Language Component & Loop

By Aksadur Rahman

aksadur@yahoo.com

Agenda

Arithmetic Operations
Operator Precedence
Math Functions
Indentation
If Statements
Logical Operators
Comparison/Relational/Conditional Operators
Assignment Operators
Ternary Operators
While Loops
Break & Continue Statement
Sum of n Numbers Program
For Loops
For-While Comparison
For with Range Function
Nested Loops

Arithmetic Operations

Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Arithmetic Operations

Example

```
val1 = 3
val2 = 2

print(val1 + val2)
print(val1 - val2)
print(val1 * val2)
print(val1 / val2)
print(val1 // val2)
print(val1 % val2)
print(val1 ** val2)
```

Operator Precedence

Operators	Meaning
()	Parentheses
**	Exponent
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction

```
print(10+3*2**2+45)
```

Math Functions/Module

Python math Functions

Python Math functions is one of the most used functions in Python Programming. In python there are different built-in math functions. Beside there is also a math module in python.

```
x=2.9
print(round(x))
print(abs(-2.9))
```

Python math Module

Python has a built-in module that you can use for mathematical tasks. The math module has a set of methods and constants.

```
import math

x=2.9
print(math.ceil(x))
print(math.floor(x))
```

https://www.w3schools.com/python/module_math.asp

Python Calendar

```
import calendar

year = 2024

print(calendar.calendar(year))
```

Python Indentation

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

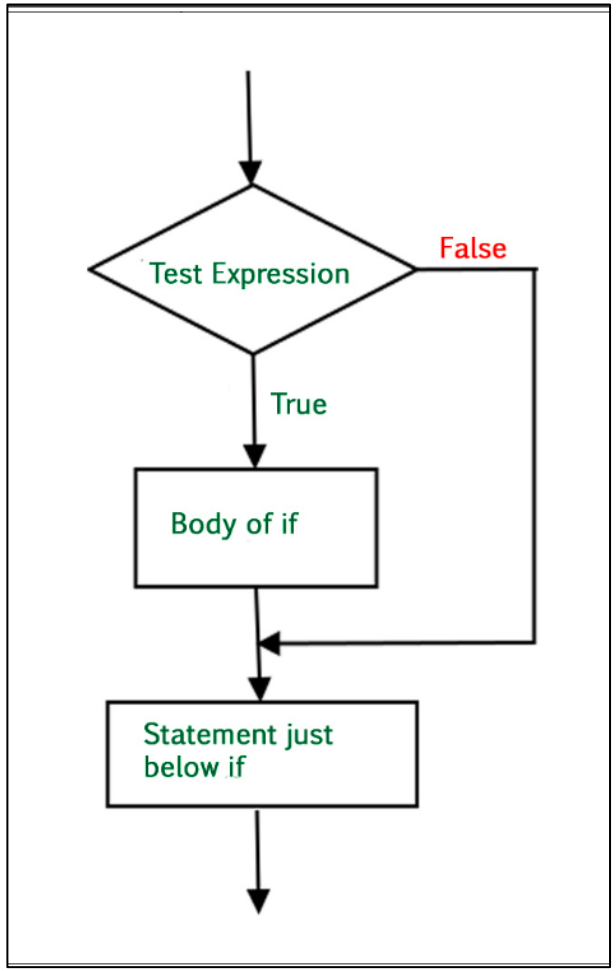
```
if 5 > 2:
    print("Five is greater than two!")
```

If Statements

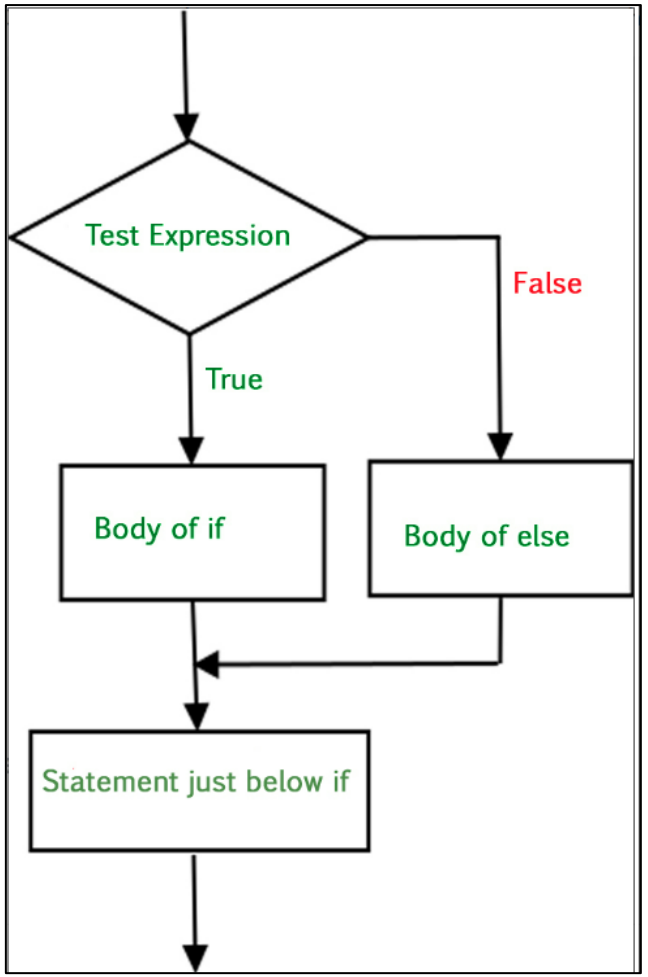
if statement

if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

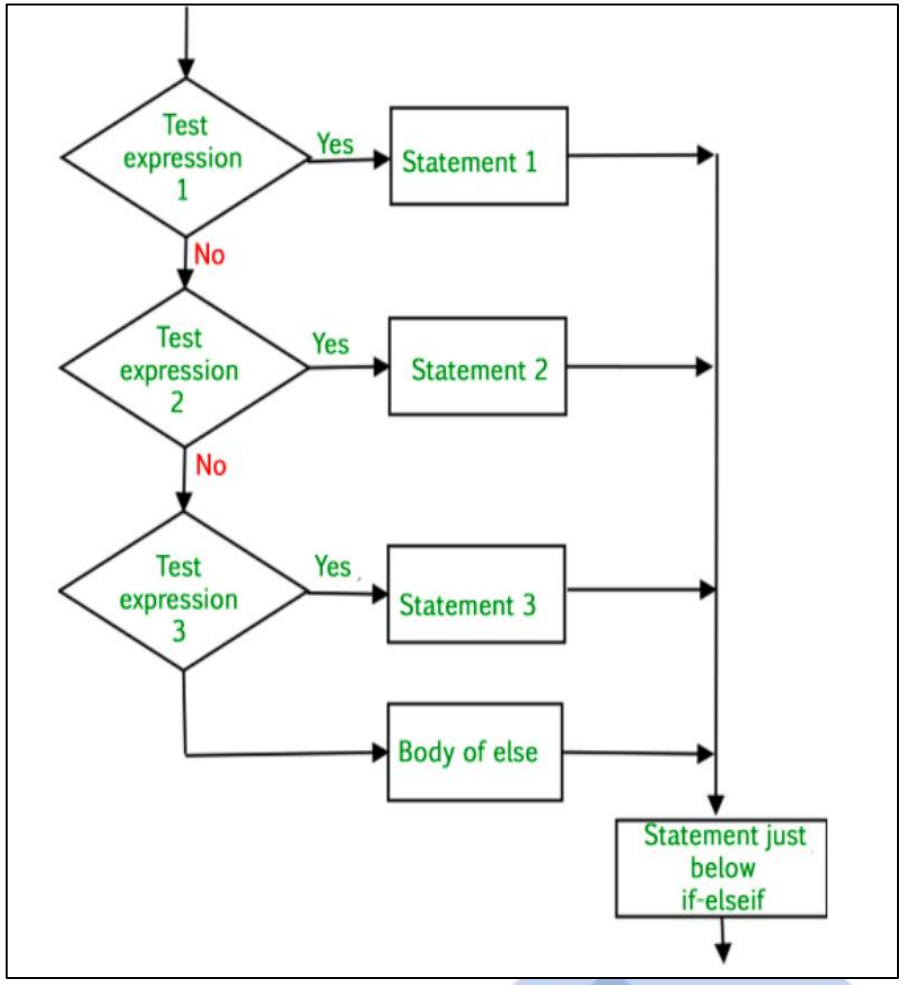
if



If……else



If……elif……else



If Statements

Example

if

```
i = 10

if i > 15:
    print("10 is less than 15")
print("I am Not in if")
```

If……else

```
i = 20
if i < 15:
    print("i is smaller than 15")
    print("i'm in if Block")
else:
    print("i is greater than 15")
    print("i'm in else Block")
print("i'm not in if and not in else Block")
```

If……elif……else

```
i = 20
if i == 10:
    print("i is 10")
elif i == 15:
    print("i is 15")
elif i == 20:
    print("i is 20")
else:
    print("i is not present")
```

Match Case

Example

```
num = 3
match num:
    # pattern 1
    case 1:
        print("One")
    # pattern 2
    case 2:
        print("Two")
    # pattern 3
    case 3:
        print("Three")
    # default pattern
    case _:
        print("Number not between 1 and 3")
```

Comparison/Relational/Conditional Operators

>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to – True if operands are not equal	x != y
>=	Greater than or equal to: True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to: True if left operand is less than or equal to the right	x <= y

Comparison/Relational/Conditional Operators

Example:

```
a = 9
b = 5

print(a > b)
print(a < b)
print(a == b)
print(a != b)
print(a >= b)
print(a <= b)
```

Logical Operators

OPERATOR	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x

and

```
a = 10
b = 10
c = -10

if a > 0 and b > 0:
    print("The numbers are greater than 0")

if a > 0 and b > 0 and c > 0:
    print("The numbers are greater than 0")
else:
    print("Atleast one number is not greater than 0")
```

Logical Operators

or

```
a = 10
b = -10
c = 0

if a > 0 or b > 0:
    print("Either of the number is greater than 0")
else:
    print("No number is greater than 0")

if b > 0 or c > 0:
    print("Either of the number is greater than 0")
else:
    print("No number is greater than 0")
```

not

```
a = 10

if not (a%3 == 0 or a%5 == 0):
    print("10 is not divisible by either 3 or 5")
else:
    print("10 is divisible by either 3 or 5")
```

Assignment Operators

Python Assignment Operators

Assignment operators are used to assign values to variables:

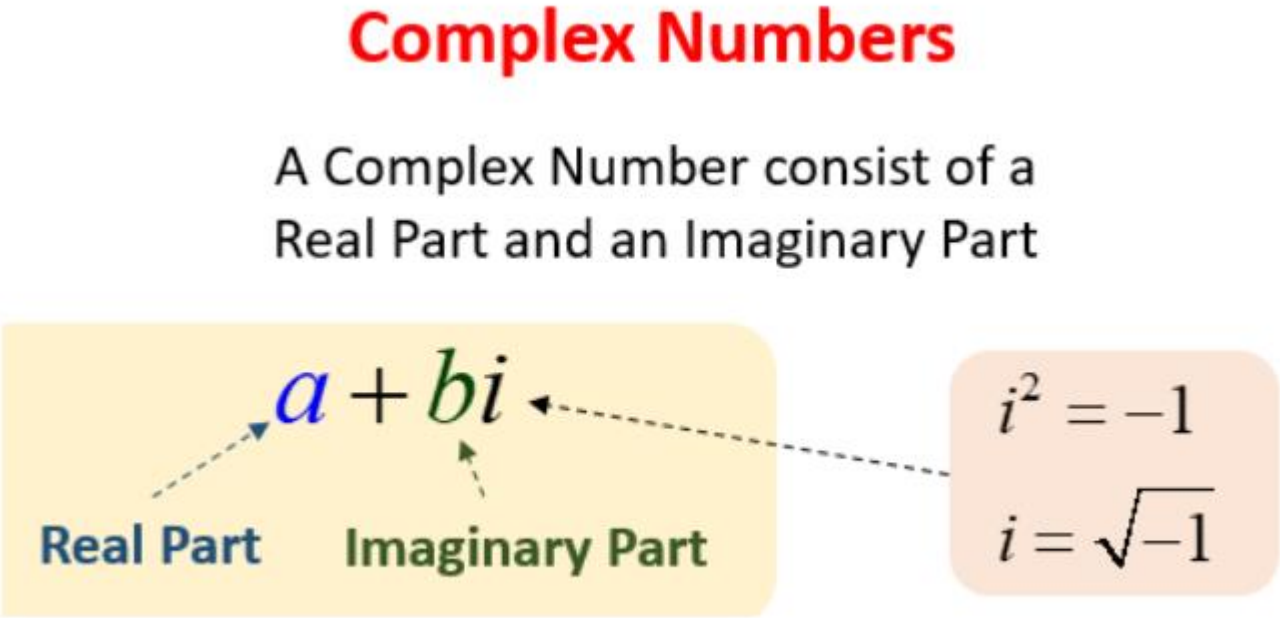
Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Numeric Data Types

Python Number Types: **int, float, complex**

Complex

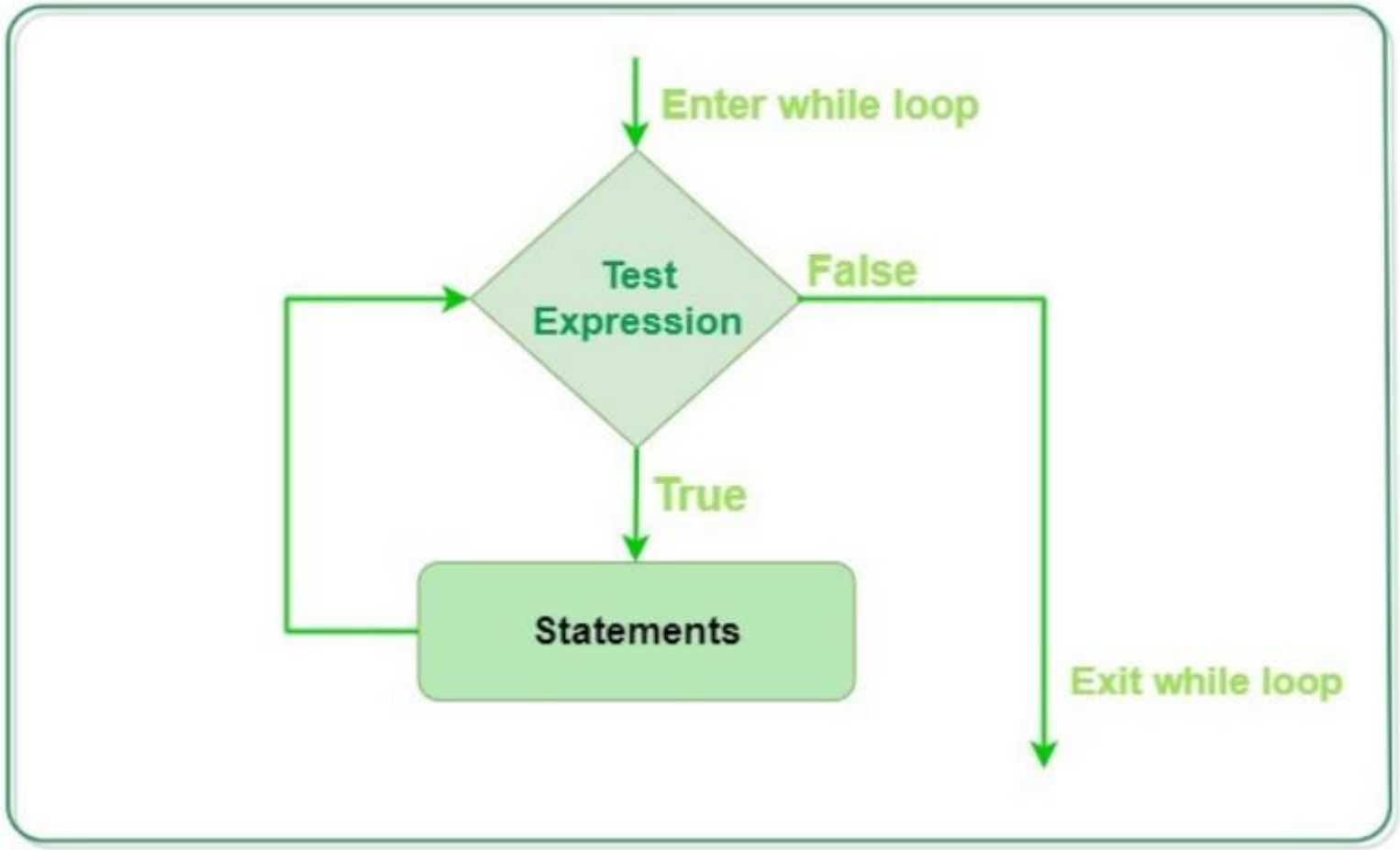
A complex number is a number with real and imaginary components. For example, 5 + 6j is a complex number where 5 is the real component and 6 multiplied by j is an imaginary component.
Complex data types is used while developing scientific applications where complex mathematical operation is required.




```
a=6+7j
print(a)
print(a.real)
print(a.imag)
print(type(a))

b=5+5j
print(a+b)
```

While Loops

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.




```
count = 0
while count < 9:
  print('The count is:', count)
  count = count + 1

print("Good bye!")
```

Sum of n Numbers Program

```
n = int(input("Enter the n Number:"))
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i +1

print(sum)
```

Break & Continue Statement

Break

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Continue

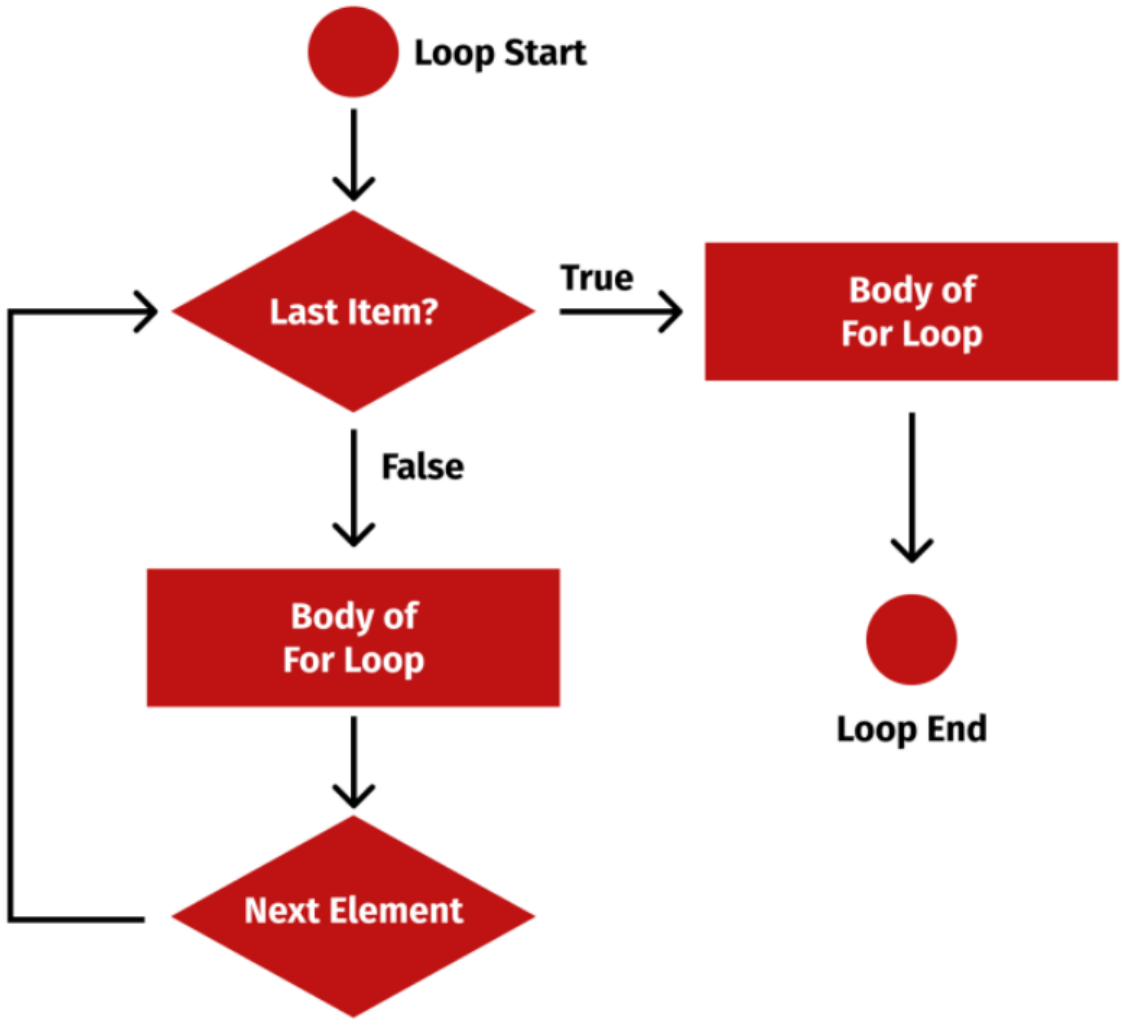
```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
sum = 0

while True:
    num = input("Enter a Number: ")
    if num == "quit":
        break

    try:
        num = int(num)
    except:
        print("Enter a valid number please.")
        continue
    sum = sum + num
    print(sum)
```

For Loops

For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).



For Loops

```
#Looping Through a String

for x in "banana":
  print(x)
```

```
#Looping Through a list

fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
#The break Statement
#Exit the loop when x is "banana"

fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
#The continue Statement
#Do not print banana:

fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

For-While Comparison

while

```
num = [10, 20, 30, 40, 50]
index = 0
n = len(num)
while index < n:
    print(num[index])
    index = index+1
```

for

```
num = [10, 20, 30, 40, 50]
for x in num:
    print(x)
```

For with Range Function

The range() Function

To loop through a set of code a specified number of times, we can use the range() function,

```
#Using the range() function:

for x in range(6):
  print(x)
```

```
#Using the start parameter:

for x in range(2, 6):
  print(x)
```

```
#Increment the sequence with 3 (default is 1):

for x in range(2, 30, 3):
  print(x)
```

For with Enumerate

```
num = [30, 10, 70, 12]

for i, x in enumerate(num):
    print(i, x)
```

Nested Loops

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana",
"cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```