**ASSIGNMENT 1 SOLUTION**

**[1]    \*         à  Expression**

**‘hello’  à  Value**

**-87.8    à   Value**

**-          à   Expression**

**/          à   Expression**

**+         à   Expression**

**6         à    Value**

**[2]   String is collections of characters inside the single or double quotes. We can take input in string.**

**There is no rule for string value we can take numbers, alphabets, etc… inside single/double quotes. But for Variable, there is some rules like we can not start to declare a variable name with  numbers, Special characters except underscore( \_ ).**

**We can store a string value into a variable.**

**E.g; name = “Awanish”   à  name is a variable name and “Awanish” is string value.**

**[3]    datatypes are à  integer, string and Boolean.**

**Integer :-  int()  by this method we can convert numerical string value into Integer. It contains               positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.**

**e.g;      n = 25**

**String :-   str() by this method we can convert any input into String datatype. String is collections of characters inside the single or double quotes.**

**e.g;      name = “Awanish”**

**Boolean :- bool() by this method we can convert any input into Boolean value like True, False. It will give out**

**e.g;      a = 10;**

**print(bool(a))**

**[4]    An expression is a combination of operators and operands that is interpreted to produce some other value. In any programming language, an expression is evaluated as per the precedence of its operators. So that if there is more than one operator in an expression, their precedence decides which operation will be performed first. We have many different types of expressions in Python.**

**Constant Expressions: These are the expressions that have constant values only.**

**x = 15 + 1.3;  print(x)**

**Arithmetic Expressions: An arithmetic expression is a combination of numeric values, operators, and sometimes parenthesis. The result of this type of expression is also a numeric value. The operators used in these expressions are arithmetic operators like addition, subtraction, etc…**

**x = 40**

**y = 12**

**add = x + y**

**sub = x - y**

**pro = x \* y**

**div = x / y**

**print(add)**

**print(sub)**

**print(pro)**

**print(div)**

**Integral Expressions: These are the kind of expressions that produce only integer results after all computations and type conversions.**

**a = 13**

**b = 12.0**

**c = a + int(b)**

**print(c)**

**Floating Expressions: These are the kind of expressions which produce floating point numbers as result after all computations and type conversions.**

**a = 13**

**b = 5**

**c = a / b**

**print(c)**

**[5]   A statement is a group of expressions and/or statements that you design to carry out a task or an action.**

**Spam = 10  à  this is statement**

**Expressions are values or execute to values.**

**Spam = 10  à  spam, =, 10 all three a expressions.**

**[6]   bacon = 22**

**bacon + 1**

**After running this above code, value of bacon will be 22 because in 2nd line bacon + 1 we are not storing in bacon variable.**

**[7]   ‘spam’ + ‘spamspam’  = ‘spamspamspam’  à here concatenation happen.**

**‘spam’ \* 3 = ‘spamspamspam’**

1. **Variable name can only start with alphabets or underscore( \_ ) that’s why Variable name eggs is valid        but variable name 100 is not valid.**

**[9]  int()  à used for integer conversion we can convert any string numeric value into integer.**

**E.g; print(int(“3”)) output:- 3**

**float()  à  used for conversion in float/decimal number.  E.g; print(float(3)) output:- 3.0**

**str()  à  used to convert any number or any input into string datatype.**

1. **g;  age = str(23)  print(type(age)) à  <class 'str'**

**[10]  ‘I have eaten’ + 99 + ‘burritos.’  will Give error because we can concatenate number with string we need to convert that data type into string also.**

**‘I have eaten’ + str(99) + ‘burritos’  or   ‘ I have eaten’ + ‘99’ + ‘burritos’  both of them will give same output.**