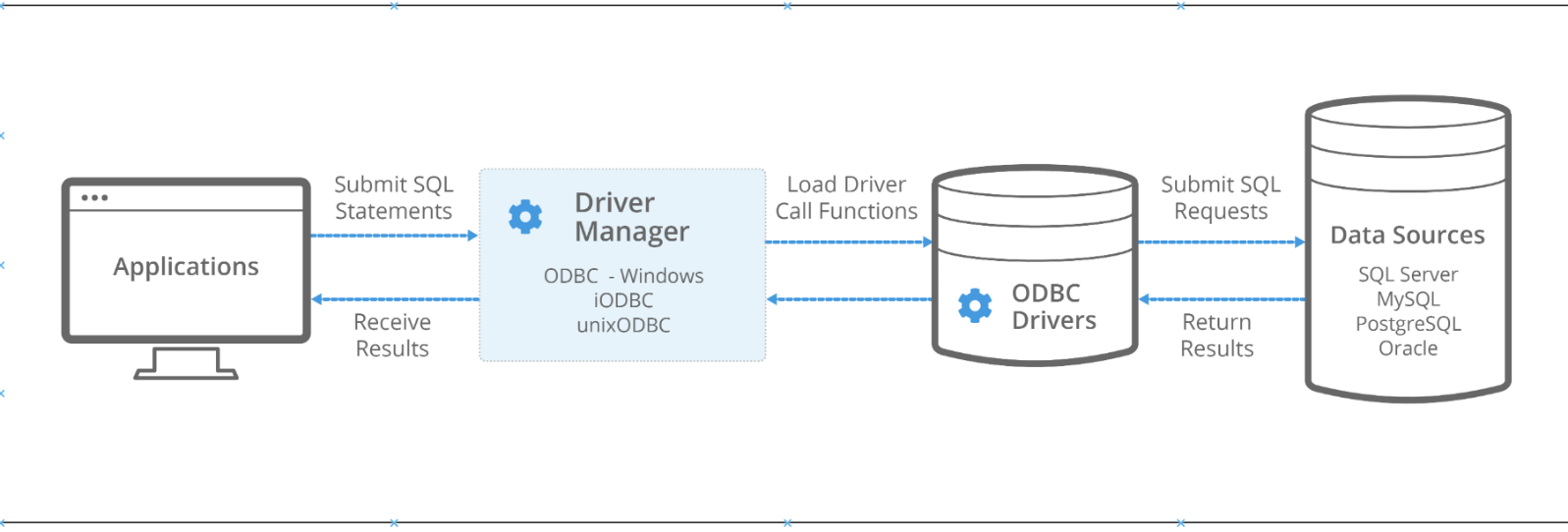
# Database Connection with Python

To connect any database using python application, we ensure following

* Drivers are present and installed for the type database we are connecting. Ex- SQL Server driver for SQL Server, Oracle driver for Oracle DB
* Certification error – Proper SSL certificated are installed.

ODBC

* ODBC – Object Database connectivity



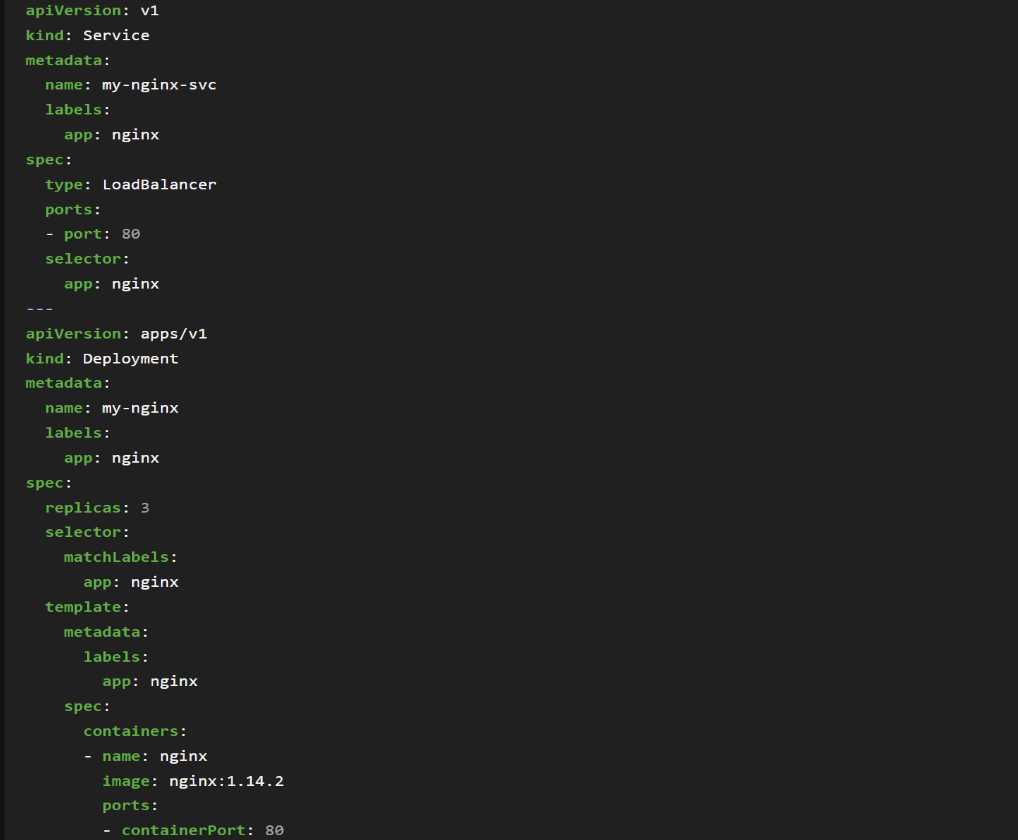
Libraries for databases :

* + - 1. pyodbc
      2. pymssql
      3. SQLite
      4. Sqlalchemy
      5. Pg8000 – A PostgresSQL database adapter for Python.

## SQLAlchemy

* SQLAlchemy is a popular ORM for Python. It allows you to interact with database using Python objects instead of writing SQL queries.

# Working with YAML

* Python uses PyYAML (import yaml) library to work with YAML files.
* Although YAML is not part of standard Python Libraries, while XML and JSON are. (protocol buffers & Avro files are also used for configurations and data exchange)
* YAML stands for “YAML Ain’t Markup Language”.
* It is human readable and data serialization language.
* It is popular because of 2 factors :
  + - 1. Human readable
      2. Simple to understand
    - YAML application
      1. Writing configuration files – used in DevOps domain for tools like Kubernetes, Ansible.
      2. Data persistence - <https://www.datastax.com/blog/what-persistence-and-why-does-it-matter>
      3. Internet messaging
      4. Cross language data sharing
      5. Multiple documents can be stored in one YAML file using --- separator. This feature is used in Kubernetes as shown below.
    - It has extensions like .yml and .yaml

## Structure of YAML

* A YAML format primarily has 3 node types
  + - 1. Maps/Dictionaries – key : value pairs
      2. Arrays/Lists
      3. Literals
* Unlike JSON, YAML can support comments using ‘#’. YAML supports single line, inline comments but does not support multiple-line comment, for multiple line comment use ‘#’ at multiple lines.

json.dumps()

# Working with JSON

* JSON stands for JavaScript object notation. It is used to communicate between different application in system.
* It is used to exchange data between different applications for storing data, configuration files and APIs.
* json library comes with standard python library package. It is important to note json uses double quotes (“ “) instead of single quotes ( ‘ ‘).
* We deal with json in 3 different ways:

1. JSON as a string
2. JSON as a file
3. JSON from services (API responses)
4. JSON as a string

* For working with json, we need to import json module.
* While working with json we need conversion from string to dictionary (json.loads) and dictionary to json like string (json.dumps).
* JSON string does not recognize “None” value and it should be replaced by “null” value for any empty values in JSON

1. JSON as a file

* json.dump and json.load is used to work with json files.
* We use context manager to read or write files from and to in files.json format.

1. JSON as a service (API responses)

# Working with XML

# Notes

1. f-string
2. to convert excel to json

<https://www.programmingfunda.com/how-to-convert-excel-to-json-in-python/#:~:text=)%20print(json_data)-,Code%20explanation,of%20Python%20Dictionaries%20to%20JSON>.

1. Reading an excel file

<https://www.marsja.se/your-guide-to-reading-excel-xlsx-files-in-python/>

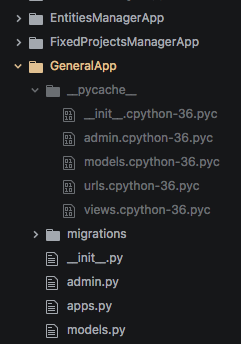
## Python Object core

There are 3 core properties of objects in python: identity, type and value.

## PEP8

## \_\_pycache\_\_

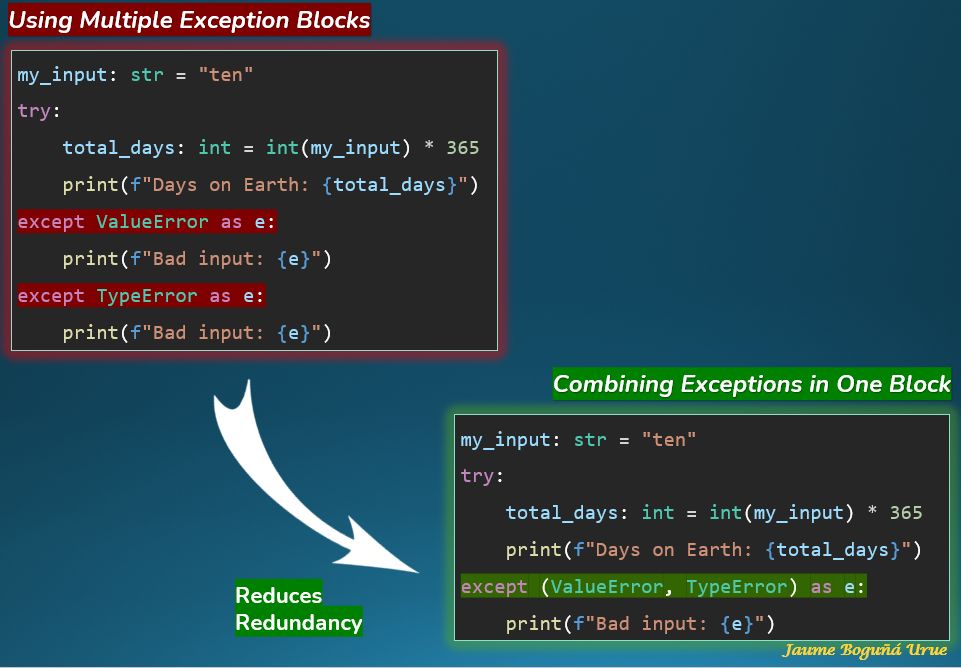
* It is directory automatically created by Python interpreter when a module is imported.
* It contains complied bytecode files (`.pyc`) of the imported modules.



## Python wheel file

* A python wheel file is a type of built package used for distributing Python software.
* It has a file extension ‘.whl’ and is defined to simplify installation process of python packages.

# Tips and Tricks in Python



LRU cache :

Use the @functools.lru\_cache decorator to optimize performance-intensive functions.

When working with large datasets or complex algorithms, you often encounter functions that take significant time to execute because of repetitive calculations with the same input parameters.

The @functools.lru\_cache decorator can be applied to these functions to store the results of expensive function calls and return the cached result when the same inputs occur again.

This significantly reduces the execution time for these functions, especially in data-heavy or computation-intensive tasks.

When to Use

Repetitive Computations with Identical Inputs: Apply it to functions that are called repeatedly with the same arguments, where each call performs the same computation.

Performance-Intensive Functions: Use it for functions that are computationally expensive, such as those involving heavy data processing, complex calculations, or deep learning algorithms.

Limited and Predictable Input Variance: It's most effective when the function has a relatively small and predictable set of input values, making caching more efficient.

# To merge with Above Python\_continued……

1. f-string
2. to convert excel to json

<https://www.programmingfunda.com/how-to-convert-excel-to-json-in-python/#:~:text=)%20print(json_data)-,Code%20explanation,of%20Python%20Dictionaries%20to%20JSON>.

1. Reading an excel file

<https://www.marsja.se/your-guide-to-reading-excel-xlsx-files-in-python/>

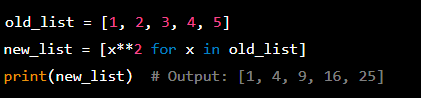
### List Comprehension in Python

In Python, there are two types of list comprehensions: basic and nested.

1. Basic list comprehension:

A basic list comprehension is a concise way of creating a new list by transforming or filtering an existing list. It has the following syntax:

* **expression** is a transformation or computation to be performed on each element of the iterable
* **item** is an element of the iterable
* **iterable** is the existing list or sequence
* **condition** is an optional condition that filters the elements of the iterable based on a boolean expression

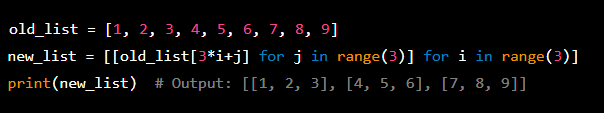
For example, the following code uses a basic list comprehension to create a new list of squared values from an existing list of integers:

1. Nested list comprehension:

A nested list comprehension is a way of creating a list of lists (or a multi-dimensional list) by applying a list comprehension to each element of an existing list. It has the following syntax:



* **expression** is a transformation or computation to be performed on each element of the inner iterable
* **item** is an element of the inner iterable
* **iterable2** is the inner iterable
* **item2** is an element of the outer iterable
* **iterable1** is the outer iterable
* **condition** is an optional condition that filters the elements of the outer iterable based on a boolean expression

For example, the following code uses a nested list comprehension to create a 3x3 matrix (a list of lists) from an existing list of integers:

In this example, the outer iterable is a range from 0 to 2 (inclusive), and the inner iterable is a range from 0 to 2 (inclusive). The expression is the indexing operation that extracts the appropriate element from the old\_list.

## warning module

## Py4j library

* It is a library to communicate between python and java programs

## datetime module

## PyYAML module

<https://python.land/data-processing/python-yaml>

<https://pyyaml.org/wiki/PyYAMLDocumentation>

## Error in Python

There are 3 types of errors:

1. Compile time error – syntax error. Ex- missing (:) colon, indentation error
2. Logical error – code gets compiled and gives output, but the output is wrong. Ex- 2+3 = 4
3. Runtime error – when inputs are given by user are wrong. Ex- 5/2 = 2.5 but 5/0 will give a zero-division error.

Among these 3, compile time error is the easiest to debug.

Logical error can also be handled with a testing team.

But runtime error is not done by you but by the user. These issues should be handled by developer.

Every statement you write in any language can be of two types

* + Normal statement
  + Critical statement

## Error Handling in Python

### Error in runtime

1. IndexError : list index out of range

Empty pandas dataframe getting to be converted into spark dataframe with schemas as shown

1. Py4J - <https://www.py4j.org/py4j_java_protocol.html>
2. Struct type - <https://docs.python.org/3/library/struct.html>

## File handling in Python

### Output generation styles

There are several ways to give the output of a program

1. data can be printed in a human-readable form
2. written to a file – for future read

There are 3 ways to do it :-

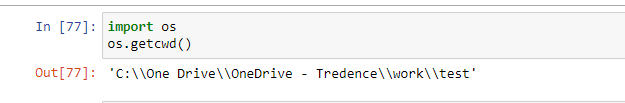
1. expression statements
2. print() statements
3. write() statements

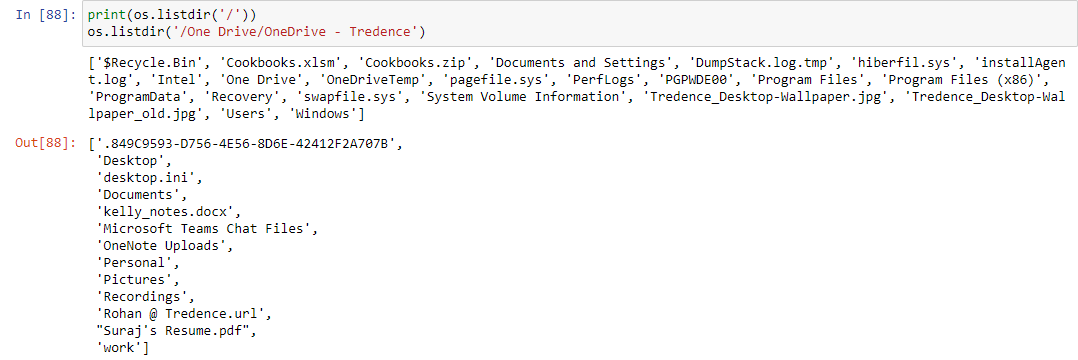
this standard output file can be referenced as ***sys.stdout***

### Reading and Writing files

# Others

## OS module

1. To get current working directory
2. To list files in directory

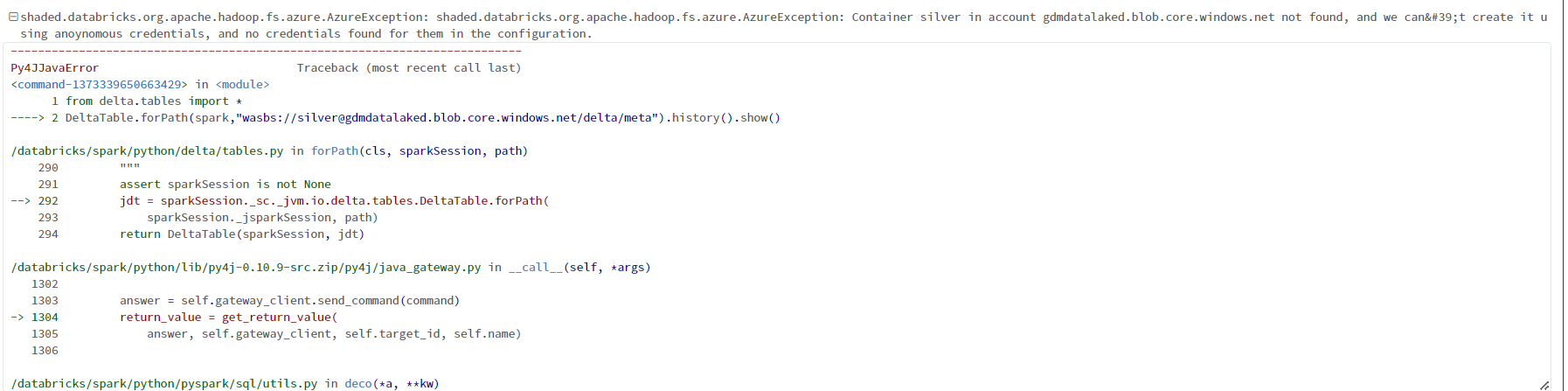


## Notes

To delete a complete line of code 🡪 **ctrl + d**

## Errors in databricks

* + 1. Py4JJavaError –

This error happened because the storage account(ADLS) cannot be accessed since it was not mounted to databricks.

2)

## Notes

1. Managed and External tables – <https://docs.cloudera.com/HDPDocuments/DLM1/DLM1.5.1/administration/content/dlm_managed_tables_external_tables.html>
2. <https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-use-databricks-spark>
3. Log4j – to capture logging information

