Click to prove you're human

The base language of R provides a number of operators, such as %*% (matrix multiplication), %/% (integer division), and %in% (is lhs a component of rhs?). However, %% represents modulo. The expm package defines a matrix power operator %^%. R has also defined a |> pipe, which can substitute into the first argument of the right-hand side. This new pipe is included in base-R as of R v4.1.0 and is advocated by the Tidyverse for most use cases.The use of Byte Order Marks (BOM) in text files can lead to compatibility issues and problems with certain software. To avoid these issues, it's recommended to change the first lookahead from (?=.{8,}) to (?=[A-Za-z@#$%^&+=]{8,}$). This ensures that only characters like "at" or "@" are present in the regular expression.The BOM is composed of two ASCII characters and should not be included before the "#!" characters in a script file. This can lead to problems with interpreters and misinterpretation of the encoding. Some authorities recommend against using the BOM in POSIX scripts, as it's unnecessary and can cause interoperability issues.In JSON files, the BOM is illegal and can break parsers that rely on specific byte sequences to determine the encoding. The correct way to identify the character encoding in JSON is through more reliable methods, such as examining the first four bytes for a NUL byte.The use of BOMs can also lead to issues with other data formats, such as JSON, and should be avoided unless absolutely necessary. In general, it's recommended to avoid using BOMs as an optional feature, especially in cases where the encoding is already determined by other means.import threading threading Lock() used to synchronize access to shared resources threading thread.start_new_thread() function is used to start a new thread. However, if you want to be able to import the thread module with the new version, then you should still use if __name__ == "__main__": as it will allow both versions of the script to run. But for most cases, this is better off in the main method because it has no purpose if your code doesnt need any functionality when it's imported. A thread can be started from another thread but not vice-versa; you would have an error otherwise. For example: threading._active contains a list of currently active threads. When youre done with the thread, just remove your reference to it and Python will delete the thread if there are no more references to it.In order to utilize the functions in another program without rewriting it, it's recommended to import them and include a check for the __name__ variable at the end of the file. When a module is imported, all its code is executed from start to finish. However, when the condition is met, the func, func2, etc., which corresponds to the Wikipedia __scrape__, are not executed.In the global scope, a Python __name__ is defined as '__main__' for the current program. When a module is imported, it becomes a variable in the namespace of our current program and the current program's __name__ is set to '__main__'. This can be seen by printing 'The program name is set to ' followed by the value of globals()['__name__'].Here's an example code snippet that demonstrates this:```pythondef func(): passdef func2(): passprint('The program name is set to ', globals()['__name__'])if __name__=='__main__': print('inside of current program') print('name is current program', __name__) test1.func(' ') test1.func2(' ')```In this case, the output will be: `inside of current program name is current program test1`.When running a module directly (e.g., `python test.py`), the __name__ is set to '__main__', and the code inside the if statement is executed.However, when importing the module in another program (e.g., `import test1`), the __name__ remains as 'test1', which means that the func and func2 are not executed.To understand process exit status, we need to first comprehend the concept of process exit status defined by POSIX. In Linux, when a process calls the exit system call, the kernel stores the value passed to the system call (an int) even after the process dies.The `exit` ANSI C function and indirectly return from main both call the exit system call. The process that called the exiting child process can retrieve the exit status of the child with the wait system call.For instance, in Bash: `$ false` outputs 1, and when we run the command `$ echo $?`, it also outputs 1, indicating that the exit status of the previous command is 1.In C, using `exit(1)` will set the exit status to 1. We can compile and run a program like this:```bashg++ -ggdb3 -O0 -std=c++11 -Wall -Wextra -pedantic -o bash bash.c./bash```Output: `$? = 1`Similarly, in Bash, when we hit enter, a fork + exec + wait happens, and Bash then sets $? to the exit status of the forked process. Additionally, in Bash, `?` expands to the decimal exit status of the most recent pipeline (see Pipelines). The shell treats several parameters specially, including `?`, which may only be referenced; assignment to them is not allowed.For example, we can use the following code snippet:```bashfalseecho $?```Output: `1`This is equivalent to using the C "equivalent" code:```c#include /* exit */int main(void) { exit(1);}```Or in Bash:```bash#!/bin/bashif [1 = 1]; then :fi```In this case, `?` Expands to the decimal exit status of the most recently executed foreground pipeline. ANSI C and POSIX recommend that 0 means the program was successful, while other values indicate a failure.The exact value could indicate the type of failure. ANSI C does not define the meaning of any values, and POSIX specifies values larger than 125.In Bash, we often use the exit status $? implicitly to control if statements as in: `if true; then : fi`, where true is a program that just returns 0.The above is equivalent to: true result=$? if [$result = 0]; then : fi And in: if [1 = 1]; then : fi [is just an program with a weird name (and Bash built-in that behaves like it), and 1 = 1] its arguments, see also: Difference between single and double square brackets in BashThis is a representation of Unicode Character 'LINE FEED (LF)' (U+000A) is the HTML character entity's way of saying: give me the Unicode character at hexadecimal codepoint 0xA And because hex 0xA is the same as decimal 10, here's another way of getting the same character: Related: $ echo -n ' ' | recode html..ascii | xxd 00000000: 0a . $ ascii 0a ASCII 0/10 is decimal 010, hex 0a, octal 012, bits 00001010: called ^J, LF, NL Official name: Line Feed Other names: Newline, Further reading: Bash script to convert from HTML entities to characters ###ARTICLEKnowing that the problem may already be obsolete>_