# Week 1 – Homework 1 # Python #

1. Is the literal 4 a valid python expression?

Yes. A literal like 4 is example of simple expressions.

2. Is the variable x a valid python expression?

Yes. A variable like x is example of simple expressions.

3. Is the x + 4 a valid python expression?

Yes. If x declared before, x + 4 is called a simple expression.

4. What affect does the unary + operator have when applied to a numeric expression?

In contrast, the unary operator can be applied to any type of argument. The plus symbol does not affect the type or value and is simply negative value, if applied to a numeric value.

5. Sort the following binary operators in order of high to low precedence:

+ , - , \* , // , / , % , =

1– Unary positive, Unary negative means: + , - , +y , -y

2- Multiplication, Division, Floor Division, Modulus: \* , / , // , %

3- Addition, Subtraction: + , -

6. Given the following assignment:

x = 2

* Indicate what each of the following python

statements would print.

1. print (“x”)

x

1. print (‘x’)

x

1. print (x)

2

1. print (“x + 1”)

x + 1

1. print (‘x’ + 1)

Type Error: can only concatenate str (not "int") to str.

1. print (x + 1)

3

7. Given the following assignment:

i1 = 2

i2 = 5

i3 = -3

d1 = 2.0

d2 = 5.0

d3 = -0.5

* Evaluate each of the following Python expressions.

(a) i1 + i2 = 7

(b) i1 / i2 = 0.4

(c) i1 // i2 = 0

(d) i2 / i1 = 2.5

(e) i2 // i1 = 2

(f) i1 \* i3 = -6

(g) d1 + d2 = 7.0

(h) d1 / d2 = 0.4

(i) d2 / d1 = 2.5

(j) d3 \* d1 = -0.1

(k) d1 + i2 = 0.7

(l) i1 / d2 = 0.4

(m) d2 / i1 = 2.5

(n) i2 / d1 = 2.5

(o) i1/i2\*d1 = 0.8

(p) d1\*i1/i2 = 0.8

(q) d1/d2\*i1 = 0.8

(r) i1\*d1/d2 = 0.8

(s) i2/i1\*d1 = 5.0

(t) d1\*i2/i1 = 5.0

(u) d2/d1\*i1 = 5.0

(v) i1\*d2/d1 = 5.0

8. What is printed by the following statement:

# print (5/3)

print (5/3)

1.6666666666666667

9. Given the following assignments:

i1 = 2

i2 = 5

i3 = -3

d1 = 2.0

d2 = 5.0

d3 = -0.5

* Evaluate each of the following Python expressions.

(a) i1 + (i2 \* i3) = -13

(b) i1 \* (i2 + i3) = 4

(c) i1 / (i2 + i3) = 1.0

(d) i1 // (i2 + i3) = 1

(e) i1 / i2 + i3 = -2.6

(f) i1 // i2 + i3 = -3

(g) 3 + 4 + 5 / 3 = 8.666666666666666

(h) 3 + 4 + 5 // 3 = 8

(i) (3 + 4 + 5) / 3 = 4.0

(j) (3 + 4 + 5) // 3 = 4

(k) d1 + (d2 \* d3) = -0.5

(l) d1 + d2 \* d3 = -0.5

(m) d1 / d2 - d3 = 0.9

(n) d1 / (d2 - d3) = 0.36363636363636365

(o) d1 + d2 + d3 / 3 = 6.833333333333333

(p) (d1 + d2 + d3) / 3 = 2.1666666666666665

(q) d1 + d2 + (d3 / 3) = 6.833333333333333

(r) 3 \* (d1 + d2) \* (d1 - d3) = 52.5

10. What symbol signifies the beginning of a comment in Python?

Comments in python begin with a hash mark (#) and continues until the end of that line.

11. How do Python comments end?

We can comment line by # and also we can do this

""" This is comment """

12. Which is better, too many comments or too few comments?

A comment is basically a text note that gives an explanation about the source code. Furthermore, they act as documentation in the source code. We include comments to increase the readability of the program. Besides, comments make it easy for the programmer to remember the complex things added to the code. So comments are important, and enough or too many comments are better choice.

13. What is the purpose of comments?

Comments can be used to explain code, and can be used to make the code more readable, and can be used to prevent execution when testing code.

14. Why is human readability such an important consideration?

Because understanding codes in program, that how it works.

15. What circumstances can cause each of the following run-time errors to arise?

• NameError

Explanation: NameError occurs when you try to use a variable, function, or module that doesn't exist or wasn't used in a valid way. Some of the common mistakes that cause this error are: Using a variable or function name that is yet to be defined.

• ValueError

Explanation: ValueError is an exception that occurs when a function receives an argument of the correct data type but an inappropriate value. This error usually occurs in mathematical operations that require a certain kind of value.

• ZeroDivisionError

Explanation: ZeroDivisionError is a built-in Python exception thrown when a number is divided by 0. This means that the exception raised when the second argument of a division or modulo operation is zero. In Mathematics, when a number is divided by a zero, the result is an infinite number.

• IndentationError

Explanation: The cause of IndentationError in Python Since python makes use of procedural language, if you miss out on adding tabs or spaces between your lines of code, then you will most likely experience this error.

• OverflowError

Explanation: An OverflowError exception is raised when an arithmetic operation exceeds the limits to be represented. This is part of the ArithmeticError Exception class.

• SyntaxError

Explanation: SyntaxError occurs when the interpreter encounters invalid syntax in code. When Python code is executed, the interpreter parses it to convert it into bytecode. If the interpreter finds any invalid syntax during the parsing stage, a SyntaxError is thrown.

• TypeError

Explanation: TypeError is an exception in Python programming language that occurs when the data type of objects in an operation is inappropriate. For example, If you attempt to divide an integer with a string, the data types of the integer and the string object will not be compatible.

Hint: Try some of following activities in the interpreter or within a Python program:

• print a variable that has not been assigned (NameError)

• convert the string 'two' to an integer (ValueError)

• add an integer to a string (TypeError)

• assign to a variable named end-point (syntaxerror)

• experiment adding spaces and tabs at various places in the code of an error-free Python program (IndentationError)

• compute raise a floating-point number to a large power, as in 1.5 10,00 (OverflowError)

16. Consider the following program which contains some errors. You may assume that the comments within the program accurately describe the program’s intended behavior.

# Get two numbers from the user

n1 = float(input()) # 1

n2 = float(input()) # 2

# Compute sum of the two numbers

print(n1 + n2) # 3

# Compute average of the two numbers

print(n1+n2/2) # 4

# Assign some variables

d1 = d2 = 0 # 5

# Compute a quotient

print(n1/d1) # 6

Error: ZeroDivisionError: float division by zero in line 6.

# Compute a product

n1\*n2 = d1 # 7

Error: To place an expression that consists of a set of variables and operators, they must be placed on the right side of equality and the desired variable on the left side of equality, so that no error occurs in the program and this placement is done. This means that the value or expression is considered in the variable. In general; moving the sides of the equality in line 7 of this program solves the error. Like this: d1 = n1\*n2

# Print result

print(d1) # 8

Now it works (print) properly.

For each line listed in the comments, indicate whether or not an interpreter error, run-time exception, or logic error is present. Not all lines contain an error.

It’s Done.

17. Write the shortest way to express each of the following statements.

(a) x = x + 1 x +=1

(b) x = x / 2 x /= 2

(c) x = x - 1 x -= 1

(d) x = x + y x += y

(e) x = x - (y + 7) x -= (y+7)

(f) x = 2\*x x \*= 2

(g) number\_of\_closed\_cases = number\_of\_closed\_cases + 2\*ncc

Number\_of\_closed\_cases += 2\*ncc

18. What is printed by the following code fragment?

x1 = 2

x2 = 2

x1 += 1

x2 -= 1

print(x1)

3

print(x2)

1

Why does the output appear as it does?

x1 = 2 x2 = 2

(2) (2)

x1 = x1 + 1 x2 = x2 - 1

print(x1) print(x2)

3 1

Explanation: Because it works like this, firstly, the number 2 is placed in the two specified variables, and when this number (within the variable) is added to the number 1 and it is placed again in the same name as the previous variable, the value changes as well for the second case. Which was done by subtraction and we know that with the last change of value in the variable, when we print that variable, it will show us the last value (changing).

19. Consider the following program that attempts to compute the circumference of a circle given the radius entered by the user. Given a circle’s radius, r, the circle’s circumference, C is given by the formula:

C = 2πr

r = 0

PI = 3.14159

# Formula for the area of a circle given its radius

C = 2\*PI\*r

# Get the radius from the user

r = float(input("Please enter the circle's radius: "))

# Print the circumference

Print("Circumference is", C)

(a) The program does not produce the intended result. Why? Because the prioritization of the lines is not followed correctly and they must be moved, in fact, in this program, r must be defined before the command C, that is what C = 2\*PI\*r means.

(b) How can it be repaired so that it works correctly?

We can solve this problem by moving lines 2 and 3, firstly the PI number in line 1 is defined, so the radius means, r, should be defined in line 2, then the Circumference means, C, should be defined in line 3.

20. Write a Python program that ...

Consider a program that tries to calculate and print the average of three numbers taken from the user:

P01 in my Github

# Get the three numbers from the user

A = float(input(“Enter first number: “))  
B = float(input(“Enter second number: “))  
C = float(input(“Enter third number: “))

# Formula for the average of three numbers  
Average = (A+B+C)/3

# Print the average  
print(“Result of Average: “, Average)

21. Write a Python program that ...

Consider a program that tries to get the coefficients of a quadratic equation from the user and calculate and print its roots:

P02 in my Github

# Library for defining square delta

import math

# Get the three values for Coefficients from the user   
a = int(input("Enter fist value: "))  
b = int(input("Enter second value: "))  
c = int(input("Enter third value: "))

# Formula for the delta of three roots  
delta = (b \* b) - (4 \* a \* c)

# Using the conditional if structure to calculate and print the roots

# Two roots  
if(delta > 0):  
 x1 = (-b + math.sqrt(delta))/(2\*a)  
 x2 = (-b - math.sqrt(delta))/(2\*a)  
 print("x1 = ",x1,"\n","x2 = ",x2)

# One root

if(delta ==0):  
 x1 = (-b) / (2\*a)  
 print("x1 = ",x1)

# Without any root  
if(delta < 0):  
 print("Undefined.")