**Differences between Interpreter and Compiler**

**Interpreter:**

* Translates program one statement at a time
* Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers.
* No Object Code is generated, hence are memory efficient.
* Programming languages like JavaScript, Python, Ruby use interpreters.

**Compiler:**

* Scans the entire program and translates it as a whole into machine code.
* Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters.
* Generates Object Code which further requires linking, hence requires more memory.
* Programming languages like C, C++, Java use compilers.

**Difference between python 2 and python 3**

**Python 2:**

* Python 2 was released in the year 2000.
* In Python 2, print is considered to be a statement and not a function.
* The values of global variables do change in Python 2 if they are used inside a for-loop
* A lot of libraries of Python 2 are not forward compatible.

**Python 3:**

* Python 3 was released in the year 2008.
* In Python 3, print is considered to be a function and not a statement.
* The value of variables never changes in Python 3.
* A lot of libraries are created in Python 3 to be strictly used with Python 3.

**10 Different Keywords used:**

False 🡪 instance of class bool

Example: x = False

True 🡪 instance of bool class.

Example: x = True

Def 🡪 keyword used to define a function.

Example: def bar(): print(“Hello”)

None 🡪 instance of NoneType object

Example: x = None

If 🡪 The if keyword represents a condition instance in python.

Else 🡪 else keyword is used to represent the false execution of an[if statement](https://www.educba.com/if-statement-in-python/)

Example: if(i %2==0):

        print("given number is even")

    else:

        print("given number is odd")

Break 🡪 It helps to control the execution of the loops. Specifically, the break is responsible for terminating the execution of the loop.

Continue 🡪 . It helps to control the execution of the loops. Specifically, Continue is responsible for switching the loop control to the condition statement again.

Example: if I less then 100 then continue loop

    If i< 100:

        continue

             # if i greater then 100 then break loop

    else:         break

**Variable creation rules:**

* A variable name must start with a letter or the underscore character.
* A variable name cannot start with a number.
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)

Example Program:

Sum of 2 numbers where a,b and c are variables

a=10

b=20

c=a+b

print(c)