Iteration

For loop	While loop
for i in range (...,...):	i = 1
-print (i * 100)	while i <= 5:
-> number of reptationis known.	-print (i * 100)
	-i = i + 1
	-> number of repetitions is unknown.

In summary, while loops are more flexible in terms of the condition they evaluate, making them suitable for situations where the number of iterations is not known beforehand or might change during runtime. For loops are ideal when you need to iterate over a sequence of known length or a predefined collection.

Sub Programs

declaration of the procedure

```
def proced ure _na me( param01 , param02):
    action(s)
    proced ure _na me( par am01, param02)
```

declaration of the function

```
def functi on_ nam e(p ara 1_, para2):
    action(s)
    return variab le_name / expression
```

In summary, the key difference lies in whether the block of code returns a value or not. Functions return values, while procedures do not. However, in languages like Python, the distinction is less strict, as functions can return None and procedures can still be defined using functions that return None. The choice of using functions or procedures depends on the specific requirements of the task and the programming paradigm being followed.

Recursivity

```
def factor ial(n):
    if n == 0:
        return 1
    else:
        return n * factor ial(n - 1)
```

This is the condition that stops the recursive calls.

Recursion can be a powerful tool for solving problems that can be broken down into smaller, similar subproblems. However, it's essential to ensure that the base case is reachable and that the recursive calls converge towards the base case to avoid infinite recursion. Additionally, recursive solutions may not always be the most efficient, as they can consume more memory due to the recursive calls creating a new stack frame for each function call.

String (cont)

```
print (s1)  print (s2)     print ("s1 and s2 are differ ent - ")
-> displays "-1LBC2"  ->displays "-LBC"  if s1 == s3:
                             print ("s1 and s3 are simila r")
                           else:
                             print ("s1 and s3 are differ ent - ")
```

String

String	Length	concatenations	comparing	iterating	ch m
Assignment:	`s1 = " 1LB C1"`	`s = " 1LB C1 "`	`s1 = " abc d"`	`for c in s:`	cc
`variab le_name = " val - ue"`	`L=len(s1)`	`s1 = " ASD P 2"`	`s2 = " abc d"`	`code`	cc
`s1 = 1LBC1`	`print(L)`	`s2 = s + s1`	`s3 = " abc d1"`	`code`	cc
Access to characters	*-> displays 5*	`print (s2)`	`if s1 == s2:`	`code`	cc
`s1 = " 1LB C1"`	`s1 = " 1LB C1"`	*->display "-1LBC1 ASP2"	`print ("s1 and s2 are simila r" )`	`code`	cc
`s1 = " 1LB C2"`	`s2 = s1 [1 : 4]`		`else:`	`code`	cc