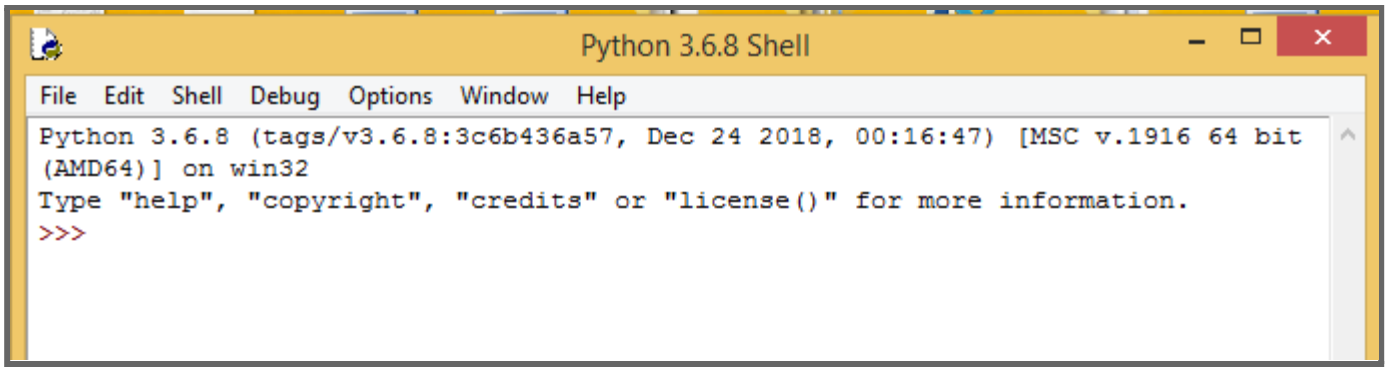
PYTHON

A. Quick Recap

We start Python by double clicking on the Python shortcut icon on the desktop or on the taskbar. You can also use Cortana to search for IDLE and open it.

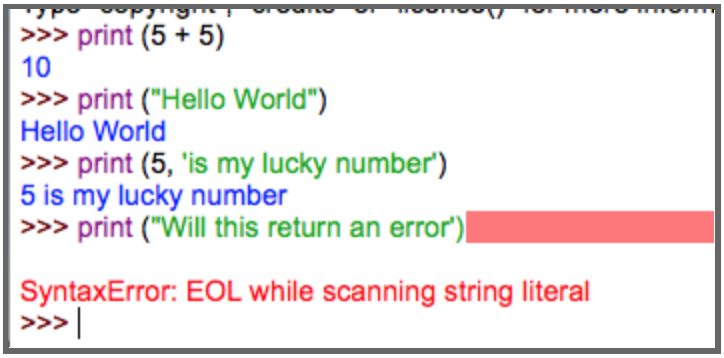
Python IDLE stands for Integrated Development and Learning Environment. Python IDLE is the interactive mode of Python where we type program commands one line (or one block of code) at a time. This line of code gets executed instantly and the output is displayed on the IDLE.

A.1 Python IDLE:



Print function in Python 3 version performs the same task as Python print statement in 2.7. It helps us display things. If you want to print some text like the example given below, you put the text within quotes. Python allows both single as well as double quotes for inputting text.

If you want to print a numeric value, then do not use quotes.



The first print function causes 10 to be printed. Since it is a number, we do not use quotes while printing it.

The second print function prints text. So, we need to use quotes.

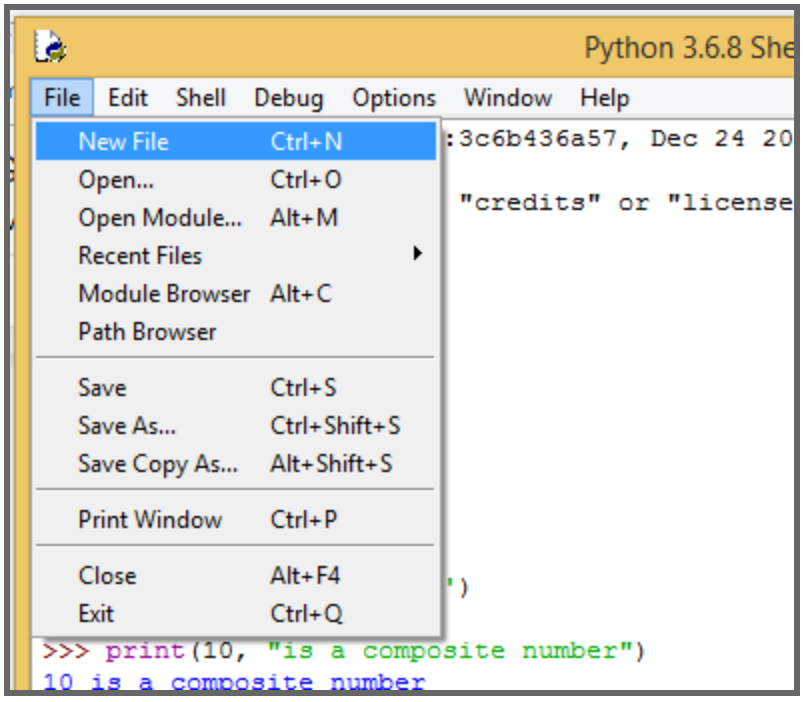
The third print function uses a combination of number and text. We use a comma to separate number and text here.

The fourth print function uses double quotes at the beginning of text and single quotes at the end. This causes Python to return an error. You can see that an error is returned by Python (line in red). Let us correct this error by matching the single quote at the end with one at the beginning.

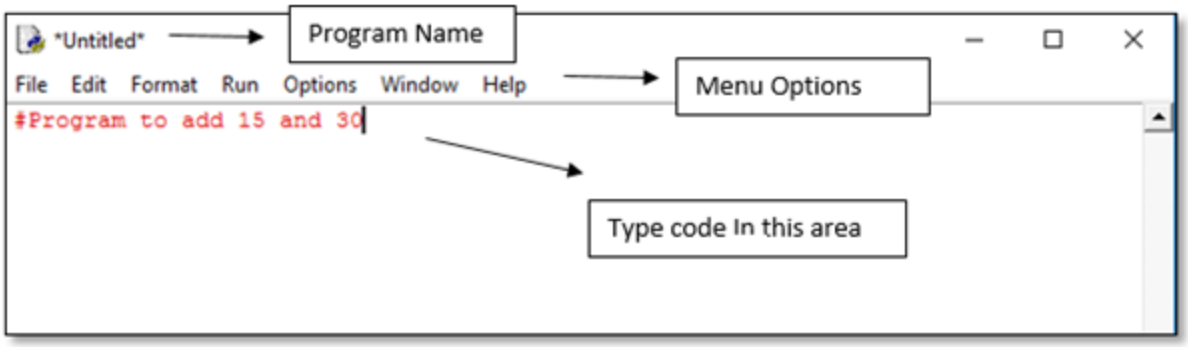
A.2 Script Mode in Python:

A script in Python is a set of code that is saved as a file. The extension for a Python script file is .py. A script is also known as a module or a program. To create a new script/program/module in Python:

1. Click on File in IDLE.
2. Click on New File.

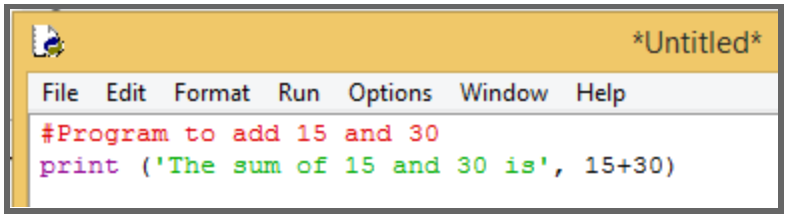


The New File that opens is a text editor and looks like this:

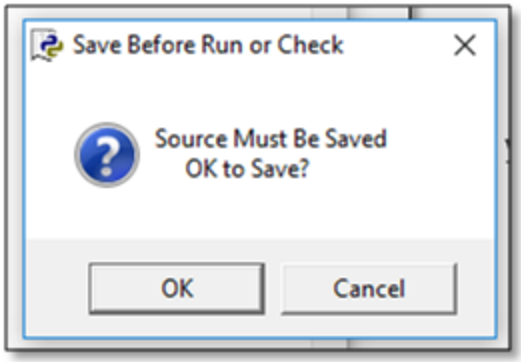


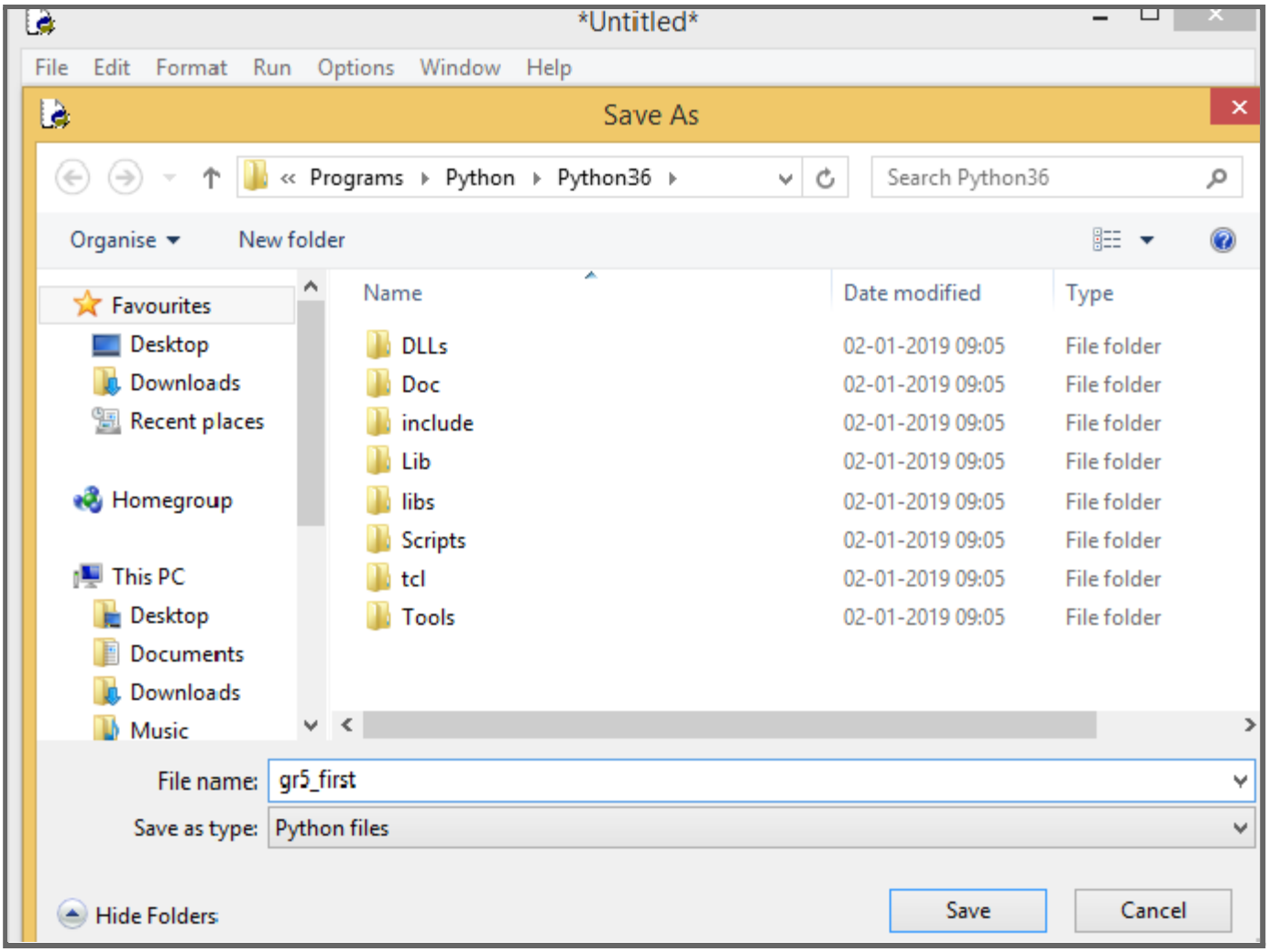
We can execute a program by following these steps:

1. Click on Run and select Run Module.



2. You will see a dialog box open that asks you to save your program first. Give a suitable name for the program and save it. In Windows, the default location is Python folder.





3. IDLE pops up with the output:

```
= RESTART: C:/Users/USER/AppData/Local/Programs/Python/Python36/gr5_first.py =
The sum of 15 and 30 is 45
```

Every time we run a module, Python shell (IDLE) is restarted.

A.3 Arithmetic operators in Python:

Arithmetic operators are used for numerical calculations.

A+ B	Addition operation
A - B	Subtraction operation
A * B	Asterisk is used to denote the multiplication operation
A / B	Forward slash is used to denote division operation. Gives the quotient when A is divided by B. Division when A and B are integers: When we have 2 integers, then the answer for division is also an integer (in Python 2.7). Ex: 5/2 gives 2 instead of 2.5. 10/3 gives 3 instead of 3.3333. If any of the variables is float, then the answer is in float. Ex: 5.0/2 gives 2.5.
A % B	% operator is known as the modulo operator and gives the remainder when A is divided by B.

A.4 Relational Operators:

Relational operators are used when we need to compare two quantities.

A > B	Checks if A is greater than B
A < B	Checks if A is less than B

A == B	Checks if A is equal to B. == operator is different from = operator. = is used for assigning a value to a variable. == is used for checking if 2 values are equal.
A >= B	Checks if A is greater than or equal to B
A <= B	Checks if A is less than or equal to B
A <> B A != B	Checks if A is not equal to B. There is no difference between <> and != .

A.5 Practice Exercise in IDLE

Type the given commands in IDLE and note down the output given by Python. Write the reason for getting that output.

Command	**Output**	**Reason**
print (‘One step at a time’)		
print (“What’s the next step?”)		
print (**“**What’s the next step?**’**)		
A = 3.4		
A = 2 + 3.0 print (A)		
9/4		
9/4.0		
9.0/2		
B = A B = 3 B == A		
17 % 7		
17.0%7		
‘Ha’ + ‘Ha’ + ‘Ha’		
‘Ha’*3		

'Ha' * '3'		
print (6, 'is a perfect number')		
print ('6' + 'is a perfect number')		
print ('red') print ('green') print ('blue')		
print ('red' , 'green', 'blue')		

A.6 User Input in Python:

We use the input( ) function to accept user value in Python 3. This has the same functionality as raw_input() in Python 2.7. Observe the example given below:

```
#Program to display name and age of user
name = input('Enter your name ')
age = input('Enter your age ')
print ('Your name is', name)
print('Your age is',age)
```

Output seen in IDLE:

```
Enter your name Ria
Enter your age 11
Your name is Ria
Your age is 11
```

input() accepts any value given by the user and assigns it to the variable on the left hand side. The text that appears inside the input() is the prompt given to the user. If you want to ask the user to input his name, then the prompt can be 'Enter your name'.

Note:input() accepts any input given by user as string (text) value. But numbers are not text. If we want to accept numbers, we need to use an additional function outside the input().

If the user has to input an integer, we use int() which encloses input(). If the user has to input decimal fractions, we use float().

```
#Program to add two integers
n1 = int(input('Enter first number '))
n2 = int(input('Enter second number '))
s = n1 + n2
print ('Sum of entered numbers is', s)
```

Note: A comma separator in print function adds a space automatically in output. In input(), we need to put an extra space when typing our prompt so that there is a space seen after the prompt when it appears on IDLE.

A.7 Practice Time - Sequential Construct

1. Write a program to accept 2 numbers of float data type and find the difference A-B.
2. Write a program to accept 3 numbers and print the product of the three numbers.
3. Write a program to accept the length and breadth of a rectangle and find the area and perimeter.
4. Write a program to accept radius measurement of a circle and print its area and circumference.

B. IF Statements in Python

If statement checks if a condition is true. We have already seen how conditions can be checked in the Algorithms and Flowcharts unit. When a condition is true, one branch of the program gets executed. Otherwise, another branch of the program gets executed. In Python, **If** statements form a block of statements. (A block is a set of statements which are executed as a unit.)

B.1 Simple IF Statement

Syntax for a simple IF statement:

In this case, a condition is tested. If it is true, then a code block is executed.

if <condition to be tested>**:**

    <statements to be executed>

Notice that the statements to be executed are indented. Python automatically indents statements when it reads the colon separator. Colon separator indicates the beginning of a code block.

Using simple IF statement:

The program given below checks if the integer entered is positive. We will get an output only for positive integers.

```
#Program to check if a number is positive
a = int(input('Enter an integer '))
if a > 0: #check if a is positive
   print (a, 'is positive')
```

Output for the program:

```
Enter an integer 9
9 is positive
>>>
========== R
Enter an integer -5
```

But wouldn't it be more informative if we had an output when the entered number is not positive? We need an IF-ELSE block in order to achieve this.

B.2 IF-ELSE Statement

Syntax for IF-ELSE Statement:

if <condition to be tested>**:**

    <statements to be executed if condition is true>

else:

<statements to be executed if condition is not true>

Using IF-ELSE Statement:

Let us add the ELSE clause in our program to find if an integer is positive. If it is not positive, we will display an output that says the number is not positive.

```
#Program to check if a number is positive
a = int(input('Enter an integer '))
if a > 0: #check if a is positive
   print (a, 'is positive')
else:
   print (a, 'is negative')
```

Output for this program:

```
Enter an integer 6
6 is positive
>>>
Enter an integer -6
-6 is negative
```

B.3 Practice Exercise - Simple IF statements

1. Write a program that accepts a number and checks if it is divisible by 3.
2. Write a program to check if a given number is odd or even.
3. Write a program that accepts 2 numbers and prints the smaller number.
4. In the previous program, what happens if both the numbers are equal? Justify your answer.

B.4 IF-ELIF-ELSE Statement in Python

If we want to test the truth of a condition, we can use IF or IF-ELSE statements. But, what if we have multiple conditions that need to be tested? One example of such a scenario is when we check the marks obtained by a student and allot a grade. Look at the table given below:

Marks	Grade
Marks < 35	F
35 <= Marks < 50	C
50<= Marks < 75	B
75 <= Marks <= 100	A

When the marks lie between a particular range, a corresponding grade will be given. This requires not just one IF statement, but multiple statements. This type of IF statements are known as concatenated IF statements.

Syntax for IF-ELIF-ELSE Block:

if <condition1>**:**
    <statements>
elif <condition2>**:**
    <statements>
elif <condition3>**:**
    <statements>
:
:
else:
    <statements>

Example 1: Program to allot grade based on marks given by user.

```
#Program that outputs grade based on marks input
name = input('Enter your name ')
m = float(input('Enter your marks '))

print() #leave a line before starting output
print('Name:', name)

if m >= 75 and m <= 100: #and operator checks for both limits simultaneously
    print('Grade: A') #Assign grade A

elif m >= 50:
    print('Grade: B') #Assign grade B

elif m >= 35:
    print('Grade: C') #Assign grade C

else:
    print('Grade: F')#Assign grade F
```

Output:

```
========== RES
Enter your name Ria
Enter your marks 77

Name: Ria
Grade: A
>>>
========== RES
Enter your name Gaurav
Enter your marks 35

Name: Gaurav
Grade: C
```

Points to ponder:

- **and** operator ensures that marks are in the range 75 to 100. It is a logical operator. There are three Logical operators in Python: *not*, *and*, *or*. When we want to check if multiple conditions are simultaneously true, we use *and*. When we want to check if any one of the conditions is true out of many, we use *or*.
- The upper limit of marks is not checked in the elif statements. Why do you think we need not check it?
- What happens if a user inputs negative number? How can we modify the program such that user gets a message 'Invalid input' if negative numbers are input.

Example 2: Accept a number and check if it is positive, negative or neutral.

```
#Accept a number and check if it is positive, negative or zero
n = int(input('Enter an integer '))
if n < 0:
    print (n, 'is negative')
elif n > 0:
    print (n, 'is positive')
else:
    print (n, 'is neutral')
```

Output:

```
========== REST
Enter an integer 4
4 is positive
>>>
========== REST
Enter an integer 0
0 is neutral
>>>
========== REST
Enter an integer -9
-9 is negative
```

Practice Exercise:

1. Write a program that accepts a number from 1 to 7 and displays the corresponding day of the week. If any other number is entered, program should display the message, “Invalid Number “.

C. Builtin Functions In Python

A function is a routine or a procedure in programming. It accepts some input, modifies the input and returns an output. Examples of Python function you have seen before include: input(), float(), int() etc. These are known as Built in functions since they already exist in Python and have not been created newly by anyone. The input given to a function is known as an argument. The output given by a function is known as its return value.

Let us take a look at some important builtin functions in Python: (All these functions can be used to compare values of different data types but this aspect is beyond the scope of this unit.)

Name	Description	Example
max(x, y, z, ...)	It returns the largest of its arguments.	`>>> max(12,-9,100)` `100`
min(x, y, z, ...)	It returns the smallest of its arguments.	`>>> min(5, 0, 2)` `0`

len (s)	It returns the length of a Python string or a list.	>>> len('red rose')
		8
		>>> len([1,2,3,4])
		4

range (start, stop, step)	This is very useful to generate sequences of numbers. range() in Python 3 returns an **iterable object**. What this means is that it holds a set of numbers that we will typically use in loops. How do we know what these numbers are?
	As seen in the output, we cannot really see the numbers, but only what we have given as input.
	In order to see the numbers stored in the created iterable object, we can use list().
		`>>> range(5)`
		`range(0, 5)`
		`>>> list(range(5))`
		`[0, 1, 2, 3, 4]`
		`>>> list(range(2,20,2))`
		`[2, 4, 6, 8, 10, 12, 14, 16, 18]`
		Why is 20 not part of the list?
		The upper limit is excluded while

	A list in Python is a series of values, which are separated by comma and placed inside square brackets. Each of these values is known as an element.
	
	Parameters in range()
	
	Start indicates the starting value of the number in the iterable.
	
	Stop value indicates the ending value in the list. However, in Python the end value is excluded from the iterable.
	
	Step value indicates by increment / decrement size from the first number.
	
	If we do not give the start value, Python assumes it to be 0.
	
	If we do not give the step size, Python assumes it to be 1.
	
	If two arguments are given, Python will assume they are the start and stop values, not stop and step values.
		generating the iterable.
		
		`>>> list(range(10,3))`
		`[]`
		Why do we get an empty list here? Python assumes that starting value is 10 and stop value is 3. Step size is 1, that means by incrementing 1 from 10, Python should reach 3. Since that is never possible, it returns an empty list.
		
		`>>> list(range(10,3,-1))`
		`[10, 9, 8, 7, 6, 5, 4]`
		A negative step size means, Python keeps decrementing 1 from 10 to arrive at 3. Remember, the stop value will not be included in the list.

round(x [, n])	This function returns a float value by rounding off a value upto n digits.	`>>> round(5.67,1)`
		`5.7`
		`>>> round(5.6777, 2)`
		`5.68`

C.1 Practice Exercise in IDLE

Type the given commands in IDLE and note down the output given by Python. Write the reason for getting that output.

Command	Output	Reason
len(5)		
len([1,2,3,4])		
len('apple')		
max(-9, 5, 0.5)		
max([1,2,3], 5.6)		
min(-22, -2, 22)		

list(range(2, 21, 3))		
list(range(4))		
list(range(10, 4, -2))		
list(range())		
round(33.336, 2)		
round(3.141592, 2)		

D. Loops in Python

A loop is the programmer's way of telling the computer to do something and keep doing until something changes. If we know the number of times a block of instructions has to be executed, we use the FOR loop. If we do not know how many times a specific block of instructions need to be executed, we use the WHILE loop in Python.
Real life examples of FOR loop:

- Completing 10 rounds of jogging around the park.
- Making 15 rotis for dinner.
- Reciting the multiplication table for any number.

Real life examples of WHILE loop:

- Tossing a coin repeatedly until you get Heads.
- Running around the park until you get tired.
- Blending pasta sauce ingredients into a paste until the right consistency is arrived at.

D.1 FOR loop using range():

FOR loop Syntax:

for i **in range(start, stop, step):**
    <statements>

We use the range function to create an iterable based on the start, stop and step values. The variable i stands for each of the numbers present in the iterable object.

Example 1: Consider the following example that prints the numbers 1 to 10 using for loop:

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


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

Notice that the stop value is 11 and not 10. range(1,11) returns the collection of numbers: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Since there are 10 elements in the iterable object, the loop runs 10 times. Each time the loop is run, the value of the variable i changes from 1 to 2 to 3 and so on.
We print each element using the print(i) function.

Example 2:

```
>>> for i in range(30, 20, -3):
        print(i)
```

What would be the output for the above example?
Let us create the list that would be created with the range function: [30,27,24,21].
So, the output should display each of these elements.

Output:

```
30
27
24
21
```

Example 3: Write a program that accepts user name and prints it 6 times.

```
#Accept user name and print it 6 times
name = input('Enter your name ')
for i in range(6):
    print(name)
```

range(6) creates the list [0, 1, 2, 3, 4, 5]. In this example, the for loop's only purpose is to keep track of the number of times the statements are executed. Since the name has to be printed 6 times, we need the loop to run 6 times. In other words, the number of elements present inside the list should be 6. If we start from 0 and stop at 5, we have 6 elements in the list. We can achieve the same result using this range function: range(1,7).
Output:

```
Enter your name Priya
Priya
Priya
Priya
Priya
Priya
Priya
```

Example 4: To print multiplication table for a user given number.

```
#Accept a number and print the multiplication table for the number
n = int(input('Enter a number '))
for i in range(1, 11):
    print(n,  'x' , i , '=' , n*i )
```

The range function creates this list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The variable i takes on the values present in the list. Let us assume that the user inputs a value of 3 for n.
Multiplication table for 3 should be:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
:
:
So, the print statement looks like this:
n x i = n*i
Output:

```
========== RESTART: /User
Enter a number 16
16 x 1 = 16
16 x 2 = 32
16 x 3 = 48
16 x 4 = 64
16 x 5 = 80
16 x 6 = 96
16 x 7 = 112
16 x 8 = 128
16 x 9 = 144
16 x 10 = 160
```

D.2 FOR loops without using range()

We can use FOR statement without using range function. Instead of creating an iterable object, we can use a list of values in our loops.

Example 5: What will be the output of these lines of code?

```
>>> L = ['a', 'b', 'c', 'd']

>>> for i in L:
        print(4*i)
```

Output:

```
aaaa
bbbb
cccc
dddd
```

i takes the value of ‘a’ first, then ‘b’ and so on.
What does 4*i mean? We are multiplying a string value with a number.

```
>>> 4*'s'
'ssss'
```

The * operator is known as replication operator when used with strings. It replicates the string that many times.

```
>>> 'abc'*3
'abcabcabc'
```

What happens if we multiply one string with another?

```
>>> 'd' * 'a'

Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    'd' * 'a'
TypeError: can't multiply sequence by non-int of type 'str'
```

Python returns an error. We can use the replication operator with strings only when there is one number and one string as operands.

Example 6: What will be the output of these lines of code?

```
>>> s = 'racecar'

>>> for i in s:
        print (i, end = ' ')
```

Here s is a string, not a list. But, s is still a sequence that has elements, just like a list. The variable, i takes on the values from the string - ‘r’, ‘a’, ‘c’, ‘e’ etc.
What is the use of end = ‘ ‘ in print()? This causes the next value to be printed on the same line instead of a new line.

Output:

```
r a c e c a r
```

D.3 Practice Exercise - FOR loop

1. Write a program to display the first 7 multiples of 35.
2. Find the sum of the first 10 natural numbers.
3. Accept the marks of 5 subjects from the user as float values and find the average of these marks.

4. Print the following pattern:
   1 1 1 1 1
   2 2 2 2 2
   3 3 3 3 3
5. Given that l = [1,3,5,7,9], print 2, 4, 6, 8, 10 using a FOR loop.

D.4 Introduction to While Loops

While Loop Syntax:
while <condition>:
    <statements>
In a while loop, we can test a condition. As long as the condition is true, the block of statements keeps getting executed. When the condition is no longer satisfied, control passes to the immediate statement after the while block.

Example 1: Accept a number from the user and print 'Hello' that many times using a while loop. (Note: This program is easier using FOR. This example is given to demonstrate how while loop works.)

```
#Accept a number and print hello that many times
n = int(input('Enter a number '))#number of times hello should be printed
while n > 0:
    print ('Hello ')
    n -= 1 #This is the same as n = n-1
```

Output:

```
Enter a number 5
Hello
Hello
Hello
Hello
Hello
```

The condition in the program is that as long as the number entered is greater than 0, 'hello' will be printed. Within the body of the loop, everytime we print 'hello', we have to reduce n by 1. This statement, n = n - 1 is known as the update statement. What will happen if we do not decrement n?

When n is not decremented:

```
#Accept a number and print hello that many times
n = int(input('Enter a number '))#number of times hello should be printed
while n > 0:
    print ('Hello ')
    #n -= 1 #This is the same as n = n-1
```

Output:

The word Hello keeps printing on IDLE and the program execution will not end unless we stop it by closing IDLE.

If an update statement is not given in a while block, we will have an **infinite loop**. An infinite loop keeps running until we close the program.

Example 2: Accept a number n and print numbers 1 to n as output.

```
#Accept a number n and print 1 to n
i = 1 #First number to be printed is 1
n = int(input('Enter a number greater than or equal to 1 ')
        while i <= n:
        print (i)
        i += 1 #Move to the next number
```

Output:

```
Enter a number greater than or equal to 1 9
1
2
3
4
5
6
7
8
9
```

Example 3: Accept number n1 and number n2. Print the first ‘n2’ multiples of ‘n1’. If n1 = 5 and n2 = 3, then the program needs to print the first 3 multiples of 5.

```
#print n2 multiples of a number, n1 using while loop
n1 = int(input('Enter a number '))
n2 = int(input('Enter number of multiples '))
i = 1
while i <= n2:
    print ('Multiple', i, 'is', n1*i)
    i += 1 #update statement
```

Output:

```
Enter a number 6
Enter number of multiples 7
Multiple 1 is 6
Multiple 2 is 12
Multiple 3 is 18
Multiple 4 is 24
Multiple 5 is 30
Multiple 6 is 36
Multiple 7 is 42
```

D.5 Practice Exercises for While Loop

1. Write a program that simulates a card game. The user inputs a card number from 1 to 13. If the input number is 13, the user does not get another card. If the input number is negative, then the player does not get another card.
2. Write a program that keeps accepting the age of employees in a loop. If the age entered is negative or above 57, then the loop should stop executing. The program should also print the number of employees who are in the age group 25 to 35.
3. Write a program that calculates the factorial of a number using a while loop. (Factorial of 4 is 1*2*3*4, Factorial of 6 is 1 *2*3*4*5*6).

Exercises

A. Correct the errors in the given programs:

1.
```
#program to print pi upto 5 decimal places
p = 22/7
print round(p, 2)
```

2.
```
#Program to check if given string has more than 5 characters
s = int('Enter a string '))
if length(s) < 5:
   print ('Yes')
else:
   print (No)
```

3.
```
L = [1,2,3,4,5,6,7,8,9,10]
for i in L:
   print (L[i] * 3)
```

B. Predict the output of the given programs and state the purpose of the program.

1.
```
n1 = int(input('Enter a number '))
n2 = int(input('Enter a second number '))
n3 = int(input('Enter a third number '))
n = max(n1, n2, n3)
if n == n1:
   print( max(n2, n3))
elif n == n2:
   print( max(n1, n3))
else:
   print( max(n1, n2))
```
n1 = 12, n2 = 3, n3 = 5.

2.
```
l = [1,2,3,4,5, 6, 7, 8, 9, 10]
for i in l:
    print i*3
```

3.
```
while False:
        print ('Hello')
```

C. Describe these functions:

1. print()
2. input()
3. int()
4. max()
5. round()

C. Answer the following.

1. When do we need to use IF-ELIF-ELSE block?
2. Explain range() with examples.
3. What is the difference between FOR and WHILE loop?
4. Write the syntax for while loop.