Level 1: If statement

A program sometimes may have to make choices. These choices can execute different code depending on a certain condition. In Python the **if statement** is used for conditional execution.

The if statement may be combined with certain operator such as equality (==), greater than (>), smaller than (<) and not equal (!=). Conditions may be combined using the keywords **or** and **and**.

<u>Problem to solve:</u>

Write a program that prompts the user to input a year and determine whether the year is a leap year or not.
Leap Years are any year that can be evenly divided by 4. A year that is evenly divisible by 100 is a leap year only if it is also evenly divisible by 400.

Example:

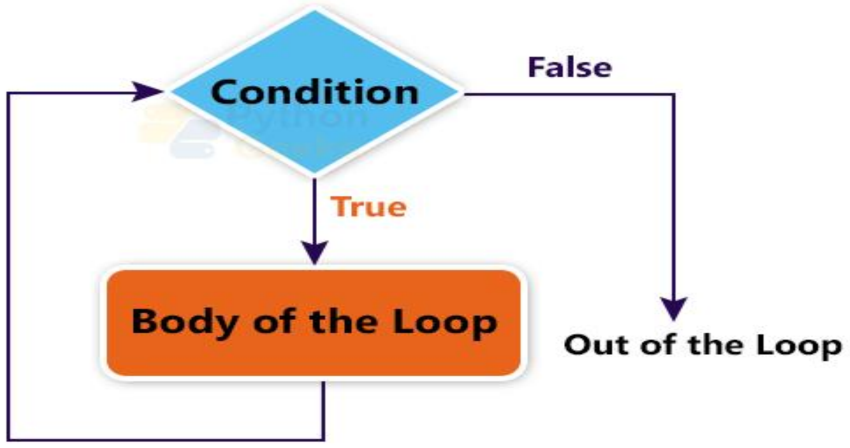
1992	Leap Year
2000	Leap Year
1900	NOT a Leap Year
1995	NOT a Leap Year

Level 2: Loops

In programming, the loops are the constructs that repeatedly execute a piece of code based on the conditions. These are useful in many situations like going through every element of a list, doing an operation on a range of values, etc.

There are two types of loops in Python, and these are **for** and **while** loops. Both work by following the below steps:

1. Check the condition
2. If True, execute the body of the block under it. And update the iterator (the value on which the condition is checked)
3. If False, come out of the loop



While Loop in Python

While loops execute a set of lines of code iteratively till a condition is satisfied. Once the condition results is False, it stops execution, and the part of the program after the loop starts executing.

The syntax of the while loop is:

```
while condition:
    statement(s)
```

Python For loop

For loop in Python works on a sequence of values. For each value in the sequence, it executes the loop till it reaches the end of the sequence. The syntax for the for loop is:

```
for iterator in sequence:
        statement(s)
```

Infinite loops

Infinite loop is the condition where the loop executes infinite time -> it does not stop the execution of the loop. This happens in two cases:

a. When we forget to increment the variable based on which the condition is checked.

b. When we give a wrong condition.

Be careful of infinite loops in your program, **always** make sure that your loop will stop when it needs to.

Problem to solve:

Make a multiplication table using a loop (you can use any loop you would like)

Example:

```
     1      2      3      4      5      6      7      8      9      10
1    1      2      3      4      5      6      7      8      9      10
2    2      4      6      8      10     12     14     16     18     20
3    3      6      9      12     15     18     21     24     27     30
4    4      8      12     16     20     24     28     32     36     40
5    5      10     15     20     25     30     35     40     45     50
6    6      12     18     24     30     36     42     48     54     60
7    7      14     21     28     35     42     49     56     63     70
8    8      16     24     32     40     48     56     64     72     80
9    9      18     27     36     45     54     63     72     81     90
10   10     20     30     40     50     60     70     80     90     100
```

Level 3: Functions

A function is a group of related statements that performs a specific task.

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

Furthermore, it avoids repetition and makes the code reusable.

Syntax of Function

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

Above shown is a function definition that consists of the following components.

1. Keyword def that marks the start of the function header.
2. A function name to uniquely identify the function.
3. Parameters (arguments) through which we pass values to a function. They are optional.
4. A colon (:) to mark the end of the function header.
5. Optional documentation string (docstring) to describe what the function does.
6. One or more valid python statements that make up the function body. Statements must have the same indentation level (usually 4 spaces).
7. An <u>optional</u> return statement to return a value from the function.

How to call a function in python?

Once we have defined a function, we can call it from another function, program, or even the Python prompt. To call a function we simply type the function name with appropriate parameters.

```
>>> greet('Joe')

Hello, Joe. Good morning!
```

Note: In python, the function definition should always be present before the function call. Otherwise, we will get an error.

The return statement

The return statement is used to exit a function and go back to the place from where it was called.

Syntax of return

```
return [expression_list]
```

This statement can contain an expression that gets evaluated and the value is returned. If there is no expression in the statement or the return statement itself is not present inside a function, then the function will return the **None** object.

For example:

```
>>> print(greet("Leo"))

Hello, Leo. Good morning!

None
```

Here, None is the returned value since greet() directly prints the name and no return statement is used.

Scope and Lifetime of variables

Scope of a variable is the portion of a program where the variable is recognized. Parameters and variables defined inside a function are **not visible from outside the function**. Hence, they have a **local scope**.

The lifetime of a variable is the period throughout which the variable exists in the memory. The lifetime of variables inside a function is if the function executes.

They are destroyed once we return from the function. Hence, a function does not remember the value of a variable from its previous calls.

Types of Functions

Basically, we can divide functions into the following two types:

1. Built-in functions - Functions that are built into Python. Ex: print() function
2. User-defined functions - Functions defined by the users themselves. Ex: functions that we did in the lab

Problem to solve:

Write a Python function to check whether a number is perfect or not.

According to Wikipedia: In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself (also known as its aliquot sum). Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself).

Example: The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and 1 + 2 + 3 = 6. Equivalently, the number 6 is equal to half the sum of all its positive divisors: (1 + 2 + 3 + 6) / 2 = 6. The next perfect number is 28 = 1 + 2 + 4 + 7 + 14. This is followed by the perfect numbers 496 and 8128.

Level 4: Strings

A **string** is a data type used in programming, such as an integer and floating-point unit, but is used to represent text rather than numbers. It is comprised of a set of characters that can also contain spaces and numbers. For example, the word "hamburger" and the phrase "I ate 3 hamburgers" are both strings. Even "12345" could be considered a string, if specified correctly. Typically, programmers must enclose strings in quotation marks for the data to recognized as a string and not a number or variable name.

For example, in the comparison:

if (Hello== Helloyou) then ...

Hello and Helloyou may be variables containing integers, strings, or other data. If the values are the same, the test returns a value of true, otherwise the result is false. In the comparison:

if ("Hello" == "Helloyou") then ...

Hello and Helloyou are being treated as strings. Therefore, the test is comparing the words " Hello" and " Helloyou," which would return false.

Problem to solve:

Write a function that will take a string and returns the reverse of this string.

Example1: Input: “`olleh`”

Output: “`hello`”

Example2: Input: “`gnirts desrever a si siht`”

Output: “`this is a reversed string`”

Remember: the input of the function is what the function takes as argument, and the output of the function is what the function returns (sometimes the output needs to be printed and not returned so the function just prints the answer and do not return anything).

Try your function on this string:

“`erutuf eht ni ti deen lliw uoy drow siht peek namretaw si deen lliw uoy drow terces ehT`”

Level 5: recursive method

A **method** that calls itself is known as a **recursive method**, and this technique is known as **recursion**.

A recursive method should have a **base case** to ensure that the call stops at one point, or else it will be an infinite call and it will **never stop.**

Therefore, you should **NEVER** use a function call inside the same function without a way to stop it.

Example of a recursive method without a base case:

```
def infinite ():
  print("still running")
  return infinite()
```

This method will not stop because every time it prints "still running" it will be called again through "return infinite()" and it will enter the method again.

Example of a recursive method with a base case:

```
def infinite (count):
  if count<=0:
    return "done"
  print("still running")
  count-=1
  return infinite(count)
```

This method will enter, if count is <=0 it will return "done" and finish, If count is still bigger than 0 it will print "still running", decrease count, and then call the method again with the new count until it reaches the base case.

You will learn Recursion and all about recursive methods in CP2. Until you learn how to use them, please **DO NOT CALL A FUNCTION INSIDE THE SAME FUNCTION.** Thank you 😊

Problem to solve:

Find the recursive methods:

```
def method1(count):

  if (count<=0):
    return count
  else:
    return count+method1(count-1)
```

```python
def method2(count):
  sum=0
  for i in range (count+1):
    sum+=i
  return sum
```

```python
def method3 (nterms):
  # first two terms
  n1, n2 = 0, 1
  count = 0
  # check if the number of terms is valid
  if nterms <= 0:
    print(-1)
  # if there is only one term, return n1
  elif nterms == 1:
    print(n1)
  # generate fibonacci sequence
  else:
    print("Fibonacci sequence:")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        # update values
        n1 = n2
        n2 = nth
        count += 1
```

```python
def method4 (n):
  if n <= 1:
       return n
  else:
       return(method4(n-1) + method4(n-2))
```

```python
def method5(x):
    """This is a function to find the factorial of an integer"""
    if x == 1:
        return 1
    else:
        return (x * method5(x-1))
```

Next Exercises:

To open the next set of exercises you need the password to this link:

https://lauedu74602-my.sharepoint.com/:f:/g/personal/nour_kfoury_lau_edu/Eko1HPuQVQtPmOOncghkrbcBpV6fF3AAaO6RpLjeKWBfbg?e=5fSzTZ

To get the password you need to apply the function you created in Level 4 on the string below, get the secret word add to it the number of the methods that are recursive in level 5 and you have your password.
Example: if the secret word you got in level 4 is “mypass” and method 1 and 2 are the recursive methods in level 5 then your password is: mypass12

The string to apply level 4 on is:
“erutuf eht ni ti deen lliw uoy drow siht peek namretaw si deen lliw uoy drow terces ehT”

References:

https://www.programiz.com/python-programming/function#:~:text=In%20Python%2C%20a%20function%20is%20a%20group%20of,it%20avoids%20repetition%20and%20makes%20the%20code%20reusable.

https://pythongeeks.org/loops-in-python/#:~:text=Introduction%20to%20Loops%20in%20Python%20In%20programming%2C%20the,Python%20and%20these%20are%20for%20and%20while%20loops.

https://pythonbasics.org/if-statements/