```
LEVEL3 - Cybersecurity
                                              Matthias Naumann - 42 Heilbronn
pyjail - small
                                                       github.com/MaNafromSaar
gist:
Break a pyjail that has very limited input size and only "print" as
built-in.
steps:
1. run script
2. test limitations
3. try and research commands and flags
4. another approach - redirection
5. proof of concept and further ideas
1. In this case you just need to navigate to the folder of the challenge
and run it via:
      python chall.py
2. Now I tried different inputs and yes, this jail is very narrow and 42
chars is not much to do something meaningful.
The first inputs that yielded some info was:
      >>> print(print. module )
      builtins
and.
      >>> print(__builtins__.__dir__)
      <built-in method __dir__ of dict object at 0x7ff1f8c38500>
since we know it is a dict, I tried:
      >>> print(__builtins__.keys())
      dict_keys(['print'])
and:
      >>> print( builtins .values())
      dict_values([<built-in function print>])
      >>> print(print. _class__)
      <class 'builtin function or method'>
which rather feels like running in circles.
      >>> print(().__class__._bases__[0].__name__)
      object
      >>> print(().__class__._mro__)
      (<class 'tuple'>, <class 'object'>)
led me to another path, which I later abandoned. As there is so little to
leverage, what candidates are left?
```

- print
- eval

- AssertionError
- throw

And then I had the idea that may be considered as a solve: redirection!

Alas, redirection from within the jail would have been too easy:

```
>>> print("J") > flag.txt
J
```

didn't create the file but threw an error that the name was not defined. But what if I already set up the redirection to file when calling up the jail?

python chall.py > flag.txt print("ctf{brev}")

did the trick, the file was created.

To demonstrate that this is a **proper RCE**, I tested the same with an input as text which is way too large for the jailed prompt.

python chall.py > large_test.txt print("Mein Vati hat drei grunzende Schweine")

Should yield the assertion error and **it did!** The file was created but only contained the arrowheads that were missing from the prompt when calling the jailed script this way.

Further ideas:

While I could not really read out the flag from within the jail, I thought about other places than a file to redirect to and environment variables came to my mind.

export MY_FLAG_CONTENT=\$(python chall.py) print("ctf{brev}")

Again I could verify the functionality with **printenv** and then tested for large input, to see if we are actually in the jailed shell. As you can see in the picture, the **arrowheads** from the prompt format are mistakenly added to the printed output

```
MY_FLAG_CONTENT=>>> ctf{brev}
LESS_TERMCAP_ue=
SHLVL=1
LESS_TERMCAP_us=
```

an undoubtable sign, that this has run via the jail. The large input:

export LARGE_CONTENT=\$(python chall.py) print("eins schwarz, eins braun, eins hat zu kurze Beine")

threw the **AssertionError** again and the **env** entry only contained the arrowheads from the forlorn prompt.

```
LESS_TERMCAP_us=
LARGE_CONTENT=>>>
```

Quod erat demonstrandum.