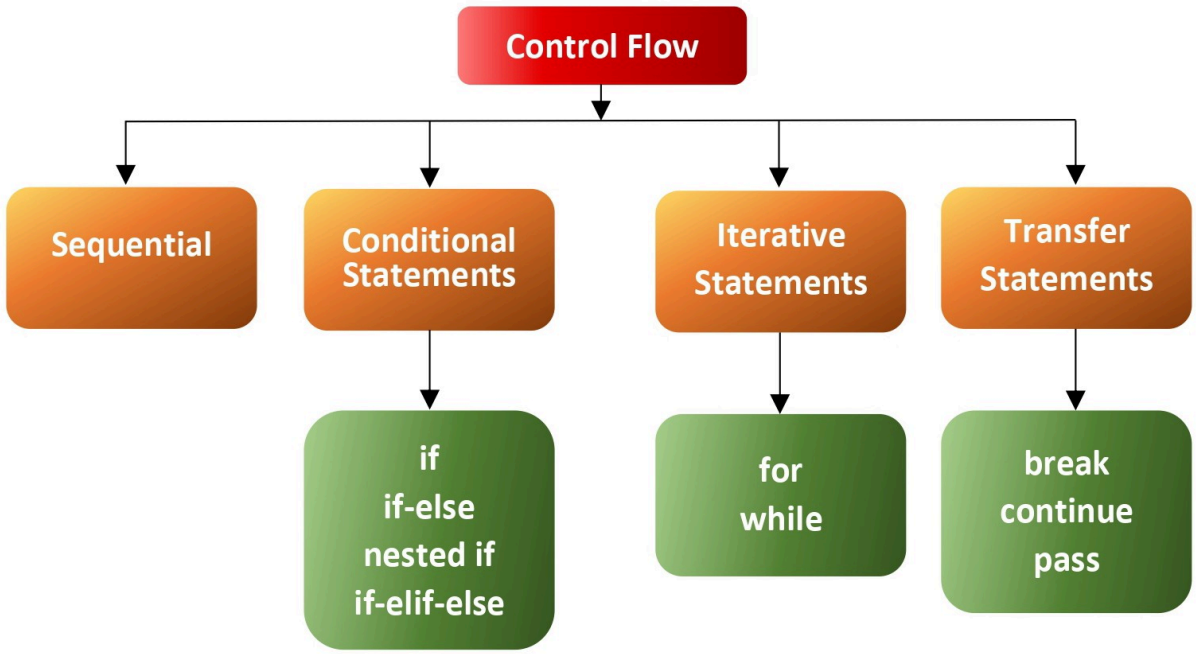
The order in which program's code is executed is called program's control flow.
The control flow of a Python program is regulated by conditional statements, loops, and function calls.
Python has three types of control structures:

- Sequential: Default mode.
- Selection and Decision: Used for Decisions and branching. Example: simple if, if-else, nested if, if-elif-else.
- Repetition: Used for looping, repeating a piece of code multiple times.



Sequential

Sequential statements are a set of statements whose execution process happens in a sequence. The problem with these statements is that if the logic in these statements has broken even if once, then the complete source code execution will break.

```
# This is a Sequential Statement
a=20; b=10; c=a-b
print("Subtraction is:",c)
```

Selection and Decision.

The Selection statement allows a program to test several conditions and execute instructions based on which condition is true.

Some Decision Control Statements are: Simple if, if-else, nested if, if-elif-else

1. Simple if:

A Simple IF statement helps a programmer to run the code only when a specified condition is satisfied. When we have only one condition to be checked then we can use IF statement.

Synatax:

```
if condition:
    # Code to executed if given condition is True
```

Example:

```
n=10

if n % 2 == 0:
    print(f"{n} is an even number")
```

```
OUTPUT:
10 is an even number
```

2. if-else:

This is used in the following situation:

If a condition is true, then execute the given command, else execute the command in the else statement.

For example: “If I finish my homework, my mom will let me play. Else, she will not allow me to play”.

Example 1:

```
n=5

if n % 2 == 0:
    print(f"{n} is an even number")
else:
    print(f"{n} is an odd number")
```

```
OUTPUT:
5 is an odd number
```

Example 2:

```
alpha = input("Enter an alphabet: ")

if alpha=="a" or alpha=="e" or alpha=="i" or alpha=="o" or alpha=="u":
    print(f"{alpha} is an Vowel")
else:
    print(f"{alpha} is an Consonant")
```

```
OUTPUT:
Enter an alphabet: i
i is an Vowel
```

Example 3:

```
age = int(input("Enter your Age: "))

if age>=18 and age<125:
    print("You are eligible to cast your vote")
else:
    print("You are NOT eligible to cast your vote")
```

```
OUTPUT:
Enter your Age: 17
You are NOT eligible to cast your vote
```

Example 4:

```
num = int(input("Enter a Number: "))

if num>0:
    print(f"{num} is Positive")
else:
    print(f"{num} is Negative")
```

```
OUTPUT:
Enter a Number: -4
-4 is Negative
```

3. nested if:

The nested if statement is used when you need to check whether two or conditions are met. We put one 'if' statement inside another 'if' statement.

This is called nesting effect.

Example 1:

```
a = 5
b = 10
c = 15

if a>b:
    if a>c:
        print("a value is big")
    else:
        print("c value is big")
elif b>c:
    print("b value is big")
else:
    print("c value is big")
```

```
OUTPUT:
c value is big
```

Example 2:

```
num1 = int(input("Enter the 1st Number: "))
num2 = int(input("Enter the 2nd Number: "))
num3 = int(input("Enter the 3rd Number: "))

    if num1>num2 and num1>num3:
        print(f"{num1} is the Greatest Number")
    if num2>num1 and num2>num3:
        print(f"{num2} is the Greatest Number")
    if num3>num1 and num3>num2:
        print(f"{num3} is the Greatest Number")
    if num1<num2 and num1<num3:
        print(f"{num1} is the Smallest Number")
    if num2<num1 and num2<num3:
        print(f"{num2} is the Smallest Number")
    if num3<num1 and num3<num2:
        print(f"{num3} is the Smallest Number")
```

3. if-elif-else: The if-elif-else statement is used to conditionally execute a statement or a block of statements.

Example 1:

```
num = int(input("Enter a Number: "))

if num>0:
    print(f"{num} is Positive")
elif num==0:
    print(f"{num} is Positive")
else:
    print(f"{num} is Negative")

OUTPUT:
Enter a Number: 0
0 is Positive
```

Example 2:

```
per = eval(input("Enter your Percentage: "))

if per<=100 and per>=90:
    print(f"With a score of {per}%, you've earned an A+ Grade")
elif per<90 and per>=80:
    print(f"With a score of {per}%, you've earned an A Grade")
elif per<80 and per>=70:
    print(f"With a score of {per}%, you've earned an B Grade")
elif per<70 and per>=50:
    print(f"With a score of {per}%, you've earned an C Grade")
elif per<50 and per>=35:
    print(f"With a score of {per}%, you've earned an D Grade")
else:
    print("You've Failed")

OUTPUT:
Enter your Percentage: 35
With a score of 35%, you've earned a D Grade
```

Example 3:

```
num1 = eval(input("Enter a number: "))
num2 = eval(input("Enter another number: "))

print("----MENU----")
print("1. Addition")
print("2. Subtraction")
print("3. Multiplication")
print("4. Division")

choice = int(input("Enter your Choice: "))

if choice == 1:
    print(f"\nAddition of {num1} and {num2} is {num1+num2}")
elif choice == 2:
    print(f"\nSubtraction of {num1} and {num2} is {num1-num2}")
elif choice == 3:
    print(f"\nMultiplication of {num1} and {num2} is {num1*num2}")
elif choice == 4:
    print(f"\nDivision of {num1} and {num2} is {num1/num2}")
else:
    print("In-valid Choice")
```

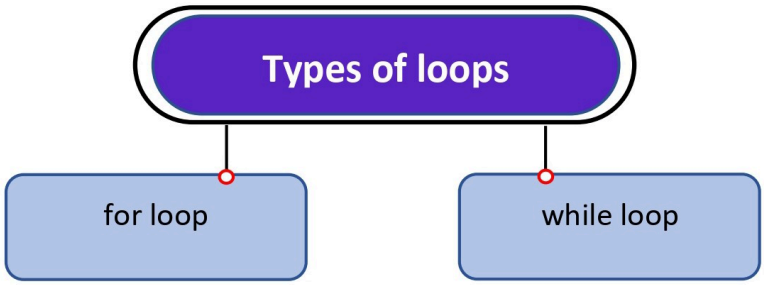
```
OUTPUT:
Enter a number: 1
Enter another number: 2
----MENU----
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your Choice: 1

Addition of 1 and 2 is 3
```

Repetition

Repetition statements are used to repeat a group of programming instructions.

Example: Whenever you do the same thing over and over again this called a loop.
i.e our daily routine.

Let's understand for and while loop



1. while loop:

In Python, while loops execute a block of statements repeatedly until the given condition is satisfied. Then, the expression is checked again if it is still true, the body is executed again. This continues until the expression becomes false.

Syntax:

```
while condition:
    # do something
```

Example 1: Print numbers from 1-10

```
i = 1

while i<=10:
    print(i)
    i+=1
```

OUTPUT:

```
1
2
3
4
5
6
7
8
9
10
```

Example 2: Print only even numbers from 1-10

```
i = 1

while i<=10:
    if i % 2 == 0:
        print(i)
    i+=1
```

OUTPUT:

```
2
4
6
8
10
```

Example 3: **Print even-odd numbers from 1-10 and also count them**

```
i = 1
count1 = 0
count2 = 0

while i<=10:
    if i % 2 == 1:
        print(f"Odd: {i}", end="    ")
        count1+=1
    else:
        print(f"Even: {i}")
        count2+=1
    i+=1

print(f"\nTotal Odd Numbers are: {count1}")
print(f"Total Even Numbers are: {count2}")
```

```
OUTPUT:
Odd: 1      Even: 2
Odd: 3      Even: 4
Odd: 5      Even: 6
Odd: 7      Even: 8
Odd: 9      Even: 10

Total Odd Numbers are: 5
Total Even Numbers are: 5
```

Example 4: **Write a program to check whether a given 3-digit number is an Armstrong number or not.**

- **What is an Armstrong number?**
- Definition of the Armstrong Number: An n-digit Armstrong number is a special number which is equal to the sum of its own digits, each raised to the power of n.
  An 3-digit Armstrong number is a special number which is equal to the sum of its own digits, each raised to the power of 3.

$$abcd.... = a^n + b^n + c^n + d^n + ...$$

$$153 = 1^3 + 5^3 + 3^3 = 153$$

```
num = int(input("Enter a 3-digit Number: "))
sum = 0
temp = num

while temp>0:
    digit = temp % 10
    sum += digit**3
    temp //= 10

if num == sum:
    print(f"{num} is an Armstrong number.")
else:
    print(f"{num} is not an Armstrong number.")
```

```
OUTPUT:
Enter a 3-digit Number: 153
153 is an Armstrong number.
```

Example 5: **Write a program to check whether a given string is a palindrome or not.**

Palindrome?
A Palindrome is a word, or a sentence, or a number that reads the same backwards as forwards
Example: level, madam, 1221, madam, etc.

```
s = input("Enter a String: ")
s = s.replace(" ", "").lower()

start = 0
end = len(s) - 1
flag = True

while start < end:
    if s[start] != s[end]:
        flag = False
    start += 1
    end -= 1

if flag:
    print("The string is a Palindrome.")
else:
    print("The string is not a Palindrome.")
```

```
OUTPUT:
Enter a String: level
The string is a Palindrome.
```

Example 6: **Write a program to find the result of base^{pow}.**

```
base = int(input("Enter Base: "))
pow = int(input("Enter Power: "))

result = 1; i = 1

if pow == 0:
    print(f"{base} raised to the power {pow} is {result}")
else:
    while i<=pow:
        result *= base
        i += 1

print(f"{base} raised to the power {pow} is {result}")
```

```
OUTPUT:
Enter Base: 9
Enter Power: 3
9 raised to the power 3 is 729
```

2. for loop:

for loop is used to iterate over elements of a sequence such as string, list, tuple, set, dictionary. for loop repeat a set of statements a finite number of times.

Syntax:

```
for var in iterable:
    # do something
```

Example 1: **Print numbers from 1-10**

```
for i in range(1, 11):
    print(i)
```

OUTPUT:

```
1
2
3
4
5
6
7
8
9
10
```

Example 2: **Print numbers from 10-1**

```
for i in range(10, 0, -1):
    print(i)
```

OUTPUT:

```
10
9
8
7
6
5
4
3
2
1
```

Example 3: **Display multiplication table, take input from user**

```
num = int(input("Display multiplication table of: "))

for i in range(1, 11):
    print(num, 'x', i, "=", num*i)
```

OUTPUT:

```
22 x 1 = 22
22 x 2 = 44
22 x 3 = 66
22 x 4 = 88
22 x 5 = 110
22 x 6 = 132
22 x 7 = 154
22 x 8 = 176
22 x 9 = 198
22 x 10 = 220
```

Example 4: Write a program to check whether a given number is prime or not.

Prime Numbers:

A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

Example: 2, 3, 5, 7, 11, 13, ... are all prime numbers.

```
num = int(input("Enter an Integer: "))

if num <= 1:
    print(f"{num} is not prime.")
elif num > 1:
    for i in range(2, num):
        if num % i == 0:
            print(f"{num} is not prime.")
            print(f"{i} is the factor of {num}")
            break
    else:
        print(f"{num} is prime.")


OUTPUT:
Enter an Integer: 45
45 is not prime.
3 is the factor of 45
```

Example 5: Write a program to find the HCF (Highest Common Factor) of two numbers.

Highest Common Factor (HCF): The Highest Common Factor (HCF) of two number is the largest positive integer that perfectly divides the given two numbers.

Example: 1. HCF of 12 and 18 is 6.

Factors of 12: 1, 2, 3, 4, **6**, 12

Factors of 18: 1, 2, 3, **6**, 9, 18

```
num1 = int(input("Enter the first Number: "))
num2 = int(input("Enter the second Number: "))

if num1 < num2:
    smaller = num1
else:
    samller = num2

for i in range(1,smaller + 1):
    if ((num1 % i ==0) and (num2 % i == 0)):
        hcf = i

print(f"The HCF is {hcf}.")


OUTPUT:
Enter the first Number: 24
Enter the second Number: 62
The HCF is 2.
```

Example 6: Write a program to print the Fibonacci Sequence.

What is Fibonacci Sequence?
A Fibonacci Sequence is a sequence in which each term can be obtained by adding the previous two terms.
Example: 0, 1, 1, 2, 3

```
terms = int(input("Enter the number of terms: "))

n1, n2 = 0, 1

if terms<=0:
    print("Please enter a positive integer.")
elif terms == 1:
    print(f"Fibonnaci Sequence: {n1}")
else:
    for term in range(terms):
        print(n1,end=" ")
        n = n1 + n2
        n1 = n2
        n2 = n
```

```
OUTPUT:
Enter the number of terms: 10
0 1 1 2 3 5 8 13 21 34
```

for Loop vs while Loop

- for Loop needs an iterable object to iterate.
- while Loop executes based on some condition.

```
Syntax:

for var in iterable:
    # do something
```

```
Syntax:

while condition:
    # do something
```

- for Loop is used when the number of iterations is known in advance.
- while Loop is used when the number of iterations is not known in advance.

- Both for and while loop can run infinite times.

3. nested-for Loop

Example 1:

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
rows = int(input("Enter number of Rows: "))

for i in range(1, rows+1):
    for j in range(1, rows+1):
        print("* ", end = "")
    print()
```

Example 2:

```
*
* *
* * *
* * * *
* * * * *
```

```
rows = int(input("Enter number of Rows: "))

for i in range(1, rows+1):
    for j in range(1, rows+1):
        print("* ", end = "")
    print()
```

Example 3:

```
* * * * *
* * * *
* * *
* *
*
```

```
rows = int(input("Enter number of Rows: "))

for i in range(rows,0,-1):
    for j in range(1, i+1):
        print("* ", end = "")
    print()
```

Example 4:

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
rows = int(input("Enter number of Rows: "))

for i in range(1,rows+1):
    for j in range(1, i+1):
        print(i, end = " ")
    print()
```

Example 5:

```
    *
   * *
  * * *
 * * * *
* * * * *
```

```
rows = int(input("Enter number of Rows: "))

for i in range(1,rows+1):
    for j in range(1, rows-i+1):
        print(" ", end = "")
    for j in range(1, i+1):
        print("* ", end = "")
    print()
```

Example 6:

```
    * * * * *
   * * * * *
  * * * * *
 * * * * *
* * * * *
```

```
rows = int(input("Enter number of Rows: "))

for i in range(1,rows+1):
    for j in range(1, rows-i+1):
        print(" ", end = "")
    for j in range(1, rows+1):
        print("* ", end = "")
    print()
```

Example 7:

```
* * * * *
 * * * *
  * * *
   * *
    *
```

```
rows = int(input("Enter number of Rows: "))

for i in range(rows,0,-1):
    for j in range(1, rows-i+1):
        print(" ", end = "")
    for j in range(1, i+1):
        print("* ", end = "")
    print()
```

Example 8:

```
1 2 3 4 5
 1 2 3 4
  1 2 3
   1 2
    1
```

```
rows = int(input("Enter number of Rows: "))

for i in range(rows,0,-1):
    for j in range(1, rows-i+1):
        print(" ", end = "")
    for j in range(1, i+1):
        print(j, end = " ")
    print()
```

Try This: