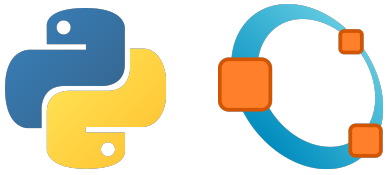
University of the Philippines Cebu
College of Science
Department of Computer Science

Design and Implementation of Programming Languages
(CMSC 124)



Python and Octave: Characteristics and Comparison

Final Project

Alex A. Diola
2019-11984
aadiola@up.edu.ph
CMSC 124-B

Dianne M. Mondido
2019-04460
dmmondido2@up.edu.ph
CMSC 124-B

1st semester
2021-2022

Table of Contents

LaTeX source code file

The LaTeX source code file can be found here:
`https://www.overleaf.com/read/pqfsqzyzffhp`.
This link, however, only gives read access as the tool used (OverLeaf) only allows for a maximum of two (2) collaborators (read-write access) in its free version, and these two are already used by the authors of the paper.

Python

Purpose and Motivation

Python is an interpreted high-level programming language that emphasizes simplicity and efficiency in its philosophy. Like any other programming language, Python has plenty of applications for almost all fields such as:

- Data Science
- Scientific and mathematical computing
- Web development
- Finance and trading
- System automation and administration
- Computer graphics
- Basic game development
- Security and penetration testing
- General and application-specific scripting
- Mapping and geography (GIS software)

Python was created with simplicity as its goal. That is why it is considered an important programming language to learn as it is commonly used in the industry. According to TIOBE index, Python is ranked 1st as the highest increase rating of 2021 and still ranked first in January of 2022. In addition, it is also the most searched Programming Language (PL) tutorial.

Python was created because the existing programming languages during that time were time-consuming to learn and hard to understand. With these limitations, Python was created.

History

On February 20, 1991, Dutch Programmer Guido Van Rossum first released his invention, Python, which actually came from an old BBC television comedy sketch series named Monty Python's Flying Circus.

Before Python came to be, Guido Van Rossum used to work at CWI working for a distributed system called Amoeba. C was the used language at his work, yet he felt that the language was too time-consuming. Therefore, he began developing a new language that supports simplicity and efficiency in his free time. What makes Python interesting is the unique mixture of C bash script capabilities and again, its simplicity. Guido Van Rossum heavily relied on the ABC programming language but also included tools for imperative programming and dynamic types.

Unlike other programming languages that rely heavily on professionals and large companies to develop features of that language, Python continuously spread and evolved around the world with the contributions from anonymous programmers, users, testers, and enthusiasts in which most of them are not IT specialists. Through Van Rossum's idea and continuous effort of the community, Python became the most used programming language across the world.

Development of Python

Guido Van Rossum published the first open-source version (Python version 0.9.0) at alt.sources. This version already includes functions, exception handling, and the core data types of list, dict, str, and others. Moreover, it had already supported object-oriented programming and had a module system.

Python 1

Python's next version (Python version 1.0) was released in January 1994. The major features of this version now supported functional programming tools such as map, filter and reduce, and lambda. In version 1.4, it brought the Modula–3 style keyword arguments and support for complex number operations.

Python 2

Python 2.0 was launched in 2000 with a huge change in its source code. It now supports many desired features such as:

1. Unicode
   The Unicode string data type now allocates 16-bit numbers to represent characters instead of the 8-bit strings. This means that there are an additional 65,000 supported symbols from non-Latin script languages like Russian, Chinese, or Arabic to non-letter characters like emojis.

2. List Comprehension
   These are used to create lists from different iterables. It takes and returns a list, allowing the user to trim to a specific need or to create a new list with newly manipulated elements.

3. Garbage Collection
   Python now switches its garbage collection cycles from counter-based system to cycle-based systems. In the counter-based system, each object contains a counter that records how many other objects are pointing to it. In deleting, the objects are automatically deleted when the counter reaches zero. The disadvantage of using this system is that it couldn't delete objects that were pointed to but were not accessible, resulting in a memory leak. The cycle-based system fixes that problem by using periodic cycles

to call the garbage collector to delete inaccessible objects. The cycles use additional overhead but reduce the risk of a memory leak. Overhead costs of the deletion cycles have since been reduced due to optimizations.

4. Assignment Shortcutes
   This version fully supports a shorter version of assigning variables. The list of supported assignment operators include `+=`, `-=`, `*=`, `/=`, `%=`, `**=`, `&=`, `|=`, `^=`, `>>=`, and `<<=`. These operators are important for simplicity and readability of the code.

5. Other Improvements
   Although Python 2 terminated its support in June of 2009, there are still some functions they added. Support for nested scopes was added to perform like other statically scoped languages. Generator routines for controlling loop behavior and creating unified types and classes into one Python hierarchy were added for a more object-oriented programming feel. Furthermore, here are some additions to the Python functionality:

   - generator expressions for function decorators, decimal data types, and iterator function
   - Added `with` statement and also its optimization
   - Backwards compatible 1.0 features such as `bin()`, `typeError`, and `_complex_()`.

Python 3

The rise of Python as a frontrunner for data science has been made possible by the big changes in Python 3. The aim of this update is to remove design flaws by deleting duplicated modules and constructs. This also means that Python 2 is not compatible with Python 3. In case the user wants to change its code from Python 2 to Python 3, there is a tool named 2to3 that will review and convert the syntax to Python 3 automatically. Listed below are the upgrades of Python 3:

1. Print Built-in Function
   In Python 2, the `print` function needed to be imported from the `__future__` module. Now, it is considered as a built-in function with arguments enclosed in parentheses.

2. NumPy Library
   With the need of data automation tools, the NumPy library became a field for data enthusiasts. It became the preferred language for data science and machine learning followed by R, and MATLAB.

3. Renaming raw-input to input
   Input now always returns a string rather than an expression. This is to

remove confusion with two similar inputs and instead favor the more common use. Using `eval(input())` will let the user use the old functionality.

4. Unification of `str` and unicode
   All text content such as strings is now Unicode by default. This is to reduce errors caused by mixing Unicode and encoded data types of the same program. If the user opts for encoded string, the user can use an immutable bytes type by using the method `bytes()` or a mutable bytes type through `bytearray()` method. The advantage of using bytes is that they can store varying data whereas Unicode only represents text-like data.

5. Integer Division
   Division operations now return an int data type instead of float. This is to remove unexpected behavior from unintentional truncation (Thelin, 2021).

6. Support for asynchronous programming
   With the use of an async module, it can now handle event loop implementations, coroutines, asynchronous iteration and context managers, and awaitable objects. The reserved keywords `async` and await, and the method `breakpoint()` were added to the module.

7. Other improvements
   Although Python 2 is incompatible with Python 3, reusing functions is still common when updating versions of the programming languages. Some changes in exception handling was implemented such as the need for the keyword as (e.g. exec as *var*). The `with` statement is now built-in like `print` function. Renaming some methods were also implemented for readability, simplicity, and writability.

Features

Python is an interpreted, free-source, object-oriented, high-level programming language with dynamic semantics (Python). From the definition itself, Python has several features that fit the needs of almost all users and fields. With its continuous support and contributions from the community, Python has yet to plummet its popularity. Here are some features of Python:

1. Easy to use
   High-level languages tend to become more human-readable than machine ones, but some of these languages aren't easy for some beginners such as Java, C++, and C. Python's philosophies are revolved around simplicity both in coding and reading that is why Python is the most searched language for it is easy to use and learn.

2. Free-Open Source
   Python is made available to the public through its website with support from different operating systems and without the need for financial subscriptions or contributions.

3. Object-oriented Programming
   Python supports OOP concepts such as encapsulation, classes, polymorphism, inheritance, etc. This paradigm helps users to reuse code for efficiency. Python caters to a lot of programming paradigms such as procedural, functional, declarative, and more.

4. Interpreters than compilers
   Python translates programs one line at a time compared to compilers that scan the entire line first and then translate them to machine code. Interpreters help to debug efficiently both in time and memory.

5. Large Standard Libraries
   For Python to be applicable to most of the different fields with their needs, a vast library should be able to help them find the solutions to their problem. There are standard libraries good for data science, web development, machine learning, and even scripting. Libraries such as NumPy, TensorFlow, Pandas, Django, Pytorch, and more are some of the most used libraries in Python for different projects in different fields.

6. Supports GUI Programming
   With its vast libraries, Python can also be used for graphics interfaces. PyQT5, Tkinter, Kivy are the libraries that are used for developing web applications.

7. Integratable to other Programming Languages
   Python can also be integrated through different programming languages such as C, C++, Java, and more. This is one of the advantages of having interpreters

8. Dynamic Memory Allocation
   Specifying data types is no longer needed when coding in Python since it automatically allocates the memory to the variable at run time. This means that a variable's data type is decided at run-time, not in advance.

Paradigms

Most modern programming languages have a philosophy for multi-paradigm programming since different users have different needs. Developers must be able to meet their demands now that innovations are constantly improving.

Python is considered a multi-paradigm programming language. Here are some paradigms that Python supports.

Object-Oriented Paradigm

The key feature of object-oriented programming is the "object" styled way of thinking. An object is termed as the instance of a class that consists of data members and methods. This programming paradigm supports classes, encapsulation, abstraction, and inheritance that will make coding more efficient.

Procedural Paradigm

In procedural programming, a series of computations and procedures in the program are executed in a systematic order of statements, functions, and commands to complete a computational program. This programming paradigm facilitates the practice of good program design, organization due to its container-style programming, and modularization.

Functional Paradigm

Python also supports the functional programming paradigm because it can bind pure mathematical functions and manages every statement as functional expressions and declarations rather than the execution of statements. Recursion and Lambda expression are one of the approaches of functional programming. This programming paradigm focuses more on "what to solve" than "how to solve" like in procedural programming.

Language Evaluation

Readability

Python is fairly easy to read and understand by people with or without much experience in programming using this language and is the easiest to read among the languages compared here. Since its philosophy emphasizes more on simplicity and readability, its use of fewer yet readable constructs is a feature that highlights orthogonality. In addition, Python is the closest to the natural language, English, than other PLs whose syntax design is closest to machine code. Python has well-defined control and extensive data structures since its memory is dynamically allocated so users don't need to define the data type. Although, this lack of explicit data typing may somehow be a disadvantage when code reviewing.

Writability

As Python is easy to read, it also has good writability. Python is very easy to write since it has simple constructs and dynamic memory allocation. Users can still write a simple program to complex ones with efficiency in writing them since it uses fewer constructs. This also means that expressivity exists in Python because there is more work done in a single line of code than a single line of code in other languages. Although, expressivity may sometimes confuse users like using the operator `==`, `===`, and `is`.

Reliability

Python's exception handling and aliasing features are quite reliable. In aliasing, variables in Python are just containers that store references to values. Though there is no type checking, the intuitive syntax as well as the extensive exception handling capabilities and runtime debugging greatly make up for this.

Octave

Purpose and Motivation

GNU Octave is a high-level programming language that consists of a powerful mathematics-oriented syntax that supports 2D and 3D plotting and visualization tools. It provides a command-line interface for solving linear and nonlinear problems numerically, and for performing other numerical-based problems using a language compatible with MATLAB (GNU).

It provides extensive tools for solving mathematical problems such as linear algebra problems, finding the roots of nonlinear equations, integrating ordinary functions, manipulating polynomials, and integrating ordinary differential and differential-algebraic equations. It can be extended and customized further by creating user-defined functions written in Octave itself, or by using dynamic libraries written in C, C++, Fortran, or other languages.

It is easily extensible and customizable via user-defined functions written in Octave's own language, or using dynamically loaded modules written in C++, C, Fortran, or other languages.

Octave was created to be a companion for a chemical reactor design textbook for a chemical engineering course but throughout its development, it became a general flexible tool that is used widely in fields other than chemical engineering. Nowadays, it is used for teaching, research, and commercial applications.

History

Octave was originally conceived around 1988 to be an accompanying software for a chemical reactor design textbook authored by James Rawlings of the University of the Wisconsin-Madison and John Ekerdt of the University of Texas. It was initially intended to be specialized in solving chemical reactor design problems. However, this initial approach limited the capabilities of the tool being developed, and so the creators attempted to build a more flexible tool.

During this time, Fortran was still very popular which is why people still opted for it as it is the computer language of engineering. However, the creators found that every time they tried to utilize Fortran, their students spent too much time debugging their Fortran code and not enough time was allotted to learning about chemical engineering. The creators believed that in a more interactive environment, such as the case for Octave, most students would be able to grasp the basics easily and use the tool with ease in just a few hours.

In the Spring of 1992, the creators began the full-time development of Octave. The first alpha version of Octave was released on January 4, 1993. More than a year later, version 1.0 was released on February 17, 1994. Octave has undergone several major revisions. It is also included in several GNU/Linux distributions.

When talking about Octave, people typically think about a musical note. However, the GNU Octave is from the name of the creator's former professor, Prof. Octave Levenspiel. Professor Octave wrote a popular textbook on chemical reaction engineering and is known for its ability to do quick back-of-the-envelope calculations.

Development of Octave

The latest stable release (as of writing) of Octave is version 6.4 It is worth noting that the creators somehow followed an unusual numbering scheme. For example, the first version after the alpha release was version 6.0 rather than the usual version 1.0. On this end, discussed below are the major revisions only. Major features not discussed below were introduced to Octave in iteration through minor revisions which are not included below but can found here.

Version 1.1.0

Octave 1.1.0 was released on January 12, 1995. It has major updates when compared to the update to version 1.0 from 0.x.x.

1. `Data structure` type (`struct` type)

Added a new fundamental type in Octave which allows variables to act similar to C's struct construct. This allows variables to hold sub-elements, e.g., `someVar.someUnderlyingAttribute = 124`.

2. Exception Handling
   Octave now supports a limited form of exception handling modeled after the unwind-protect form of Lisp. This is useful to protect temporary changes to global variables from possible errors.

3. Short-circuit evaluation
   The `&&` and `||` logical operators are now evaluated in a short-circuit fashion and work differently than the element by element operators `&` and `|`.

4. Variable return list
   Octave can now return a list of values with unknown cardinality from functions. This means that it is no longer necessary to place an upper bound on the number of outputs that a function can produce.

5. New functions and other improvements
   New functions for operating on polynomials have been added such as evaluation of polynomials at a point, deriving polynomials, integrating polynomials and others. Set manipulation and image processing functions have also been added.

Version 2.0

Octave 2.0 was released on December 10, 1996.

1. Crash handling
   If Octave crashes, it now attempts to save all user-defined variables in a file named `octave-core` in the current directory before exiting. This is very useful for processing intensive computations in which results take time to be computed.

2. Ported to Windows
   Octave has been mostly ported to Windows NT and Windows 95 using the beta 17 release of the Cygnus GNU-WIN32 tools. While not all features work, it is already usable.

3. User-defined data types
   A user can now add custom data-types by writing a class in C++. On systems that support dynamic linking, new data types can be added to an already running Octave binary.

4. Improved exception handling
   A real exception-handling has been implemented using the try-catch method. In this case, the `catch` block is only implemented if there is an error caught in the `try` block.

5. More functions and other improvements
   Several changes have been made to be compatible with newer versions of MATLAB. New functions have also been added for audio processing, file management, and others.

Version 3.8

Octave 3.8 was released on December 27, 2013.

1. Graphical User Interface
   While the new graphical user interface is not yet the default option when running Octave interactively, it is already working. As one of the most requested features throughout the years, it is one of the biggest updates in this version.

2. Nested functions
   Octave now supports functions defined with other functions. The rules and scope of the nested functions are consistent with those used in Matlab.

3. Named Exceptions
   Octave now has limited support for named exceptions. This is very useful for catching unique errors as well as easier debugging.

4. TEX Parser
   A TEX parser has been implemented for the FLTK toolkit and is the default for any text object including titles and axis labels. Commands such `\bf`, `\it`, `\rm`, etc. are now supported.

5. Citation command
   This command returns the citation of Octave and used packages that the user can use in their papers and publications.

6. More functions and other improvements
   Graphics processor has been changed to improved plotting and other visualization tools. New functions have also been added while outdated functions are deprecated or revised.

Version 4.4

Octave 4.4 was released on April 30, 2018.

1. Graphical Variable Editor
   It uses a spreadsheet-like interface for quick, intuitive editing of variables. This allows the user to easily edit variable values as well as view their real-time.

2. Hash data structure
   A map data structure that allows efficient searching values using a key. This allows the user to efficiently store sets of data where the order is not important.

3. More functions and other improvements
   External dependencies have been adjusted to accommodate newer versions of these external tools.

Version 5.1

Octave 5.1 was released on February 23, 2019.

1. High-Resolution support
   The Octave plotting system now supports high-resolution screens, i.e., those with greater than 96 DPI. This is useful for sharper details in visualization tools.

2. Unicode support
   Support for Unicode characters in the names of files and folders accessed externally in Windows.

3. More functions and other improvements
   External dependencies have been adjusted to accommodate newer versions of these external tools.

Version 6.1

Octave 6.1 was released on November 26, 2020.

1. Complex web service support
   Complex RESTful web services can now be accessed by the webread and webwrite functions. One major feature is the support for cookies to enable RESTful communication with the web service. This is really useful as the web continues to be the primary platform nowadays.

2. More functions and other improvements
   External dependencies have been adjusted to accommodate newer versions of these external tools.

Features

Octave is an interpreted, free source, high-level programming language. From its conception, Octave was intended for numerical and math-related applications.

Hence, it has several notable features that allow it to be used in teaching, research, and even commercial applications.

1. Easy-to-use
   Octave was created to allow students to learn it easily and focus more on problems being solved than the language being used. Though Octave is not beginner-friendly for those without programming experience, it is much easier to use in solving tasks for its problem domain when comparing it to its alternatives.

2. Open-source software
   Octave was created to be accessible. For the problem domain of numerical computations, MATLAB is a really popular tool. However, it has the downside of being expensive for the ordinary individual. Octave addresses this difficulty by being a freely distributable software under the GNU General Public License (GPL).

3. User-defined functions
   Octave allows the user to create their own named and unnamed functions. Naturally, named functions can be called when the user has a need for them. This ability greatly extends the capacity of Octave as it allows the user to create more constructs using the basic language-native constructs.

4. Dynamic data typing
   Octave does not require the user to indicate the type of data before storing the data. Octave dynamically determines the appropriate type of data based on the data itself. This has advantages such as added user flexibility as well as disadvantages such as run-time type checking.

5. Native support for matrices and their operations
   Matrix is a fundamental type in Octave. As a fundamental type, it has operations supportive natively as a basic construct of the language. This is really beneficial as it greatly eases the use of matrices. Matrices are used in all branches of math and engineering as well as most of the science where there is a need for numerical or geometric computations.

6. Complex Number support
   Octave also has native support for complex numbers. This allows the user to compute complex operations using complex numbers without the need and hassle of importing libraries. This also saves the user from learning a separate library aside from learning the language.

7. Powerful built-in math functions
   Aside from its native support for matrices and complex numbers, Octave has powerful built-in math functions such as polynomial solvers, differentiation operators, and integral operators.

8. Strong cross-language compatibility
   Octave is written using C, C++, and Fortran which is why Octave code can easily be integrated to run on programs made using these languages. Aside from this, Octave was also built with MATLAB compatibility in mind given that MATLAB is the most popular tool used for the same problem domain.

Paradigms

Like most modern programming languages, Octave is multi-paradigm in nature. While it is multi-paradigm, it is important to note that Octave was meant to solve numerical problems. Thus, its multi-paradigm nature is just to allow the user to tackle a problem using multiple approaches. Some of the paradigms that can be used in Octave are the following.

Procedure-Oriented Paradigm

As a structured language, Octave is often used to solve problems using a sequence of computation steps. Octave has also looping and branching constructs that allow the user to alter the procedure as necessary. As procedural in nature, Octave natively supports the calling of subroutines from imported packages.

Object-Oriented Paradigm

Octave supports the creation of classes which is why it can be used in an object-oriented paradigm. Aside from its capacity to create objects, it also supports inheritance, which is a very important feature in object-oriented programming.

Functional Paradigm

As Octave allows the creation of anonymous functions and the use of these functions as data, then naturally, Octave can be used with a functional paradigm. This allows Octave to use functions as they are used mathematically.

Language Evaluation

Readability

Since Octave is intended for a field-specific problem domain, numerical computations, in this case, it is fairly difficult to read for someone without a background in programming. However, when one has experience in programming, especially for numerical computation, and in languages such R or MATLAB, then Octave

won't be difficult to read. As indices are referenced using parentheses, it is often confusing whether one is referring to matrices or a list and not to a user-defined (user-named) function. Another thing worth noting in terms of readability is that Octave supports numerous mathematical functions at its core, thus, there are many basic constructs that need to be understood when reading Octave code, which is quite problematic given the diversity of math problems Octave is used for.

Writability

In terms of writability, Octave is easy since a lot of the most used constructs such as matrices and plotting are native Octave and so users do not have to learn external libraries. Octave also has simple ways of creating complex equations such as those involving matrices, and this greatly helps it in terms of writability. Octave also has straightforward constructs for minute tasks such as editing axis labels, graph legends, and others which is a point towards its ability to solve the task without much hassle in writing.

Reliability

As Octave runs through an interpreter, it is easy to debug. Its exception handling is also good which increases the reliability of the code developed. Though Octave has no type-checking, it is not a critical downside considering the octave was not made for general-purpose programming. Octave also has good methods to address errors involving undefined values such as empty data in matrices

Existent Features in Python, but not in Octave

Feature 1: Coroutines

Coroutines are defined as "cooperative functions". It generalizes procedures for non-preemptive multitasking, by allowing execution to be suspended and resumed. This means that control from a subroutine can be suspended and resumed to another subroutine. In Python, there are two ways to achieve this: (1) using the `yield` statement and (2) using Python's library `asyncio`. Octave on the other hand does not support asynchronous programming, therefore, coroutines have not been created yet in their latest version.

Asynchronous programming is an important concept in computer science because it solves the problem of long-running tasks and functions in the main

thread. Through coroutines, memory is saved as the items are produced when required, unlike subroutines where memory is destroyed once its execution is finished. It also has a pipeline structure that allows numerous instructions executed at once, since it can be resumed or suspended depending on the programmer.

In Octave, the lack of this feature allows users to easily use simple functions in its system, and not risk its readability, writability, and reliability. Although Octave's main objective is for vectorizable numerical computation, and not much on creating complex and general programs like Python.

```
def fibonacci():
    a, b = 0, 1
    # continues to loop as long as value is returned
    while True:
        # yield suspends the execution and stores value of a and b
        yield b
        a, b = b , b + a

coroutine = fibonacci()

# The next( ) method, executes the code written up to first yield
# statement and then returns the value generated.

print(next(coroutine))  # Prints 1
print(next(coroutine))  # Prints 1
print(next(coroutine))  # Prints 2
```

In this simple coroutine, programmers no longer need to put a limit on the number of loops it must do in the coroutine `fibonacci()` since it will only loop once the `next(coroutine)` is called and the loop gets suspended in the `yield` statement, passing the control back to the next line of the caller.

Another method of doing coroutines in Python is by using `asyncio`.

```
import asyncio
import time

async def main():
    task1 = asyncio.create_task(
        say_after(1, 'Python'))

    task2 = asyncio.create_task(
        say_after(2, 'Coroutines'))

    print(f"started at {time.strftime('%X')}")

    # Wait until both tasks are completed (should take
    # around 2 seconds.)
    await task1
    await task2

    print(f"finished at {time.strftime('%X')}")
```

Here is the expected output from the code above.

```
started at 01:19:20
Python
Coroutines
finished at 01:19:22
```

The use of `asyncio.create_task` will make the coroutine run concurrently as you can see from the time stamp. When using `asyncio`, await is used instead of `yield` when awaiting on a coroutine.

Octave does not support asynchronous programming yet, therefore, providing a code using the programming language might be impossible. In the long run, coroutines might be added in the next version of Octave.

Feature 2: List Comprehension

List Comprehension is a syntactic construct that provides a concise way to create lists. Simply put, it offers a shorter and simpler syntax in creating a list with values based on another list. While this type of programming task is often done using `for` loops, it is very common programming tasks such new ways to accomplish it efficiently have been created. An example would be the task of squaring a list of numbers.

```
numbers = [1,2,3,4]
squared = []
for number in numbers:
    squared.append(number*number)
print(squared)
```

The output of the code above when run is `[1, 4, 9, 16]`.

One function that accomplishes the same task is by the use of `map` functions. Map functions are not present as basic constructs in a lot of programming languages. In these languages, importing a library or module is needed to be able to use the map function which defeats the purpose of efficient task-solving. Another problematic aspect of using the map function is in languages that do not allow anonymous functions and so require the creation of named functions in order for them to be used in the mapping. This is inefficient and adds unnecessary overhead for function definitions. An example of the code utilizing a named function can be seen below.

```
def square(num):
    return num*num

numbers = [1,2,3,4]
squared = list(map(square, numbers))
print(squared)
```

The output of the code above when run is `[1, 4, 9, 16]`. A similar example but using `lambda` functions (anonymous functions).

```
numbers = [1,2,3,4]
squared = list(map(lambda x: x*x, numbers))
print(squared)
```

The output of the code above is the same when run: `[1, 4, 9, 16]`.

The List Comprehension function in Python is present in Python 2 and Python 3. This feature relies critically on the implementation of the `list` data structure of python. It involves at least a single for loop. It often takes the form of a $<$ *listName* $>$ `=[` $<$ *item* $>$ `for` $<$ *iterable* $>$ `in` $<$ *listName* $><$ *conditional* $>$ `]`. A simple example for the task above would be the following.

```
numbers = [1,2,3,4]
squared = [num*num for num in numbers]
print(squared)
```

The output of the code above when run is `[1, 4, 9, 16]`. A similar task but involving a conditional is getting the squares only if the square is an even number. This can be done by the following.

```
numbers = [1,2,3,4]
squared = [num*num for num in numbers if (num*num)%2==0]
print(squared)
```

The output of the code above when run is `[4, 16]`.

While the given examples above are relatively simple, list comprehension can be used for more complex tasks such as nested for loops, nested conditionals, lambda expressions, and utilizing other data structures as elements of the list.

This feature is very useful in reducing the amount of code especially for repetitive tasks, which is often the case in programming. This list comprehension syntax allows us to create lists quickly. It saves from having to create an empty list at first, iterate through the original list, and process the elements of the original list accordingly. Aside from improving writability, it also improves readability because the user does not have to seek multiple lines of code to understand the lists.

The disadvantage of having this feature is that this would hamper its readability for users not familiar with this syntax. Thus, it is another basic construct to learn when learning the language. Though the similar functionality can be done using constructs common in most languages such as the use of functions and importing of libraries and modules. If the tasks being accomplished are too complex to be put into one line of list comprehension, then the traditional methods of looping and function calling are preferred.

While Octave does not have its native list comprehension construct, it can be implemented using the `arrayfun` function. This is not much of a hassle since Octave allows for the use of anonymous functions. Achieving the same task above would be done by the following

```
numbers = [1,2,3,4];
squared = arrayfun(@(x) x*x, numbers)
```

This function of Octave is created for various types of data structure such as vectors, matrices, and polynomials which makes it best suited for the problem domain Octave is aimed towards. The `arrayfun` function is an improved version of the now-deprecated `map` function in Octave.

Conclusion

In summary, when comparing the two, or any programming languages, it is important to consider the problem domain for which the language was originally intended. Considering that Python was made primarily to be beginner-friendly and general in purpose so that it can be used for a variety of tasks without the need of having much programming experience, it fulfills its purpose excellently. On the other hand, Octave was made for beginners for the problem domain of numerical computations and scientific applications. It is a really good language for the said problem domain. Judging Octave on its merits in other tasks like creating a game or mining data on the web would be unfair since it was not created for those. Judging Python for the same tasks may not be as unfair given the Python is being developed by its community for all sorts of tasks. With regards to the numeric computation problem domain, Python is excellent because the libraries developed for it are excellent. Python's glory on this end relies on the extensive support given by its community.

In conclusion, when handling day-to-day typical tasks in which no complex computations are needed, Python would be the preferred language. When complex computations and statistical graphics are needed to be done on a day-to-day basis, then going with Octave would be the better choice.

References

Arora, S. (2021). Why learn python? reasons and benefits of learning python. Retrieved January 22, 2022, from https://www.simplilearn.com/why-learn-python-a-guide-to-unlock-your-python-career-article

Eaton, J. (2022). Octave: About and history. Retrieved January 24, 2022, from https://www.gnu.org/software/octave/about

Geeks for Geeks. (2020a). Programming paradigms in python. Retrieved January 22, 2022, from https://www.geeksforgeeks.org/programming-paradigms-in-python/

Geeks for Geeks. (2020b). Python features. Retrieved January 22, 2022, from https://www.geeksforgeeks.org/python-features/

Klein, B. (2022). History and philosophy of python. Retrieved January 22, 2022, from https://python-course.eu/python-tutorial/history-and-philosophy-of-python.php

Mr. STEM EDU TV. (2021, July 29). *Octave tutorial for absolute beginners: Learn octave in 1 hr and 30 min*. Retrieved January 31, 2022, from https://www.youtube.com/watch?v=TqwSlEsbObg

Murphy, M. (1997). Octave: A free, high-level language for mathematics. Retrieved January 24, 2022, from https://www.linuxjournal.com/article/1225

Pedamkar, P. (2020). Matlab vs octave. Retrieved January 24, 2022, from https://www.educba.com/matlab-vs-octave/

Python Software Foundation. (n.d.-a). Coroutines and tasks. Retrieved February 1, 2022, from https://docs.python.org/3/library/asyncio-task.html

Python Software Foundation. (n.d.-b). Python 3 documentation: Data structures. Retrieved January 24, 2022, from https://docs.python.org/3/tutorial/datastructures.html

Sebesta, R. W. (2016). *Concepts of programming languages* (S. Mukherjee & A. K. Bhattacharjee, Eds.; Eleventh edition, global edition). Pearson.

Thelin, R. (2021). Python version history: How python has changed over the years. Retrieved January 26, 2022, from https://www.educative.io/blog/python-versions-history

Tiobe Index. (2022). Tiobe index for january 2022. Retrieved January 24, 2022, from https://www.tiobe.com/tiobe-index/