Appraise The Key Features In Standard Programming Language Python

Samyuktha Avusunapalli

Assistant Professor, Department of Computer Science, Nishitha Degree College (Autonomous)

Affiliated to Telangana University, Nizamabad.

Accredited with "A" Grade by NAAC

I. Abstract

Writing programs is a vastly innovative and rewarding activity. You can write programs for many reasons. In this Competitive world, we have come up with many programming languages. A distinct original programming language are actually settled. While compared to existing programming languages, Python programming languages has gone through different flourish techniques. It can be used to improve almost any type of application with right tools and libraries.

Python tender us never-ending chances for making small and wide-ranging software projects. Agreeable with easy-to-read code, Python is a famous top programming language to grasp and no advanced programming knowledge is required. Python mastery spread out you up to careers in close to any industry and are required if you desire to continue to other advanced, higher paying software development and engineering roles for example software engineer, systems administrator, and etc. In this Article we go through all the keywords in Python Programming language.

Keywords: Terminologies, Specifications, operations.

II. Introduction

Python is a high –level interpreted basic programming language. It was designed by Guido Van Rossum and it come out in the year 1991. Python is used for server to design web applications, to hold big data and make complex mathematics, can link to database systems. It can be able to read as well as modify files. Python has very easy syntax, which we can write and can read like English Languages. Python allows user to write extremely fewer lines then compared to other programming languages. Python runs on an interpreter system, this means that code can be executed as soon as it is write down. This means that prototyping can be speedy.

Python is convenient for writing server-side code in view of that it offers huge libraries that contain of pre-authored code for complex backend functions. Designer also use an extensive range of Python frameworks that gives all the necessary tools to design web applications quickly and more comfortably. For example, developers can design the any web application in a fewer seconds because they don't need to write it from scratch. They can then examine it with the framework's testing tools, without depending upon any other testing tools.

III. Keywords

Python has a set of reserved words, called keywords, which have special meaning and are used to define the structure of the language. These keywords cannot be used as identifiers (like variable names or function names) in Python. They are used to define the syntax and functionality of Python's constructs.

- Python has 33 Keywords
- Every Keyword has to be in Lowercase except 3 keywords those are False, None, True.
- There are 35(Three Five) Python keywords as Python 3.8., i.e., Async & await are the other two added
- Every reserved word has a unique motive.
- A programmer should not use the Python keywords for any additional purpose rather than the predefined use.

- A principle point to be noted is that programmers are not needed to be imported. They are always accessible to the programmers.
- Let us learn what the keywords are and where, why, how we use them.
- For a complete and updated list of keywords specific to your version of Python, you can use the keyword module:

Syntax:

```
import keyword
print(keyword.kwlist)
```

- Succeeding table is an index of keywords in the Python.

False	**class**	**finally**	**is**	**return**
True	**continue**	**for**	**lambda**	**try**
None	**def**	**from**	**nonlocal**	**while**
and	**del**	**global**	**not**	**with**
as	**elif**	**if**	**or**	**yield**
assert	**else**	**import**	**pass**	
break	**except**	**in**	**raise**	

IV. A detailed description of every Keyword and Their Usage.

False Keyword:

In Python, the keyword False is a built-in constant that represents the Boolean value "false". It is part of Python's core syntax and is used to indicate a logical false condition.

In Python, False is equivalent to 0 when used in numerical contexts, and it is considered falsy when evaluated in a Boolean context (like if statements). The opposite of False is True.

None Keyword:

In Python, the None keyword is an exceptional constant that acts for the absence of a value or a null value. It is often used to indicate that a variable does not have any value assigned to it or that a function does not return anything.

Here are some key points about None:

1. None as a placeholder: It can be used as a placeholder to represent the absence of a value.
2. Function return value: A function that does not explicitly return a value will return None by default.

3. Comparison: You should use the is operator to compare a variable to None, rather than the == operator, since None is a singleton in Python.

None is not the same as False, 0, or an empty string or list. They are all considered falsy, but they are distinct values.

It's important to remember that None represents the absence of a value and is commonly used in situations where you need to explicitly state that no value exists.

True Keyword:

In Python, True is a built-in constant that indicates the Boolean value "true". It is part of Python's core syntax and is one of the two Boolean values in Python, the other being False.

The bool data type has two possible values: True and False. They are often used in conditions and control structures to represent truth values in logical operations.

In summary, True is a constant in Python representing a logical truth, and it is essential for conditions, logical expressions, and comparisons.

and keyword:

In Python, the and keyword is a logical operator that is used to combine two or more conditions in a way that both conditions must be True for the overall expression to evaluate to True.

- The and operator returns True only if both condition (or operands) it combines are True.
- If either of the conditions statements are False, the whole expression results to False.
- It is commonly used in conditional expressions or if statements.
- The and operator is used to check if multiple conditions are True.

as keyword:

In Python, the as keyword is used in several contexts, primarily for creating aliases and for handling exceptions in a more readable and manageable way. Below are the main use cases for the as keyword in Python.

The as keyword is commonly used in import statements to give a module, class, or function an alias, making it easier to reference throughout your code.

assert keyword:

In Python, the assert keyword can be used for debugging motive. It tests whether a given condition is true, and if the condition is False, it raises an AssertionError exception with an optional error message. It's a way to ensure that certain conditions hold true during runtime, typically used during development and testing.

Condition: This is the expression that should evaluate to True for the program to continue. If the condition is False, an AssertionError will be occured.

"Error message" (optional): If the assertion fails, this message is displayed as part of the error.

• He assert keyword is used to verify that conditions are True during the development phase.

• If the condition statement is False, an AssertionError is occurred with an optional error message.

• It helps catch bugs early but is typically disabled in production environments.

• Assertions are not meant for handling runtime errors like invalid user input; they are for catching programming errors.

break keyword:

In Python, the break keyword is used to exit from a loop prematurely. It allows you to terminate a for loop or a while loop before the loop has iterated over all the items or before the condition of a while loop becomes False.

- break exits the current loop immediately.
- It is used in for and while loops.
- It is often used when a certain condition is met, and you want to stop further iterations.

The break statement is helpful for optimizing code when a loop no longer needs to continue based on a certain condition, saving unnecessary iterations.

class keyword:

In Python, the class keyword is used to define a new class, which is a blueprint for creating objects (instances). A class defines the properties (attributes) and behaviors (methods) that the objects created from the class will have.

- The class keyword is used to define new classes in Python.
- The __init__ method is the constructor that initializes the object's state.
- Attributes are variables that belong to an object, and methods are functions that define the behavior of an object.
- Classes can inherit from other classes, allowing you to extend and reuse code.

Using classes in Python helps in organizing code, managing complexity, and creating reusable components through object-oriented programming (OOP).

continue keyword:

In Python, the continue keyword is used inside loops (either for or while loops) to skip the rest of the code inside the current iteration and proceed to the next iteration of the loop. It effectively skips over the current loop iteration and continues with the next one.

- The continue keyword statement is only used inside loops.
- It does not terminate the loop, it just skips to the next iteration.
- It is useful when you want to skip certain iterations based on a condition without exiting the loop entirely.
- It helps improve code readability and logic flow by skipping unnecessary parts of a loop.

del keyword:

The del keyword in Python is used to delete objects, variables, list elements, or dictionary items. It's a way to remove references to objects or data in memory, effectively "deleting" them.

Deleting an object using del only removes the reference to it, and the object itself is garbage collected if no other references to it exist.

def keyword:

In Python, the def keyword is used to define a function. It marks the beginning of a function definition, followed by the function name, optional parameters, and the body of the function. Functions in Python allow you to encapsulate code that can be reused multiple times throughout a program.

The def keyword is essential for defining functions in Python, which allow you to structure your code in a modular and reusable way. Functions help in reducing redundancy and improve code clarity and maintainability.

elif keyword:

The elif keyword in Python is used to check multiple conditions in an if-else statement. It stands for "else if" and allows you to specify additional conditions to test if the initial if condition evaluates to False. If an if condition is False, the elif block (if present) will be checked next. If none of the if or elif conditions are True, the else block (if present) will be executed.

The elif keyword provides a way to check multiple conditions in an if-else chain, making it useful for handling complex decision-making scenarios. It allows you to avoid deeply nested if statements and write cleaner, more readable code.

else keyword:

The else keyword in Python is used to define a block of code that will execute if none of the preceding conditions in an if-elif chain are True. It provides a default block of code that will run when all the other conditions fail. The else block is optional but often used for completeness in conditional statements.

- else is used to execute a block of code when none of the preceding if or elif conditions are true.
- It is commonly used in conditional statements (if-elif-else) for handling default actions when specific conditions are not met.
- The else block can also be used with loops, executing when the loop completes normally without a break.

except keyword:

The except keyword in Python is used in conjunction with the try block for exception handling. It defines a block of code that will execute if an error (exception) occurs in the try block. The except block allows you to handle specific types of errors and avoid crashing the program when an exception is raised.

- try: Defines the block of code to be tested for exceptions.
- except: Handles the exception if one occurs in the try block.
- else: Optionally runs if no exception is raised in the try block.
- finally: Optionally runs after the try block (and after any except or else), regardless of whether an exception occurred.

The except block is an essential part of Python's exception handling mechanism, helping you write more robust and error-resilient programs by anticipating potential issues and handling them gracefully.

finally keyword:

The finally keyword in Python is used in exception handling to define a block of code that will always be executed, no matter if an exception was raised or not. It is typically used for cleanup actions, such as closing files, releasing resources, or performing other necessary operations that should run regardless of whether the code block completed successfully or failed due to an exception

- The finally block is useful for ensuring that certain operations (like cleanup, closing resources, or releasing connections) are always performed, regardless of whether an exception occurred.
- It always executes after the try block, even if an exception was raised or the program returned from the try block.
- It is especially useful when dealing with resource management, like file handling or database connections, to guarantee that resources are properly released.

from keyword:

- The from keyword in Python is used in import statements to import specific attributes, classes, functions, or variables from a module. This allows you to import only the parts you need from a module, rather than importing the entire module.
- The from keyword helps you import specific components from a module, which is efficient and prevents unnecessary imports of unused parts of a module.
- It makes your code more readable and concise, as you can directly reference imported items without needing to reference the module's name each time.

for keyword:

The for keyword in Python is used to implement for loops, which allow you to iterate over a sequence (such as a list, tuple, string, or range) and execute a block of code for each item in that order.

- for loops are used for iterating over a sequence (such as a list, tuple, string, or range).
- You can iterate over different types of sequences like lists, strings, and dictionaries.
- The range() function is commonly used to generate sequences of numbers for iteration.
- You can use else with a for loop, and the else block runs only if the loop completes without a break.
- Nested for loops allow you to iterate over multi-dimensional structures

global keyword:

The global keyword in Python is used to declare that a variable is global and should be accessed from the global scope, rather than being treated as a local variable within a function. It allows you to modify the value of a global variable from within a function.

Key Points about the global Keyword:

- Global Scope: Variables that are defined outside of functions are in the global scope and are accessible anywhere in the program, including inside functions.
- Local Scope: Variables defined inside functions are in the local scope, and by default, you cannot modify global variables directly within a function without using the global keyword.

if keyword:

The if keyword in Python is used to implement conditional statements. The if statement is a fundamental control structure in Python, enabling you to introduce decision-making in your program.

The if keyword is an essential part of Python's flow control, allowing you to execute different blocks of code based on conditions. You can use if, elif, and else to handle various decision-making scenarios and make your program more dynamic.

import keyword:

The import keyword in Python is used to bring modules, classes, functions, or variables from external files or libraries into the current script. This allows you to reuse code and organize your programs into multiple files or leverage pre-existing libraries without rewriting the code.

- The import keyword is essential for using external code in Python.
- It helps you reuse code, access standard libraries, and organize your programs.
- You can import entire modules, specific functions or classes, or use aliases for convenience.
- Be mindful of potential naming conflicts and avoid importing everything from a module with * unless absolutely necessary.

in keyword:

The in keyword in Python is used in several contexts, including for checking membership in a sequence, iterating over items in loops, and in comparisons. It plays a crucial role in making code more expressive and concise.

The in keyword is a powerful tool in Python for checking membership, iterating through sequences, and performing concise conditional checks. It improves code readability and reduces the need for manual looping or comparisons. By using in, you can write more expressive and Pythonic code.

is keyword:

The is keyword in Python is used to test identity. It checks whether two variables point to the same object in memory, i.e., whether they are the same object, not just if they have the same value. This is different from equality testing, which checks whether the values of two variables are equal. For value comparisons, you would use the == operator instead.

- Identity comparison: is checks if two references point to the same object in memory.
- Equality comparison: == checks if the values of two objects are the same.

lambda keyword:

The lambda keyword in Python is used to create anonymous functions, also known as lambda functions. These are small, single-expression functions that are typically used for short-term tasks where a full function definition might be overkill.

- They are anonymous, meaning they don't require a name.
- They are limited to a single expression. Unlike regular functions, they can't contain multiple statements or commands.
- Lambda functions can contain number of arguments, but only one expression.

nonlocal keyword:

The nonlocal keyword in Python is used to work with variables in nested functions. It allows you to modify the value of a variable that is declared in the enclosing (but non-global) scope, meaning in a scope that is not the local or global one.

- Accessing Variables in Enclosing Scope: The nonlocal keyword allows a function to modify a variable in its enclosing scope, which is the scope that is not the local scope, but also not the global scope.
- Avoiding Global Scope: It is specifically used to avoid working with variables in the global scope.
- Used in Nested Functions: It is commonly used when you have nested functions (functions within functions) and want to modify a variable in the enclosing function's scope.

not keyword:

The not keyword in Python is a logical operator that is used to negate or reverse the truth value of a condition or expression. It is commonly used in conditional statements to check if a condition is false.

- Negation: It inverts the truth value of an expression. If the expression evaluates to True, not will make it False, and if the expression evaluates to False, not will make it True.
- Boolean Context: It is often used in if statements or other logical conditions to check for the negation of a condition.

or keyword:

The or keyword in Python is a logical operator used to combine multiple conditions. It returns True if at least one of the conditions evaluates to True. If all conditions are False, then it returns False.

- Logical Disjunction: or is used to combine two or more conditions. The expression evaluates to True if at least one of the conditions is True.
- Short-circuiting: Python uses short-circuit evaluation with or, meaning if the first condition is True, Python will not evaluate the second condition because the result is already determined (since True or anything is always True).
- Boolean Context: or is commonly used in conditional expressions to test multiple conditions or in situations where you want to provide fallback values.

pass keyword:

The pass keyword in Python is a null operation or a placeholder. It is used when you need to write a statement syntactically but don't want it to perform any action. Essentially, it allows you to create an empty code block where code is syntactically required but you don’t want to execute any code.

- Empty Block: It is commonly used in places where a statement is syntactically required but you don’t want to implement anything yet.
- No Operation: The pass statement does nothing — it is a no-op (no operation).
- Placeholders: It is often used as a placeholder for future code or during development to avoid syntax errors while leaving parts of the code unfinished.

raise keyword:

The raise keyword in Python is used to raise exceptions explicitly. It allows you to trigger an exception manually, which can be helpful for error handling or for ensuring that certain conditions are met in your program.

- Triggering Exceptions: raise is used to generate an exception explicitly in your code, either by raising a predefined exception or by creating your own custom exception.
- Error Propagation: When an exception is raised, Python stops executing the current code block and starts looking for an exception handler (typically an except block) to handle the exception.
- Custom Exceptions: You can define and raise your own exceptions by subclassing the base Exception class or any of its derived classes.

return keyword:

The return keyword in Python is used to exit a function and return a value from that function to the caller. When a return statement is executed, the function terminates immediately, and the value specified in the return statement is sent back to the caller.

Key Characteristics of return:

- Exits the Function: As soon as return is encountered, the function stops execution and returns the specified value (if any) to the caller.
- Returns a Value: You can return a value from the function, such as a number, string, list, or even another function.
- Optional: A function doesn't need to have a return statement. If there is no return, the function will return None by default when execution reaches the end.
- Multiple return Statements: A function can have multiple return statements, but only the first one encountered during function execution will be executed. Once a return is encountered, the function exits.

try keyword:

The try keyword in Python is used to start a block of code that will be monitored for exceptions. When using try, Python will attempt to execute the code inside the try block. If an exception occurs, it will jump to the corresponding except block (if provided) to handle the error. This allows you to write error-handling code that doesn't stop the entire program when an exception occurs.

The try block is typically paired with except, and optionally with else and finally blocks to form a complete exception-handling structure.

while keyword:

The while keyword in Python is used to create a while loop, which repeatedly executes a block of code as long as a given condition is True. The loop will stop executing as soon as the condition becomes False.

- The while loop in Python is used for repeated execution of a block of code as long as a specified condition remains True.
- It is versatile and can be used for tasks that require indefinite iteration, such as waiting for user input, processing items in a collection, or waiting for a specific condition to be met.
- Careful attention is needed to avoid infinite loops, and control statements like break and continue can be used to manage the flow of the loop.

with keyword:

The with keyword in Python is used to wrap the execution of a block of code within methods defined by a context manager. The with statement simplifies exception handling and ensures that resources are properly managed, especially when dealing with tasks such as file handling, database connections, or locking mechanisms.

The primary advantage of using the with keyword is that it automatically handles the setup and cleanup actions associated with a context, ensuring that resources (such as files, network connections, or locks) are properly acquired and released, even if an error occurs.

t is most commonly used with context managers, which are objects that define the __enter__() and __exit__() methods.

yield keyword:

The yield keyword in Python is used to create a generator function. A generator is a special type of iterator that allows you to produce a sequence of values lazily (i.e., on demand) rather than computing them all at once and storing them in memory. This can be especially useful when dealing with large datasets or streams of data where it is inefficient to load everything into memory at once.

When a function contains the yield keyword, it becomes a generator function, which doesn't return a value in the conventional sense. Instead, it yields a value each time it is called, and it can be resumed later from where it left off.

- The yield keyword is used to create generators, which produce values lazily, one at a time, as needed.
- This makes generators memory-efficient for working with large data sets or infinite sequences.
- Generators are a powerful tool for implementing iterators and working with sequences of data in an efficient and readable way.

Special Note on async and await:

async and await were introduced in Python 3.5 as part of the asynchronous programming features. They allow for writing asynchronous code that runs concurrently, making it easier to work with I/O-bound tasks (like web requests or reading from files) without blocking the execution of the program.

V. Conclusion

In conclusion, Python keywords are fundamental building blocks of the language, each serving a specific purpose in defining the structure and behavior of a Python program. These reserved words cannot be used as identifiers (such as variable or function names) and are essential for controlling program flow, defining data structures, managing exceptions, and implementing various functionalities in Python. Understanding these keywords is crucial for writing efficient and error-free code. Since Python's keywords are defined by the language itself, it is important to stay updated with any changes in newer Python versions to ensure compatibility and proper usage.

VI. References

1. PYTHON FOR EVERYBODY Exploring Data Using Python 3 by Charles R. Severance.
2. From geeksforgeeks.
3. From tutorialpoints.
4. From Real Python tutorials.