**Interpreter Vs Compiler : Differences Between Interpreter and Compiler**

In this article, you will learn the differences between interpreters and compilers.

We generally write a computer program using a high-level language. A high-level language is one that is understandable by us, humans. This is called **source code**.

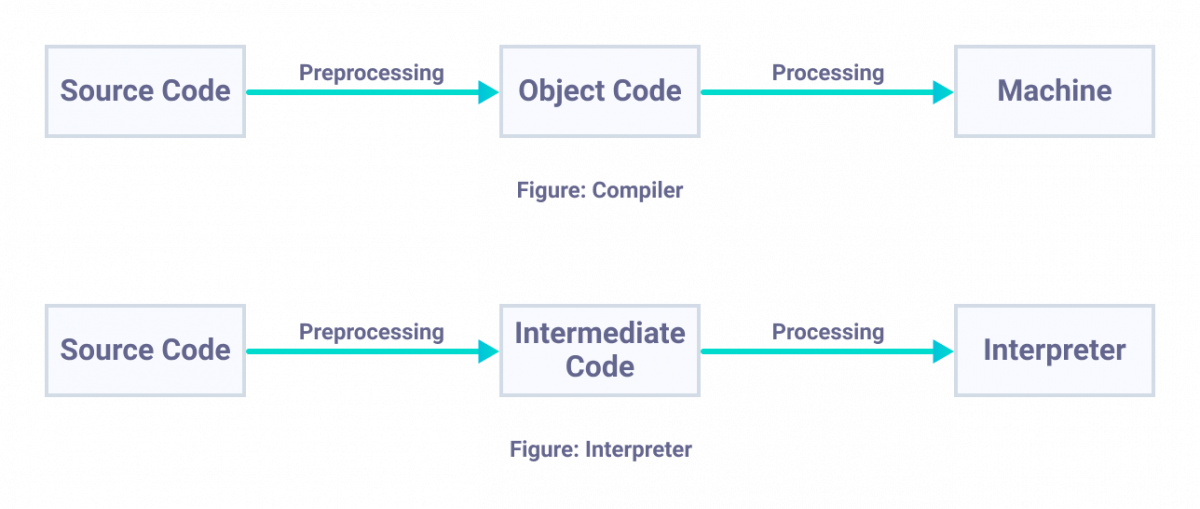
However, a computer does not understand high-level language. It only understands the program written in **0**'s and **1**'s in binary, called the **machine code**.

To convert source code into machine code, we use either a **compiler** or an **interpreter**.

Both compilers and interpreters are used to convert a program written in a high-level language into machine code understood by computers. However, there are differences between how an interpreter and a compiler works.

**Interpreter Vs Compiler**

|  |  |
| --- | --- |
| Interpreter | Compiler |
| Translates program one statement at a time. | Scans the entire program and translates it as a whole into machine code. |
| Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers. | Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters. |
| No Object Code is generated, hence are memory efficient. | Generates Object Code which further requires linking, hence requires more memory. |
| Programming languages like JavaScript, Python, Ruby use interpreters. | Programming languages like C, C++, Java use compilers. |



| **Comparison Parameter** | **Python 2** | **Python 3** |
| --- | --- | --- |
| Year of Release | Python 2 was released in the year 2000. | Python 3 was released in the year 2008. |
| “Print” Keyword | In Python 2, print is considered to be a statement and not a function. | In Python 3, print is considered to be a function and not a statement. |
| Storage of Strings | In Python 2, strings are stored as ASCII by default. | In Python 3, strings are stored as UNICODE by default. |
| Division of Integers | On the division of two integers, we get an integral value in Python 2. For instance, 7/2 yields 3 in Python 2. | On the division of two integers, we get a floating-point value in Python 3. For instance, 7/2 yields 3.5 in Python 3. |
| Exceptions | In Python 2, exceptions are enclosed in notations. | In Python 3, exceptions are enclosed in parentheses. |
| Variable leakage | The values of global variables do change in Python 2 if they are used inside a for-loop. | The value of variables never changes in Python 3. |
| Iteration | In Python 2, the xrange() function has been defined for iterations. | In Python 3, the new Range() function was introduced to perform iterations. |
| Ease of Syntax | Python 2 has more complicated syntax than Python 3. | Python 3 has an easier syntax compared to Python 2. |
| Libraries | A lot of libraries of Python 2 are not forward compatible. | A lot of libraries are created in Python 3 to be strictly used with Python 3. |
| Usage in today’s times | Python 2 is no longer in use since 2020. | Python 3 is more popular than Python 2 and is still in use in today’s times. |
| Backward compatibility | Python 2 codes can be ported to Python 3 with a lot of effort. | Python 3 is not backward compatible with Python 2. |
| Application | Python 2 was mostly used to become a DevOps Engineer. It is no longer in use after 2020. | Python 3 is used in a lot of fields like Software Engineering, Data Science, etc. |

**Python 2 vs. Python 3 Example Code**

Now that we have a good understanding of the differences between Python 2 and Python 3, let us take a look at a sample code in both these versions to print a statement out:

**Python 2**

def main():

print "Hi! This is Python 2"

if \_\_name\_\_== "\_\_main\_\_":

main()

**Python 3**

def main():

print ("Hi! This is Python 3")

if \_\_name\_\_== "\_\_main\_\_":

main()

variables and creation rules

3. **Variable creation rules (programs for this one)**

Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)

**#Legal variable names:**  
myvar = "John"  
my\_var = "John"  
\_my\_var = "John"  
myVar = "John"  
MYVAR = "John"  
myvar2 = "John"  
  
**#Illegal variable names:**  
2myvar = "John"  
my-var = "John"  
my var = "John"

1. **Explain about 10 different keywords**

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Description** | **Example** |
| break | A break is a [loop control statement](https://www.educba.com/control-statement-in-c-plus-plus/). It helps to control the execution of the loops. Specifically, the break is responsible for terminating the execution of the loop. | for i in range(1, 11):        # print the value of i      print(i)        # check the value of i is less then 5      # if i lessthen 5 then continue loop      if i < 5:          continue        # if i greater then 5 then break loop      else:          break |
| continue | Continue is a [loop control statement](https://www.educba.com/control-statements-in-c/). It helps to control the execution of the loops. Specifically, Continue is responsible for switching the loop control to the condition statement again. |
| def | The def keyword is used for defining a function or method in [python programming](https://www.educba.com/python-programming-beginners-tutorial/). The function is a block of code that can be executed. | # define GFG() function using def keyword  def GFG():      i=20      # check i is odd or not      # using if and else keyword      if(i % 2 == 0):          print("given number is even")      else:          print("given number is odd")    # call GFG() function  GFG() |
| if | The if keyword represents a condition instance in python. |
| else | The else keyword is used to represent the false execution of an[if statement](https://www.educba.com/if-statement-in-python/). |
| true | This keyword represents the Boolean value ‘true’. | check\_string = '123' print(check\_string.isdigit()) |
| false | This keyword represents the Boolean value ‘false’. | check\_string = 'asd' print(check\_string.isdigit()) |
| not | The word ‘not’ is also reserved for Boolean or logical operations. When implying this keyword, it means that an operation will be applied when the given conditional expression is not satisfied.   Popular Course in this category | check\_value = 5 if check\_value not in [1,7,4,6]: print("Hello World!\n") else: print("Nothing to print") |
| and | The word ‘and is reserved for Boolean or logical operations. When implying this keyword, it means that an operation will be applied only when both conditions stand true. | check\_value = 5 if check\_value > 1 and check\_value < 10: print("Hello World!\n") else: print("Nothing to print") |
| or | The word ‘or’ is also reserved for Boolean or logical operations. When implying this keyword, it means that an operation will be applied even when one of the conditions stands true. | check\_value = 5 if check\_value > 1 or check\_value < 10: print("Hello World!\n") else: print("Nothing to print") |