C2 Language Overview

1. Expressions and statements
    - Expression: combinations of literals, identifiers and operators which will return a value
    - Statements: line of code
    - semicolon is not required in the Python at the end of the code but there is exception:

```
print("Hello") ; print("World")
```

2. Whitespace and comments
    - whitespace is important in Python because it is used to define the block of cod
    - symbol for comments is #

3. print()
    - 3 different types of print statement by combining the string and numbers

```
#{} is work with .format to combine with the printed string
# %d is a placeholder for x (better) to use format method
# f is stand for .format()

x = 42
print('Hello, World.{}'.format(x))
print('Hello, World.%d' % x)
print(f'Hello, World.{x}')
```

4. Blocks and scope
    - Blocks do not have the scope so that the variable defined inside the block but it still can be call outside the block as they are still in the

same scope

```
if x < y:
    z = 10
    print('x < y: x is {} and y is {}'.format(x, y))

print("z is {}".format(z))
```

5. Conditionals
    - Python only use if else statements for conditionals statement

```
x = 42
y = 73

if x < y:
    print('x < y: x is {} and y is {}'.format(x, y))
elif x > y:
    print('x < y: x is {} and y is {}'.format(x, y))
else:
    print("x and y are the same")
```

6. Loops
    - There are 2 types of loops in Python which are while loop and for loop
    - While loop: will be execute if the condition is true
    - For loop: used for iteration

7. Functions
    - functions also known as methods
    - use def keyword to define a function

```
def function(n):
    print(n)

function(47)
```

8. Object

- object always work with class

```
#self: reference to the object
class Duck:
    sound = "Quack"
    walking = "Walks like a duck."

    def quack(self):
        print(self.sound)

    def walk(self):
        print(self.walking)

def main():
    #donald is an object
    donald = Duck()
    donald.quack()
    donald.walk()

if __name__ == '__main__': main()
```