In [ ]:

```
#VARIABLES: It is a reserved memory location to store values for reuse
#You do not have to declare variables before using them in Python

#Right hand side = expression or value
#Left hand side = Variable
#Value stored from Right to Left
year = 2020
print(year)
```

In [ ]:

```
#Redeclare Variables

year = 2021
print(year)
year = 2022
print(year)
```

In [ ]:

```
# VARIABLE NAMING CONVENTIONS

# ● can have letters (A-Z and a-z), digits(0-9), and underscores (_).
# ● should maintain snake_casing.
#   That means each word should be separated by underscores(_).
# ● cannot begin with digits
# ● cannot have whitespace and special signs (e.g.: +, -, !, @, $, #, %)
# ● are case sensitive meaning ABC, Abc, abc are three different variables.
# ● have to be meaningful with the data stored in it.
# ● cannot be too lengthy and too general, need to have a balance
# ● should not be written with single alphabets. Minimum length should be 3.
# ● cannot be a reserved keyword for Python. There are a total of 35 keywords.

```

In [2]:

```
#meaningful, short, start with lowercase, minimum length 3
carColor = 'red'
print(carColor)

#snake casing
car_name = 'Toyota'
print(car_name)

#camel casing
carName = 'Toyota'
print(carName)
```

```
red
Toyota
Toyota
```

In [3]:

```
#no white space
car name = 'Toyota'
print(car name)
```

```
  File "/var/folders/b8/q3zmxcts1wv9d7xr2_lrs4hr0000gn/T/ipykernel_276
5/1689239340.py", line 2
    car name = 'Toyota'
        ^
SyntaxError: invalid syntax
```

In [4]:

```
#no special characters
car+name = 'Toyota'
print(car+name)
#+carname = 'Toyota'
#print(+carname)
```

```
  File "/var/folders/b8/q3zmxcts1wv9d7xr2_lrs4hr0000gn/T/ipykernel_276
5/2835762722.py", line 2
    car+name = 'Toyota'
    ^
SyntaxError: cannot assign to operator
```

In [ ]:

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

Do not name your variable using any word from this list, it will give an error.
```

In [1]:

```
#keyword list words can't be used as variable names
continue = 2
print(continue)
```

```
  File "/var/folders/b8/q3zmxcts1wv9d7xr2_lrs4hr0000gn/T/ipykernel_77
4/2026781708.py", line 2
    continue = 2
             ^
SyntaxError: invalid syntax
```

In [6]:

```
#DATATYPE

#NUMERIC/NUMBER:

#Integer (int):
#Positive or negative whole numbers (without a fractional part).

num = 1
print(num)
print(type(num))

#Floating-point (float):
#Any real numbers with “decimal” points or floating-point representation.

num2 = 2.5
print(num2)
print(type(num2))

#Random
print(0x10)
#0x10 is a hexa decimal that gets converted to int when we print
```

```
1
<class 'int'>
2.5
<class 'float'>
16
```

In [7]:

```
#BOOLEAN TYPE:
#True and False

num1 = 5
num2 = 10
print(num1>num2)
print(num1<num2)
print(type(num1>num2))
```

```
False
True
<class 'bool'>
```

In [ ]:

```
#SEQUENCE TYPE: ORDERED COLLECTION of similar or different data types.
#String, List and Tuple
```

In [8]:

```
#String(str): A sequence of ordered characters

text = "Good Morning"
print(text)
print(type(text))
```

```
Good Morning
<class 'str'>
```

In [9]:

```
#List: (can be changed)
#It is an ordered collection of elements where the elements
#are separated with a comma (,) and enclosed within square brackets [].
#The list can have elements with more than one data types.

colors = ["black", "white", "grey"]
store = ["black", 123, 1.25]
print(colors)
print(store)
print(type(colors))
print(type(store))

#EXTRA
#You can convert everything you stored inside a list to other datatype
strs = [str(item) for item in store]
print(strs) #You'll see everything in inverted commas

numbers = ["1", 2, 3.0]
#Cast each element of list into int
ints = [int(item) for item in numbers]
print(ints)
```

```
['black', 'white', 'grey']
['black', 123, 1.25]
<class 'list'>
<class 'list'>
['black', '123', '1.25']
[1, 2, 3]
```

In [10]:

```
#Tuple: (can't be changed)
#The elements are separated with a comma (,)
#and enclosed within square brackets []
colors = ("black", "white", "grey")
print(colors)
```

```
('black', 'white', 'grey')
```

In [11]:

```
#MAPPING TYPE:
#Dictionary(dict): key:value pair.
#UNORDERED COLLECTION. Python 3.7 onwards = OREDERED COLLECTION
#Useful to store a lot of info about smth particular
book = {
    "name":"Digital Fortress",
    "author":"Dan Brown",
    "published_yr":1998
}
print(book)
print(book['author'])
print(book["author"])
```

```
{'name': 'Digital Fortress', 'author': 'Dan Brown', 'published_yr': 19
98}
Dan Brown
Dan Brown
```

In [12]:

```
#None Type: a null value or no value at all.
#It's a special data type with a single value, None.
#None type is nothing, Empty string is still a string
print(type(None))

# declaring a variable as None
var = None
print(var)

#None is used to define a null value.
#It is not the same as an empty string, False, or a zero.
#It is a data type of the class NoneType object.

#Assigning a value of None to a variable is one way
#to reset it to its original, empty state.

#You cannot do any operation with None type.
```

```
<class 'NoneType'>
None
```

Data type checking covered above. print() function is used for printing the value of an expression we use it to see output on the screen