



JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA
KAKINADA – 533 003, Andhra Pradesh, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

I Year - II Semester		L	T	P	C
		0	0	3	1.5
PYTHON PROGRAMMING LAB					

Course Objectives:

The Objectives of Python Programming are

- To learn about Python programming language syntax, semantics, and the runtime environment
- To be familiarized with universal computer programming concepts like data types, containers
- To be familiarized with general computer programming concepts like conditional execution, loops & functions
- To be familiarized with general coding techniques and object-oriented programming

Course Outcomes:

- Develop essential programming skills in computer programming concepts like data types, containers
- Apply the basics of programming in the Python language
- Solve coding tasks related conditional execution, loops
- Solve coding tasks related to the fundamental notions and techniques used in object-oriented programming

List of Experiments:

- 1) Write a program that asks the user for a weight in kilograms and converts it to pounds. There are 2.2 pounds in a kilogram.
- 2) Write a program that asks the user to enter three numbers (use three separate input statements). Create variables called total and average that hold the sum and average of the three numbers and print out the values of total and average.
- 3) Write a program that uses a *for* loop to print the numbers 8, 11, 14, 17, 20, . . . , 83, 86, 89.
- 4) Write a program that asks the user for their name and how many times to print it. The program should print out the user's name the specified number of times.
- 5) Use a *for* loop to print a triangle like the one below. Allow the user to specify how high the triangle should be.


```

      *
      **
      ***
      ****
      
```
- 6) Generate a random number between 1 and 10. Ask the user to guess the number and print a message based on whether they get it right or not.
- 7) Write a program that asks the user for two numbers and prints *Close* if the numbers are within .001 of each other and *Not close* otherwise.
- 8) Write a program that asks the user to enter a word and prints out whether that word contains any vowels.
- 9) Write a program that asks the user to enter two strings of the same length. The program should then check to see if the strings are of the same length. If they are not, the program should print an appropriate message and exit. If they are of the same



JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA
KAKINADA – 533 003, Andhra Pradesh, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

length, the program should alternate the characters of the two strings. For example, if the user enters *abcde* and *ABCDE* the program should print out *AaBbCcDdEe*.

- 10) Write a program that asks the user for a large integer and inserts commas into it according to the standard American convention for commas in large numbers. For instance, if the user enters 1000000, the output should be 1,000,000.
- 11) In algebraic expressions, the symbol for multiplication is often left out, as in $3x+4y$ or $3(x+5)$. Computers prefer those expressions to include the multiplication symbol, like $3*x+4*y$ or $3*(x+5)$. Write a program that asks the user for an algebraic expression and then inserts multiplication symbols where appropriate.
- 12) Write a program that generates a list of 20 random numbers between 1 and 100.
 - (a) Print the list.
 - (b) Print the average of the elements in the list.
 - (c) Print the largest and smallest values in the list.
 - (d) Print the second largest and second smallest entries in the list
 - (e) Print how many even numbers are in the list.
- 13) Write a program that asks the user for an integer and creates a list that consists of the factors of that integer.
- 14) Write a program that generates 100 random integers that are either 0 or 1. Then find the longest run of zeros, the largest number of zeros in a row. For instance, the longest run of zeros in $[1,0,1,1,0,0,0,1,0,0]$ is 4.
- 15) Write a program that removes any repeated items from a list so that each item appears at most once. For instance, the list $[1,1,2,3,4,3,0,0]$ would become $[1,2,3,4,0]$.
- 16) Write a program that asks the user to enter a length in feet. The program should then give the user the option to convert from feet into inches, yards, miles, millimeters, centimeters, meters, or kilometers. Say if the user enters a 1, then the program converts to inches, if they enter a 2, then the program converts to yards, etc. While this can be done with if statements, it is much shorter with lists and it is also easier to add new conversions if you use lists.
- 17) Write a function called *sum_digits* that is given an integer num and returns the sum of the digits of num.
- 18) Write a function called *first_diff* that is given two strings and returns the first location in which the strings differ. If the strings are identical, it should return -1.
- 19) Write a function called *number_of_factors* that takes an integer and returns how many factors the number has.
- 20) Write a function called *is_sorted* that is given a list and returns True if the list is sorted and False otherwise.
- 21) Write a function called *root* that is given a number x and an integer n and returns $x^{1/n}$. In the function definition, set the default value of n to 2.
- 22) Write a function called *primes* that is given a number n and returns a list of the first n primes. Let the default value of n be 100.
- 23) Write a function called *merge* that takes two already sorted lists of possibly different lengths, and merges them into a single sorted list.
 - (a) Do this using the sort method. (b) Do this without using the sort method.
- 24) Write a program that asks the user for a word and finds all the smaller words that can be made from the letters of that word. The number of occurrences of a letter in a smaller word can't exceed the number of occurrences of the letter in the user's word.
- 25) Write a program that reads a file consisting of email addresses, each on its own line. Your program should print out a string consisting of those email addresses separated by semicolons.



JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA
KAKINADA – 533 003, Andhra Pradesh, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

- 26) Write a program that reads a list of temperatures from a file called *temps.txt*, converts those temperatures to Fahrenheit, and writes the results to a file called *ftemps.txt*.
- 27) Write a class called *Product*. The class should have fields called *name*, *amount*, and *price*, holding the product's name, the number of items of that product in stock, and the regular price of the product. There should be a method *get_price* that receives the number of items to be bought and returns a the cost of buying that many items, where the regular price is charged for orders of less than 10 items, a 10% discount is applied for orders of between 10 and 99 items, and a 20% discount is applied for orders of 100 or more items. There should also be a method called *make_purchase* that receives the number of items to be bought and decreases amount by that much.
- 28) Write a class called *Time* whose only field is a time in seconds. It should have a method called *convert_to_minutes* that returns a string of minutes and seconds formatted as in the following example: if seconds is 230, the method should return '5:50'. It should also have a method called *convert_to_hours* that returns a string of hours, minutes, and seconds formatted analogously to the previous method.
- 29) Write a class called *Converter*. The user will pass a length and a unit when declaring an object from the class—for example, *c = Converter(9,'inches')*. The possible units are inches, feet, yards, miles, kilometers, meters, centimeters, and millimeters. For each of these units there should be a method that returns the length converted into those units. For example, using the *Converter* object created above, the user could call *c.feet()* and should get 0.75 as the result.
- 30) Write a Python class to implement *pow(x, n)*.
- 31) Write a Python class to reverse a string word by word.
- 32) Write a program that opens a file dialog that allows you to select a text file. The program then displays the contents of the file in a textbox.
- 33) Write a program to demonstrate Try/except/else.
- 34) Write a program to demonstrate try/finally and with/as.