# LAB #01: Introduction to Python & Python Packages

**Lab Objective:**

To introduce students with python programming language and its packages.

**Lab Description:**

**Installation:**

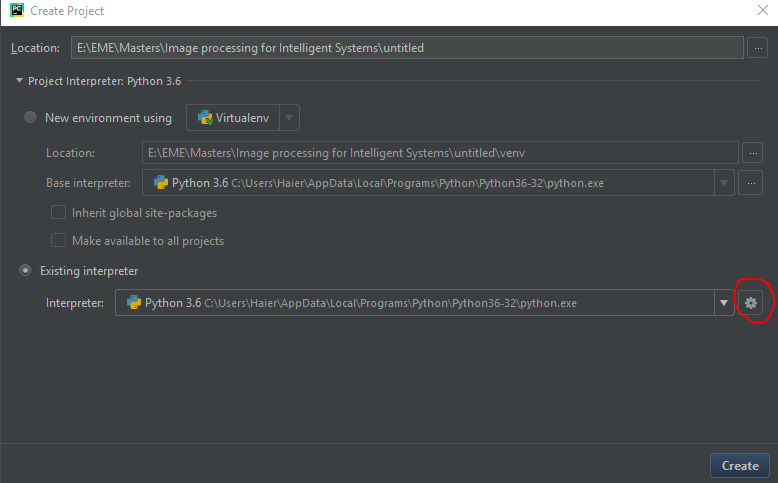
Download and install Python interpreter and PyCharm IDE from the following links below.

Python interpreter: <https://www.python.org/downloads/>

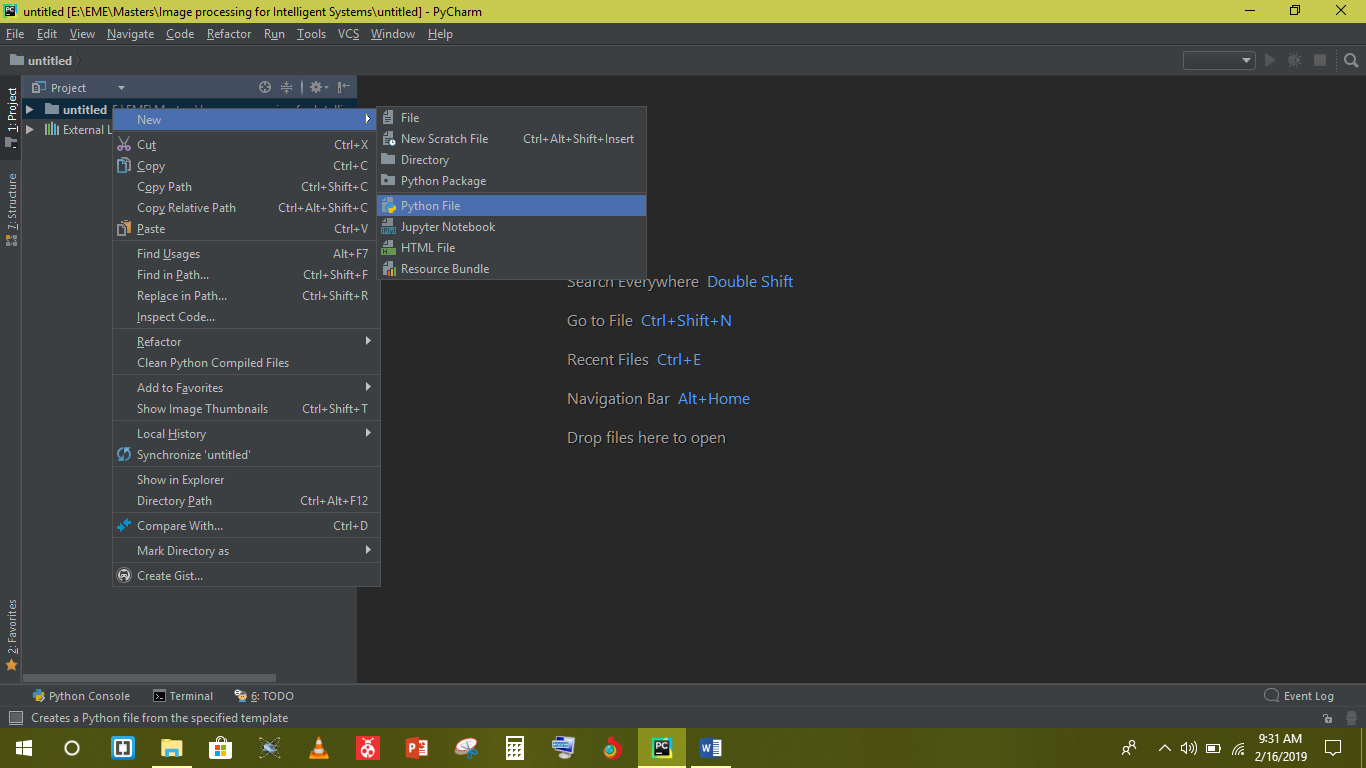
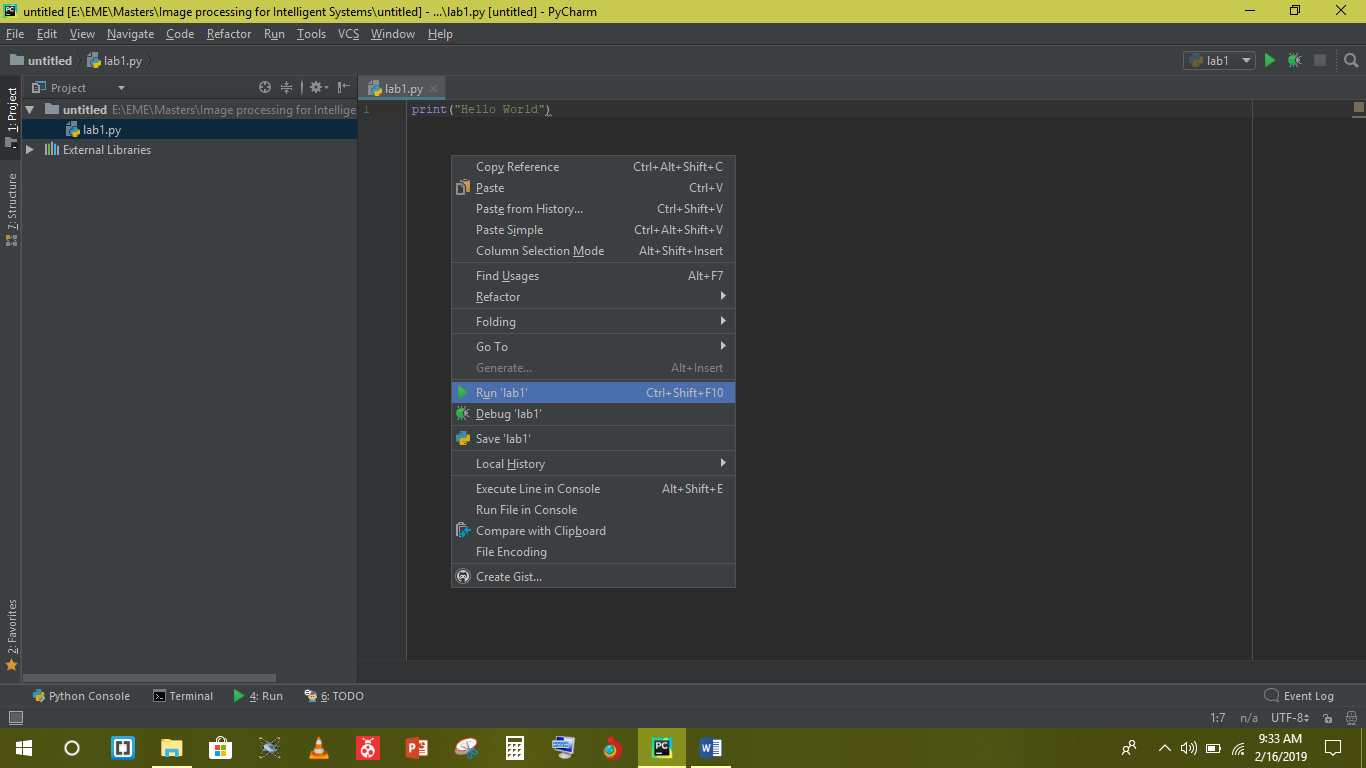
PyCharm IDE: https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=windows&code=PCC

**Setup:**

1. Create ‘New Project’
2. Now Check ‘Existing interpreter’ and Add local python interpreter that you’ve installed, and click Create.



1. Now Right click on the folder and create new python file.



1. After writing your python code right click on the window

and run the project.

**Python** is an interpreted high-level programming language for general-purpose programming. Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons.

|  |  |  |
| --- | --- | --- |
| Operation/Data Structures | Syntax | Explanation / Example |
| Comment | # | # This is a comment. |
| Print  ( is used to display anything in console window) | print( ) | print (10)  print (12+6)  print (5, end=” ”)  print (“END”) |
| Operators | + plus  - minus  \* multiply  \*\* power  / divide  // divide and floor  % modulus | print( 5\*\*2 )  print ( 5//2 )  print (5/2 )  print (5%2) |
| Variables  (Python automatically guess the data type. There is no need to explicitly define data type in python) | x=5  y, z = 3.14, 17.2  temp = None | right side is evaluated first and then assigned to left side with corresponding values  print (x + y)  print(type(x))  print(type(y)) |
| Strings | a=”py”  b=’charm’ | both single and double quotes worksexactly same  print (a + b)  print (a, b)  print (a\*5)  print (“hello’\”world\” ”) |
| Operation/Data Structures | **Syntax** | **Explanation / Example** |
| Lists  List in memory stores references to objects. Each memory location is a pointer to an object.  There is no obligation of similardata types | x=int (input (“Enter a number”)  y = 3.14 z = "HELLO" li = [x, y, z, 4] | input ( ) function always input string, int ( ) function is used to convert string to int  print(li) |
| List Indexing  # From left to right:  0 🡪 1 🡪 2  # From right to left:  -1 🡪 -2 🡪 -3 | Q = li [2] [-4] | print ( Q ) |
| List Slicing  list [start: end: step size]start is inclusive and end is exclusive  defaults values are  list [ 0 : end : 1] | R = li [1:3] | print (li [1:3])  print (li [0:4:2])  print (li [:])  print (li [0:])  print (li [:3])  print (li [2] [1:4]) |
| Tuple  Tuples are similar to lists apart from the fact that they are immutable. | Temp = (23, 45) | print (Temp) |
| Dictionary  Key value pairs for storing specific values against keys. | my\_dictionary = {1:23, 2:46, 3: 69} | print(my\_dictionary) |
| Copy  The assignment copies the reference to the original list while slicing creates a new list | w = [1, 2, 3, 4]  x = w  y = w [:] | x [0] = 6  y [1] = 9  print(w) |
| Indentation | x =2 if x==10:  print("inside”)  print("inside") print("outside") | Whitespace (spaces and tabs) at the beginning of the logical line is used to determine the indentation level of the logical line, which in turn is used to determine the grouping of statements. |
| Boolean operations | < (less than)  > (greater than)  <= (less than/equal to)  >=(greaterthan/equal)  == (equal to)  ! = (not equal to)  not (Boolean NOT) and (Boolean AND)  or (Boolean OR) | if not x:  if x==2 and y>4:  print ( 2==4 ) |
| Operation/Data Structures | **Syntax** | **Explanation / Example** |
| if  If the Boolean expression evaluates to true, then the if block of code will be executed | number = 23  if number == 24:  print (‘equal’)  elif number<24:  print (‘less’)  else:  print (‘greater’) | If statement does not include brackets and ends with colons. The if block is determined by indentation level |
| while  Repeats a statement or group of statements while a given condition is true | number = 23  while number<30:  print(number)  number+=1 |  |
| for  Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable | for i in [2,3,4,1,5]:  print(i)  for j in range(1,9,2):  print(j) | Range function is used to generate a list of numbers, which is generally used to iterate over with for loops |
| Functions  A function is a block of code which only runs when it is called | def max (x, y):  if x > y:  return x  else:  return y  m=max (3,5)  print(m) |  |

**Installing Python Packages:** Packages can be installed using pip command.

For installing a package, the following command can be used: pip install package

* opencv-python
* numpy // Used for numerical operations
* matplotlib // Used to plotting graphs

## Lab Tasks:

## 1: Given the following input list, write a function that can sort the given list in ascending or descending order based on the user input. The fields are as follows:

sub\_list = [Age, CGPA, city]. An example of the input list would be: my\_input\_list = [[29, 3.2, 'Rawalpindi'], [22, 4.0, Islamabad], [12, 0, 'Karachi']].

You need to write a function sorter that takes 3 inputs (the list, sorting criterion and ascending/descending order) as follows:

def sorter (my\_input\_list, 'Age', 'Descending'):

"""

"""

return sorted\_list

## 2: Write a program that takes an integer as an input from the user and represents it in binary, octal or hexadecimal. The user should specify the output number system i.e. whether the number should be converted to binary, octal or hexadecimal.

**3:** One method of encoding messages is known as the "expanding square code". This method encodes a message by placing the character of the message in an odd order square matrix row by row, and then retrieving them in a clockwise expanding square spiral from the centre of the matrix. If the message is not exactly the right length to fill the matrix, the rest of the matrix is filled with asterisk characters (\*). For example, the two square matrices below show the order in which the characters are placed in the matrix, and the order in which they are retrieved. Notice that the order of retrieval begins at the centre, then proceed to the right, and then spirals clockwise.

Order In

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |

Order Out

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 21 | 22 | 23 | 24 | 25 |
| 20 | 7 | 8 | 9 | 10 |
| 19 | 6 | 1 | 2 | 11 |
| 18 | 5 | 4 | 3 | 12 |
| 17 | 16 | 15 | 14 | 13 |

Message In: "This is a test message!!"

Message Out: "stssees a a\*!!egmtiThis "

Your program must be able both to encode and to decode messages by this method. A few things to keep in mind are:

1. There is no defined length of the message.

**4:** Morse code is a sequence of dashes and dots that can be used to represent a message in English. Each letter in a particular alphabet is represented by a unique pattern of dashes and dots in Morse code. You can find more details about the Morse code on Wikipedia.

For this task, your job is to create a code that can convert an English string into Morse code. Your function should return the Morse Code which can then be displayed.

Your code should also be able to convert Morse code into English text. (You may not be able to do it directly. Look up Quinary Form of Morse Code.)

You can use text files to read the mappings from and to English.

**5:** John loves to find words in a 2D puzzle of letters but isn’t very good at it. In this task, you can write a function that helps him do so.

The program starts with a 2D grid of 9x9 with randomly generated letters. Then John will type in a letter to check whether it exists in the grid or not. As an example:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | F | M | B | N | D | L | O | D |
| C | F | J | K | W | I | T | C | I |
| D | F | C | O | L | L | E | G | E |
| G | K | M | R | F | D | I | B | O |
| F | C | F | L | J | I | W | Q | C |
| F | I | A | N | C | U | O | L | F |
| Q | R | O | T | R | B | C | M | I |
| I | L | I | E | W | Q | T | N | X |
| S | L | A | F | J | L | I | K | E |

Now, if the John enters CAT, then your program should be able to tell whether it exits in the puzzle or not. Similarly, if COLLEGE is entered then the program should do the same.

The word can either exist in a straight line (both vertical and horizontal).

6. A garland word is one that starts and ends with the same N letters in the same order, for some N greater than 0, but less than the length of the word. I'll call the maximum N for which this works the garland word's degree. For instance, "onion" is a garland word of degree 2, because its first 2 letters "on" are the same as its last 2 letters. The name "garland word" comes from the fact that you can make chains of the word in this manner:

onionionionionionionionionionion...

Today's challenge is to write a function garland that, given a lowercase word, returns the degree of the word if it's a garland word, and 0 otherwise.

Examples

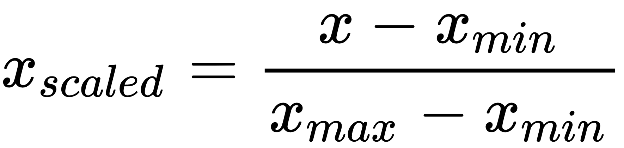
garland("programmer") -> 0

garland("ceramic") -> 1

garland("onion") -> 2

garland("alfalfa") -> 4

7. Min-Max normalization is one of the normalization techniques that can be used for numerical data. It is performed using the following operation:



Write a function min\_max\_normlaization that takes a list as input and performs the min-max normalization on it. The length of the list can vary.

8. Write a function that takes a 9x9 “array” with randomly generated numbers and returns the sum of each 3x3 non-overlapping grid. For example, if the following grid is entered as input,

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 |
| 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 |
| 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 |
| 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 |

then the outputs would be: 99, 117, 153, 342, …, 639.