1.3 Python Keywords and allowed Variable names

In [1]:

```
# To retrieve the python keyword list, we can use the keyword built-in package.
import keyword
```

Let's print the keywords present.

keyword.kwlist returns python's keywords in a list datatype.

We are using *(starred) expression to print the values returned by keyword.kwlist each separated by "\n"(newline).

In [2]:

```
print(*keyword.kwlist, sep="\n")
```

```
False
None
True
__peg_parser__
and
as
assert
async
await
break
class
continue
def
del
elif
else
except
finally
for
from
global
if
import
in
is
lambda
nonlocal
not
or
pass
raise
return
try
while
with
yield
```

Variable Names

TLDR:

- Variable names shouldn't be same as that of built-in keywords.
- Variable name shouldn't start with a number or with a symbol(except "_", protected and private attributes are created using underscore, 🤔 it's better to say it as name mangling rather than protected or private. That's for a different notebook session 😀).

PS: Don't give a try naming the variable that starts with #, it would be a Python's comment, which would be neglected by the interpreter 😅.

Allowed Variable names

In [3]:

```
x = True
_x = False
x_y = "Hey Python geek!"
x9 = "alphabet_number"
# Python is a case sensitive language, so `x` is different from `X`. Let's give it a try.
X = "one more variable"
print(f"x is equal to X:{x==X}")
```

```
x is equal to X:False
```

Invalid Variable names

We will be using `exec` within `try`-`except` to catch the syntax error. 🤔 But why? Syntax errors can't be catched, well it shouldn't for good 😉. so we are using `exec` to execute the code.

`exec` takes the string argument and interprets the string as a python code.

In [4]:

```
# variable name starting with number.
code_string = "9x=True"
try:
    exec(code_string)
except SyntaxError as exc:
    print(f"Ouch! In the exception: {exc}")
```

```
Ouch! In the exception: invalid syntax (<string>, line 1)
```

In [5]:

```
# variable name starting with a symbol(other than underscore"_").
code_string = "$g = 10"
try:
    exec(code_string)
except SyntaxError as exc:
    print(f"Ouch! In the exception: {exc}")
```

```
Ouch! In the exception: invalid syntax (<string>, line 1)
```