|  |  |  |
| --- | --- | --- |
| A picture containing text, clipart  Description automatically generated | **CS-141L INTRODUCTION TO COMPUTING LAB**  **Assignment P1** | Instructor: Drakhshan Bokhat  Total Marks: 10  Marks Obtained: \_\_\_\_\_\_\_\_ |
| Roll No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Mapping CLOs: CLO3** |

**Python Lab 1a: The print function, Unicode values**

**Before you do this lab, you should review Section 1.8 in the text.**

|  |  |
| --- | --- |
| 1. Print your name to the console. | In Python, the print function is used to print text-based information to the console. A typical function call would look like this:  **print ('Welcome to my program! ')**  Use this syntax to print out your name. Use Quotations marks. |
| 2. Print out your city of birth. | We could print out all of our program's output on a single line, but that would not be very easy to read. It’s typical to have a separate print () function call for each thing we want to print out.  After your first line of code (from above), print out your city of birth. |
| 3. Print the results of mathematical expressions. | When you want Python to print something that is not a literal sequence of characters, you do not use the quotation marks.  **print (3 \* 4 \* 5)**  Add another line of code to print out the number of seconds in the current month. Have Python do the multiplication, using the example as a guide. |
| 4. Print the results of integer division and remainder. | Use the // operator for integer division and % for the integer remainder.  **print (44 // 3)**  **print (44 % 3)**  Add code to calculate how many full dozens are made by 403 eggs, and how many eggs you have remaining. |
| 4. Place variables inside a print() function. | You can assign a value to a variable. Here are two examples, one that is an integer and one that is a string.  **age = 20**  **job\_title = 'cashier'**  **print (age)**  **print (job\_title)**  Think of other variables. Give them meaningful names. Use the = operator to store values in the variables. Print out the value of the variables. |
| 5. Print out different Unicode values and their names. | Each character in Unicode can be identified by a hexadecimal digit.  Do a search on the web for a Unicode table.  **print ('\u03bb' + ' Greek small letter lambda')**  Print out 6 different Unicode values that you have never seen before. Use the + operator to concatenate the value and the name of the character. |

**Python Lab 1b: Binary, Decimal, Hex**

**Before you do this lab, you should review Section 1.8 in the text.**

|  |  |
| --- | --- |
| 1. Print using the comma to separate inputs. | The print function, like other functions in Python, takes inputs. We can give the print function multiple inputs which are separated by a comma. In the console, the output is separated by a space.  **print (21, 4\*32, 'hello')** |
| 2. Print out the value of a variable in binary and decimal form. | The bin() and hex() functions will return the binary and hexadecimal representations of a variable.  **x = 31**  **print (x, bin(x), hex(x)**) |
| 3. Identify the error when giving an incorrect input type. | Change the value of x to  **x = 1.825**  What error did you get? Why is this?  Change the value of x to  **x = 18** |
| 4. Assigning a binary or hex value to a variable. | You can assign binary or hex values to a variable.  **y = 0b1011**  **z = 0xA3**  **print (y, z)** |
| 5. You can add numbers in any representation. | You can add variables in any representation.  **w = x + y + z**  **print ('the sum is ' , w)** |
| 7. If Time | The comma in a print function separates inputs. There are ways to make the comma separate in a different way. Investigate this and try out different ways to use the comma in a print statement. |

**Python Lab 1c: Calculating Compression Rate**

**Before you do this lab, you should review Section 1.8 in the text.**

In Activity 1c, you learned about lossless compression. In this lab, you will write a program which calculates the percent of compression.

|  |  |
| --- | --- |
| 1. Choose meaningful variables. | This program will calculate the percent of compression given the formula:  Percent of compression = (Compressed Text Size + Dictionary size) divided by original size. Subtract this value from 100%.  **original\_size = 240**  **dictionary\_size = 5** |
| 2. Assign values to your variables. Calculate the result. | Imagine you have an original text size of 240 characters, a compressed text size of 148 characters, and a dictionary size of 5 characters. The percent of compression should be 0.2792, or 27.92%. |
| 3. Add comments. | At the top of your code, add comments for your name.  Add a comment explaining what the program does.  If your code has some complicated statements, add comments explaining what they do. |
| 4 Print out your result. | Print your results. Make your output similar to the following.  **Compressed text size: 148 characters  Dictionary size: 25 characters  Total: 173 characters  Original text size: 240 characters  Compression: 27.92%**  Use spaces before the words to make the semicolons line up.  Include spaces around numbers.  **print (' Dictionary size:', str(d\_size), 'characters')** |
| 5. Double-check your program. | Test your program with different inputs to make sure that it works with many different possible inputs. Did you remember to print out the percent sign in the compression? |

**Python Lab 1d: Run Length Encoding a picture**

**Before you do this lab, you should review Section 1.8 in the text**

|  |  |
| --- | --- |
| 1. Concatenate and Replicate. | Strings in Python can be concatenated with the + operator.  **print ('@' + '.' + '')**  and replicated with the \* operator  **print ('@@@' \* 3 + '...)**  Write a single print statement that prints out  **&&&&..&&&&..&&&&..&&&&..&&&&..+++++**  Using replication and concatenation. |
| 2. Make a grid picture.  Fill certain squares to make a picture. | In the 10x10 grid below, color in squares to make a picture that represents your two initials, or an emoji, or a picture.   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |
| 3. Print out a textual representation of the picture using concatenate and replicate operators. | Using 10 print() commands, make the picture in the console.  Use one character (such as '@') for a filled in square, and another character (such as '.') for empty squares.  Use the concatenate and replicate operators. |