----------1----------

As an expression, lambda returns a value that can optionally be assigned a name. In contrast, the def statement always assigns the new function to the name in the header, instead of returning is as a result. lambda's body is a single expression, not a block of statements.

----------2----------

The lambda keyword in Python provides a shortcut for declaring small anonymous functions. Lambda functions behave just like regular functions declared with the def keyword. They can be used whenever function objects are required.

----------3----------

Map()--

The basic function of map() is to manipulate iterables. Map executes all the conditions of a function on the items in the iterable. In the above example, we have multiplied each element in the range 0–10 with 2 which gives us completely new list of elements. Map function takes all elements and allows you to apply a function on it and then passes it to the output which can have same as well as different values .

Filter()--

As the name suggests, it is used to filter the iterables as per the conditions. Filter filters the original iterable and passes the items that returns True for the function provided to filter. Therefore only the items in the iterables can be expected to be seen in the output. In the above example the condition is given in the form of lambda function and the elements which satisfy the condition are given in the list. The elements which are divisible by 2 are left and others are filtered out.

reduce()--

Python's reduce() is a function that implements a mathematical technique called folding or reduction. reduce() is useful when you need to apply a function to an iterable and reduce it to a single cumulative value.

----------4.----------

Function annotations are completely optional both for parameters and return value. Function annotations provide a way of associating various parts of a function with arbitrary python expressions at compile time.

----------5----------

A recursive function is a function that calls itself during its execution. The process may repeat several times, outputting the result and the end of each iteration. The function Count() below uses recursion to count from any number between 1 and 9, to the number 10.

----------6----------

Indeed coding and applying logic is the foundation of any programming language but there’s also another factor that every coder must keep in mind while coding and that is the coding style. Keeping this in mind, Python maintains a strict way of order and format of scripting.Following this sometimes mandatory and is a great

help on the user’s end, to understand. Making it easy for others to read code is always a good idea, and adopting a nice coding style helps tremendously for that.

1. Use 4-space indentation and no tabs.
Examples:

```
# Aligned with opening delimiter.
grow = function_name(variable_one, variable_two,
                     variable_three, variable_four)
# First line contains no argument. Second line onwards
# more indentation included to distinguish this from
# the rest.
def function_name(
        variable_one, variable_two, variable_three,
        variable_four):
    print(variable_one)
```

The 4 space rule is not always mandatory and can be overruled for continuation line.

2. Use docstrings : There are both single and multi-line docstrings that can be used in Python. However, the single line comment fits in one line, triple quotes are used in both cases. These are used to define a particular program or define a particular function.
Example:

```
def exam():
    """This is single line docstring"""

    """This is
    a
    multiline comment"""
```

3. Wrap lines so that they don’t exceed 79 characters : The Python standard library is conservative and requires limiting lines to 79 characters. The lines can be wrapped using parenthesis, brackets, and braces. They should be used in preference to backslashes.
Example:

```
with open('/path/from/where/you/want/to/read/file') as file_one, \
     open('/path/where/you/want/the/file/to/be/written', 'w') as file_two:
    file_two.write(file_one.read())
```

4. Use of regular and updated comments are valuable to both the coders and users : There are also various types and conditions that if followed can be of great help from programs and users point of view. Comments should form complete sentences. If a comment is a full sentence, its first word should be capitalized, unless it is an identifier that begins with a lower case letter. In short comments, the period at the end can be omitted. In block comments, there are more than one paragraphs and each sentence must end with a period. Block comments and inline comments can be written followed by a single ‘#’.
Example of inline comments:

```
geek = geek + 1                 # Increment
```

5. Use of trailing commas : This is not mandatory except while making a tuple.
Example:

```
tup = ("geek",)
```

5. Use Python’s default UTF-8 or ASCII encodings and not any fancy encodings, if it is meant for international environment.

6. Use spaces around operators and after commas, but not directly inside bracketing

constructs:

a = f(1, 2) + g(3, 4)

7. Naming Conventions : There are few naming conventions that should be followed in order to make the program less complex and more readable. At the same time, the naming conventions in Python is a bit of mess, but here are few conventions that can be followed easily.
There is an overriding principle that follows that the names that are visible to the user as public parts of API should follow conventions that reflect usage rather than implementation.
Here are few other naming conventions:

b (single lowercase letter)

B (single upper case letter)

lowercase

lower_case_with_underscores

UPPERCASE

UPPER_CASE_WITH_UNDERSCORES

CapitalizedWords (or CamelCase). This is also sometimes known as StudlyCaps.
Note: While using abbreviations in CapWords, capitalize all the letters of the abbreviation. Thus HTTPServerError is better than HttpServerError.

mixedCase (differs from CapitalizedWords by initial lowercase character!)

Capitalized_Words_With_Underscores

----------7----------

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

Function blocks begin with the keyword def followed by the function name and parentheses (( )).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The first statement of a function can be an optional statement - the documentation string of the function or docstring.

The code block within every function starts with a colon (:) and is indented.

The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.