

Report for Python Project



**Name : Saleh Ahmed Alheeh**

**ID : 201614721**

ICS313

**Outline**

1. **Brief about Python.**
2. **Introduction**
3. **File Name Extinction**
4. **History of Python**
5. **Python Applications:**
6. **Environment**
7. **Important Reasons make you use Python**
8. **Advantages**
9. **Disadvantages**
10. **Python Programming Examples**
11. **Data types**
12. primitive data types
13. Complex data types:
14. **Memory management**
15. **Python Type checking**
16. **Exception Handling**
17. **Exception Handling**
18. **Assertions**
19. **Basic method structure**
20. **IF statements in Python**
21. **If statement Example**
22. **FOR loop in Python**
23. **For loop Example**
24. **Array structure in Python**
25. **Array example :**
26. **overloaded operators**
27. **Addition Symbol : +**
28. **Multiplication Symbol : \***
29. **reserved words**
30. **Why Python is special language.**
31. **Comparing Python to Other Languages**

**1-Brief about Python.**

* + - 1. Introduction

Python is a high level, multi-paradigm programming language. It was created by Guido van Rossum, and released in 1991. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages.

Influenced by

ABC, ALGOL 68, APL, C, C ++, CLU, Dylan, Haskell, Icon, Java,] Lisp, Modula-3, Perl, Standard ML

Python logo:



* + - 1. File Name Extinction
* **py** - Regular scripts
* **pyc** - compiled script (Bytecode)
* **pyo** - optimized pyc bytecode file (As of Python3.5, Python will only use pyc rather than pyo and pyc)
* **pyw** - Python script for Windows that is executed with pythonw.exe
* **pyx** - Cython src to be converted to C/C++
* **pyd** - Python script made as a Windows DLL**(see figure 1).**

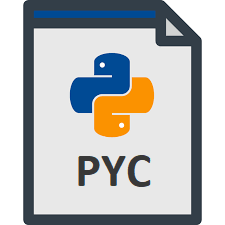


Figure 1 : File Name extinction

1.3 History of Python

The first ever version of Python(i.e. Python 1.0) was introduced in 1991. Since its inception and introduction of Version 1, the evolution of Python has reached up to Version 3.7 (2018).

Here is the brief chart depicting the timeline of the release of different versions of Python programming language.(see figure 2)



Figure 2: History of Python

1. Python Applications:
2. Python can be used on a server to create web applications.
3. Python can be used alongside software to create workflows.
4. Python can connect to database systems. It can also read and modify files.
5. Python can be used to handle big data and perform complex mathematics.
6. Python can be used for rapid prototyping, or for production-ready software development.(see figure 3 and 4)

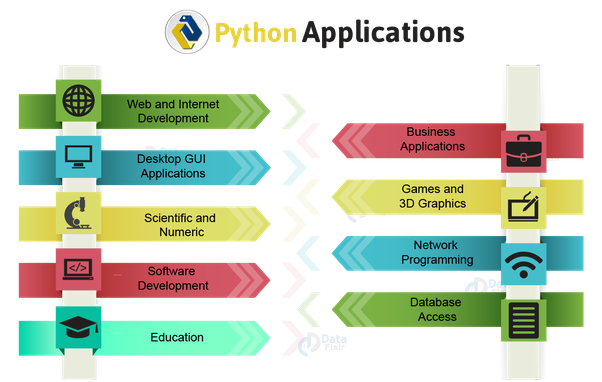


Figure 3: Python Applications



Figure 4: Companies using Python

1. Environment:

Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

# **Important Reasons make you use Python**

**1) Readable and Maintainable Code**

**2) Multiple Programming Paradigms**

**3) Compatible with Major Platforms and Systems**

**4) Robust Standard Library**

**5) Many Open Source Frameworks and Tools**

**6) Simplify Complex Software Development**

**7) Adopt Test Driven Development**

1. Advantages of Python :
2. **Easy to learn**

Python is very easy to get started by learning programming. Python contains unusually easy compositions, as already mentioned.

1. **Free and open source**

Python is an example of free open source software

1. **High level programming language**

When you write programs in Python, you don't need to pay attention to fine-grained details such as memory management used by your program, etc.

1. **Portable**

Due to its open source nature, many platforms operate. All of your Python programs can work on any of these platforms without requiring any changes at all if you are accurate enough to avoid any system-dependent features.

1. **Object Oriented**

Python supports both procedural as well as object-oriented programming.

1. **Multiple uses**

Python is versatile as it is not specific to specific usage

1. **Readability and writability**

easy to read and write

1. Disadvantages of Python:

### Speed

Python is **slower** than C or C++. But of course, **Python** is a high-level language, unlike C or C++ it's not closer to hardware.

### Mobile Development

Python is not a very good language for **mobile development** . It is seen as a **weak language** for mobile computing. This is the reason very few mobile applications are built in it like Carbonnelle.

### Memory Consumption

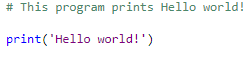
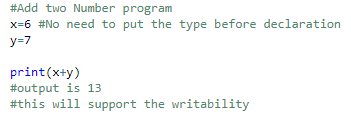
Python is not a good choice for **memory intensive** tasks. Due to the flexibility of the data-types, Python's memory consumption is also high.

### Database Access

Python has limitations with **database access** . As compared to the popular technologies like JDBC and ODBC, the Python's database access layer is found to be bit underdeveloped and **primitive** .

### Runtime Errors

Python programmers cited several issues with the **design** of the language. Because the language is **dynamically typed** , it requires more testing and has errors that only show up at **runtime** .

1. Python Programming Examples:

## Data types

1. **primitive data types**:

* Integer
* Float
* String
* Boolean

## Complex data types:

## Python complex()

## The complex() method returns a complex number when real and imaginary parts are provided, or it converts a string to a complex number.

## The syntax of complex() is:

## complex([real], [imag])

## real - real part. If real is omitted, it defaults to 0.

## imag - imaginary part. If imag is omitted, it default to 0.

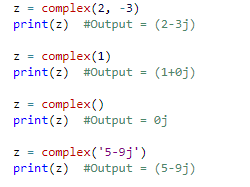


Figure 5: Example for comlex number

Also can handle complex number operations (see figure 6)

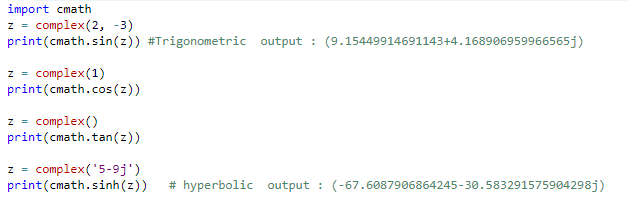
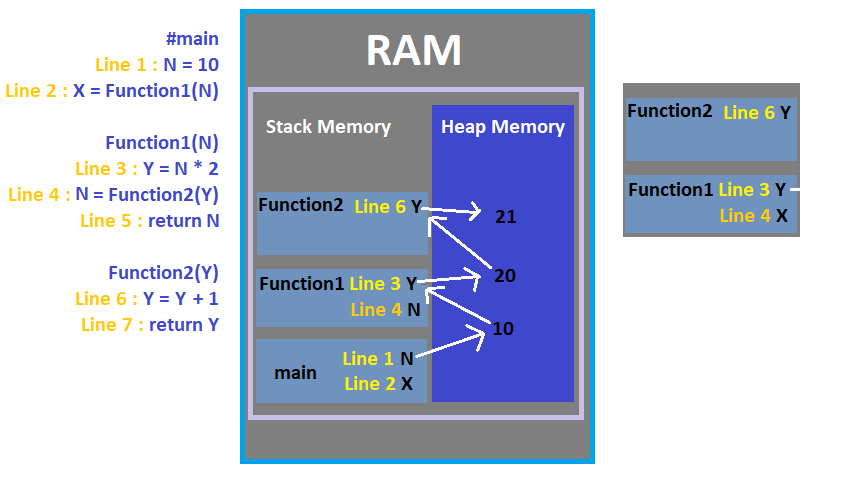


Figure 6 : Example for Trigonometric and Hyperbolic Functions

**2-Memory management**

*Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the*Python memory manager*. The Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching.*

* the methods and variables are created on stack memory.
* the objects and instance variables are created on heap memory.
* A new stack frame is created on invocation of a function/method.
* Stack frames are destroyed as soon as the function/method returns.
* Grabage celloctor is a meachaism to clean up to the dead objects.for more detailed (see Figure 7)



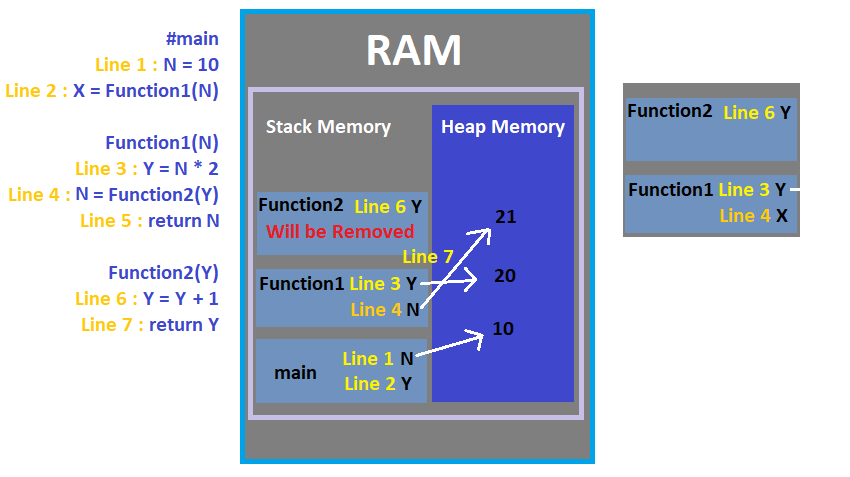
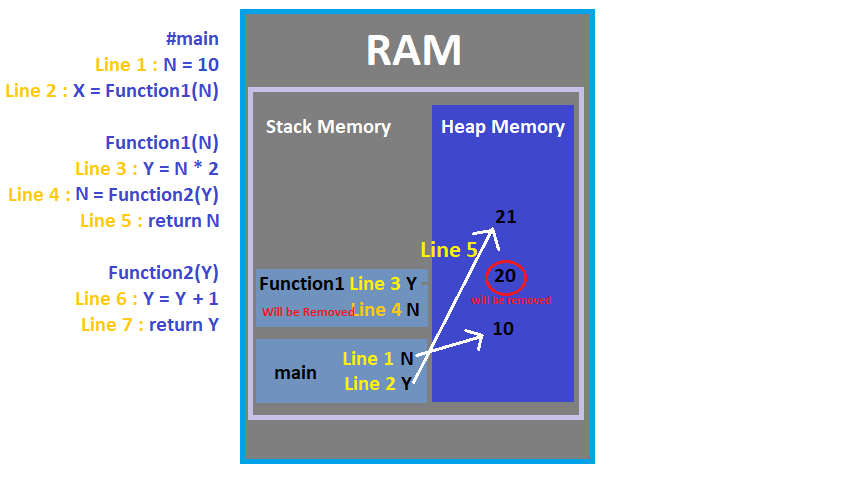
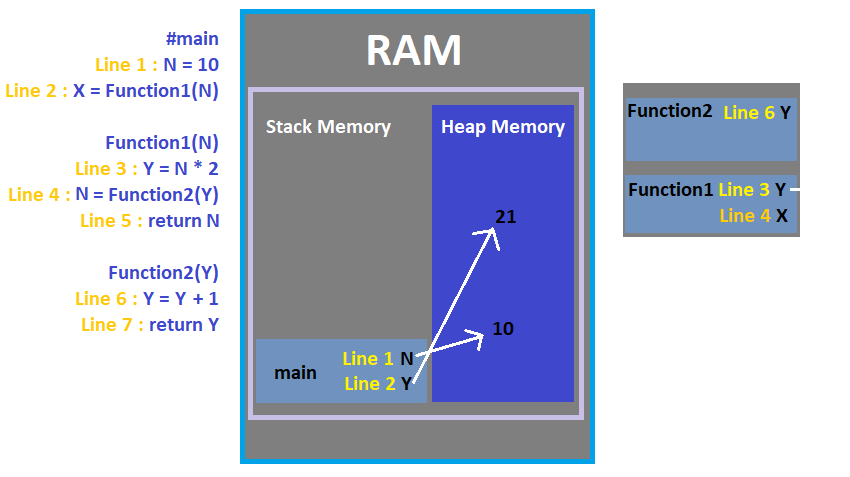


Figure 7: example Memory management

**Object 20 does not have a reference (0) and is called a dead object. Python has a method called a garbage calculator when the reference becomes equal to zero will delete the dead object from memory. The algorithm that I use garbage calculation is called the reference counter.**

**3-Python Type checking**

Python is a dynamically typed language. This means that the Python interpreter does type checking only as code runs, and that the type of a variable is allowed to change over its lifetime. (see figure 8)

if False:

1 + "two" # This line never runs, so no TypeError is raised

else:

1 + 2

3

1 + "two" # Now this is type checked, and a TypeError is raised

TypeError: unsupported operand type(s) for +: 'int' and 'str'

Figure 8 : Example Python Type checking

**4-Exception handling**

Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them

1. **Exception Handling**
2. **Assertions**

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.The error handling is done through the use of exceptions that are caught in try blocks and handled in except blocks. If an error is encountered, a try block code execution is stopped and transferred down to the except block. In addition to using an except block after the try block, you can also use the finally block. The code in the finally block will be executed regardless of whether an exception Occurs.(see figure 9)

1. **IOError**

If the file cannot be opened.

1. **ImportError**

If python cannot find the module.

1. **ValueError**

Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value .

1. **KeyboardInterrupt**

Raised when the user hits the interrupt key (normally Control-C or Delete)

1. **EOFError**

Raised when one of the built-in functions input() or raw\_input() hits an end-of-file condition (EOF) without reading any data

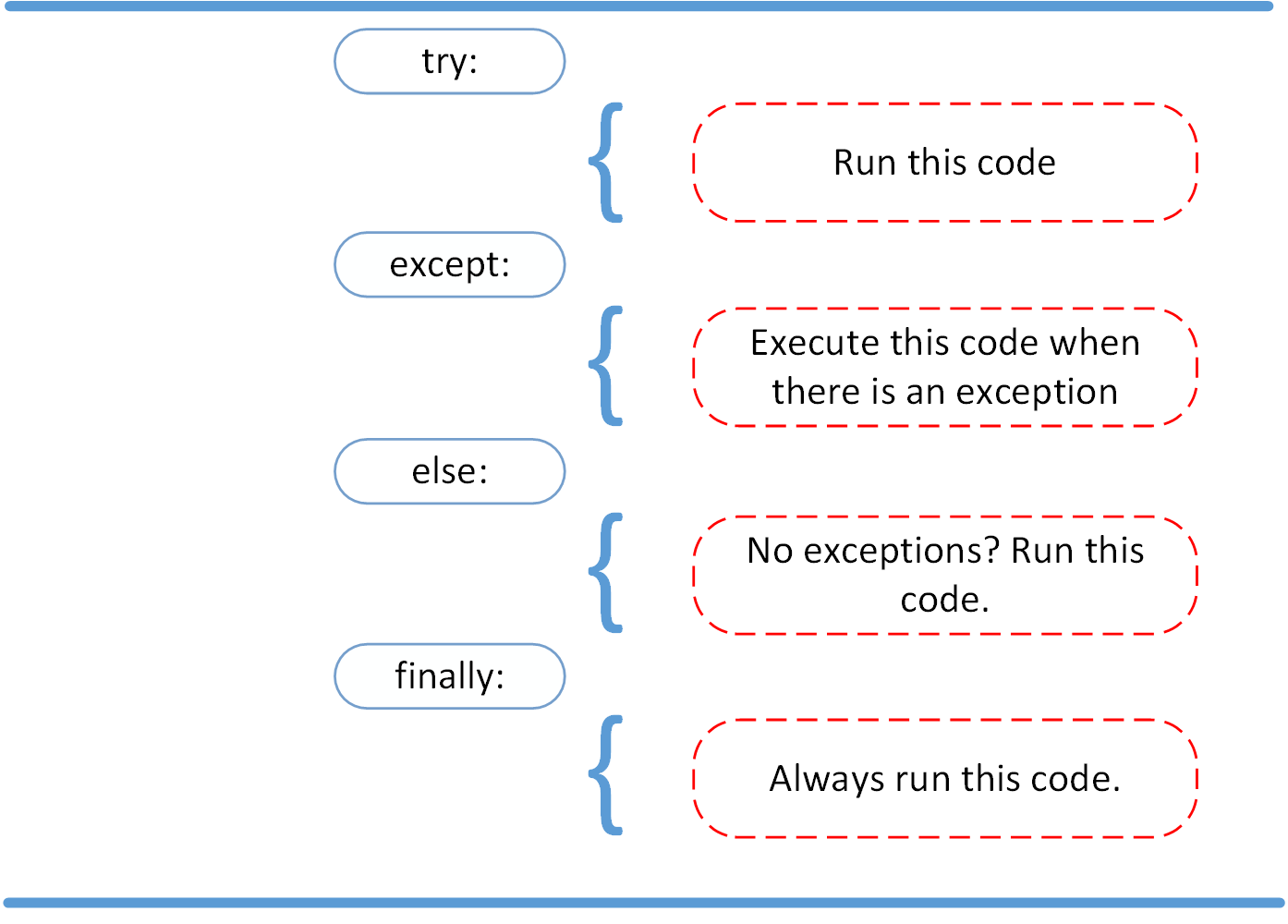


Figure 9 : try except else finally

**5-Basic method structure**

A Python method, as we know it, is much like a function, except for the fact that it is associated with an object.

a='Saleh'

1. **a.lower()**

returns the lowercase version of a string.

1. **a.upper()**

returns the uppercase version of a string.

1. **()a.strip**

if the string has whitespaces at the beginning or at the end, it removes them.

1. **a.replace('old', 'new')**

replaces a given string with another string. Note that it’s case sensitive.

1. **a.split('delimiter')**

splits your string into a list. Your argument specifies the delimite.

1. **delimiter'.join(a)**

It joins elements of a list into one string. You can specify the delimiter gain.

**6-IF statements in Python**

Python supports the usual logical conditions from mathematics:

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

1. **If statement Example:**

Num1 = 300   
Num2 = 200  
if Num1 > Num2:  
  print("Num1 is greater than Num2")

**7-FOR loop in Python**

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string.

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

1. For loop Example :

course = ["ics313", "ics353", "swe214"]  
for n in course:  
  if n == "ics313":  
    continue  
  print(n)

Output:ics353

swe214

**8-Array structure in Python**

An array is a fundamental data structure available in most programming languages and it has a wide range of uses across different algorithms.

Arrays are used to store multiple values in one single variable

1. **Array example :**

course = ["ics313", "ics353", "swe214"]  
print(course[0])

output: ics313

**9-overloaded operators**

If you’ve used the + or \* operator on a str object in Python, you must have noticed its different behavior when compared to int or float objects

1. **Addition** **Symbol : +** (see figure 10)
2. **Multiplication**  **Symbol : \*** (see figure 11)

x=1 + 2

print(x)

Output=3

# + whith two strings

A='ics313'

B='course'

Print(A+B)

output'ics313course'

Figure 10 : Example Addition Symbol : +

x= 3 \* 2

print(x)

output=6

# \*with string

'Ics313 ' \* 4

'Ics313 Ics313 Ics313 Ics313'

Figure 11 : **Multiplication**  **Symbol \***

**10-List FIVE reserved words and their meaning:**

Keywords are the reserved words in Python. We cannot use a keyword as variable name, function name or any other identifier.

### 1-None

None is a special constant in Python that represents the absence of a value or a null value.

### 2-continue

### continue are used inside for and while loops to alter their normal behavior.

### 3- class

class is used to define a new user-defined class in Python.

### 4-def

def is used to define a user-defined function.

### 5-del

del is used to delete the reference to an object.

**11-Why Python is special languages**

Because is the fastest-growing programming language in the world, as it increasingly becomes used in a wide range of developer job roles and data science positions across industries . Python is a flexible programming language Because of its readability and writeability and its set of robust libraries that make it such a dynamic and a fast. Finally, Python meets my programming needs.

1. Comparing Python to Other Languages

Python is often compared to other interpreted languages such as Java, JavaScript, Perl, Tcl, or Smalltalk. Comparisons to C++, Common Lisp and Scheme can also be enlightening. In this section I will briefly compare Python to each of these languages. These comparisons concentrate on language issues only.

1. **Java**

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing.

For these reasons, Python is much better suited as a "glue" language, while Java is better characterized as a low-level implementation language. In fact, the two together make an excellent combination.

1. **Javascript**

Python's "object-based" subset is roughly equivalent to JavaScript. Like JavaScript (and unlike Java), Python supports a programming style that uses simple functions and variables without engaging in class definitions. However, for JavaScript, that's all there is. Python, on the other hand, supports writing much larger programs and better code reuse through a true object-oriented programming style, where classes and inheritance play an important role.

1. **C++**

Almost everything said for Java also applies for C++, just more so: where Python code is typically 3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code! Anecdotal evidence suggests that one Python programmer can finish in two months what two C++ programmers can't complete in a year. Python shines as a glue language, used to combine components written in C++.