Arithmetic Operators

3

Types of Operator

- Python language supports the following types of operators:
- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Operators

- Arithmetic operators. The arithmetic operators available in C are
- Both '+' and '-' uses the same syntax:

  expr1 + *expr2*

  expr1 - *expr2*

  Legal operand types for *expr1* + *expr2* are:

  Both *expr1* and *expr2* are of arithmetic type

Operators

- **Arithmetic operators.**
- **Multiplicative operators ('*', '/' and '%')**

There are three multiplicative operators in C:

- ***(multiplication: the product of the two operands)**
- **/ (division: the quotient of the first operand divided by the second operand)**
- **% (modulus: the remainder of the first operand divided by the second operand)**
- **They use the following syntax:**

  expr1 * expr2

  expr1 / expr2 **Note expr2 should be non-zero.**

  expr1 % expr2

Operators

Priority of Operators

The hierarchy of commonly used operators is

Priority	Operators	Description
1st	* / %	multiplication, division, modular division
2nd	+ -	addition, subtraction
3rd	=	assignment

Operators

Priority of Commonly used Operators

Example: Determine the hierarchy of operations and evaluate the following expression:

i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8

Stepwise evaluation of this expression is shown below:

- **i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8**
- **i = 6 / 4 + 4 / 4 + 8 - 2 + 5 / 8 operation: ***
- **i = 1 + 4 / 4 + 8 - 2 + 5 / 8 operation: /**
- **i = 1 + 1+ 8 - 2 + 5 / 8 operation: /**
- **i = 1 + 1 + 8 - 2 + 0 operation: /**
- **i = 2 + 8 - 2 + 0 operation: +**
- **i = 10 - 2 + 0 operation: +**
- **i = 8 + 0 operation : -**
- **i = 8 operation: +**

In C int is treated as int

- **Note in C that 6 / 4 gives 1 and not 1.5.**
- **This so happens because 6 and 4 both are integers and therefore would evaluate to only an integer constant.**
- **Similarly 5 / 8 evaluates to zero, since 5 and 8 are integer constants and hence must return an integer value.**

In python int is internally converted to float

- >>> 6/4
- 1.5
- >>> 10/3
- 3.3333333333333335
- >>> 5/8
- 0.625

Convert Arithmetic Statement to C

We should know how to convert a general arithmetic statement to a C statement. Examples:

Algebric Expression	C Expression
$a \times b - c \times d$	a * b – c * d
$(m + n)\ (a + b)$	(m + n) * (a + b)
$3x2 + 2x + 5$	3 * x * x + 2 * x + 5
$\frac{a + b + c}{d + e}$	(a + b + c) / (d + e)
$\left[\frac{2BY}{d+1} - \frac{x}{3(z+y)}\right]$	2 * b * y / (d + 1) – x / 3 * (z + y)

input()

- **The input Function.**
- **There are hardly any programs without any input.**
- **Input can come in various ways, for example**
- **from a database,**
- **another computer,**
- **mouse clicks**
- **mouse clicks and movements**
- **or from the internet. ...**
- **For this purpose, Python provides the function input().**
- **Input() has an optional parameter, which is the prompt string.**

input()

- >>> X = input() # input() returns string
- 34
- >>> print(x)
- 34
- >>> x+6
- Traceback (most recent call last):
- File "<pyshell#2>", line 1, in <module>
- x+6
- TypeError: Can't convert 'int' object to str implicitly

Input function

- X= int(input()) # Typecasting string to integer
- X= int(input(“Enter value to x: ”))
- # we are giving some text in input function

- >>> x = int(input())
- 34
- >>> x
- 34
- >>> x+9
- 43
- >>> x = int(input("Enter value of x: "))
- Enter value of x: 22
- >>> x
- 22

Exercise

- **Convert following arithmetic expressions to python expressions**

Algebric Expression
$a \times b - c \times d$
$(m + n)\ (a + b)$
$3x2 + 2x + 5$
$\frac{a + b + c}{d + e}$

Exercise

- Write a program which uses multiple assignments to a,b,c,d with values 40 30 20 10 and also m & n values 5 4. Use arithmetic operators to obtain output

Algebric Expression
$a \times b - c \times d$
$(m + n)(a + b)$
$3x2 + 2x + 5$
$\frac{a + b + c}{d + e}$

Exercise

- Write a program which uses multiple assignments to a,b,c,d with values 40 30 20 10 and also m & n values 5 4. Use arithmetic operators to obtain output
- a, b, c,d = 40,30,20,10
- print(a*b-c*d)
- m , n = 5, 4
- print((m+n)*(a+b))
- print(3*a*a + 2*a + 5)
- print((a+b+c)/(m+n))

1000
630
4885
5.0

math module

- The **math** module is a standard module in **Python** and is always available. To use**mathematical** functions under this module, you have to **import** the module using**import math**

Different forms of import

- import math #imports math module
- print(math.sqrt(49))
- import math as m
- print(m.sqrt(25))
- from math import sqrt
- #imports a method from math module
- print(sqrt(36))
- from math import sqrt as s
- print(s(4))
- from math import *
- print(sqrt(81))

Exercise 1

- Find the volume of cylinder after assigning value to radius and height.
- Hint: import math for getting constant pi by using
- math.pi

import math

- import math
- r = int(input("Enter Radius: "))
- h = int(input("Enter Height: "))
- print("Volume of Cylinder ",math.pi*r**2*h)

Exercise 2

- Assign values of a, b and c. Find the roots of quadratic equation.
- Use pow and sqrt function
- Hint: import math for sqrt()

Solution Exercise 2

- Import math
- >>> a = int(input())
- 1
- >>> b = int(input())
- 2
- >>> c = int(input())
- 1
- >>> -b + (math.sqrt(pow(b,2) - 4*a*c)/2*a)
- -2.0
- >>> -b - (math.sqrt(pow(b,2) - 4*a*c)/2*a)
- -2.0

- >>> a=10
- >>> b=20
- >>> (a+b)**2
- 900
- >>> (a+b)*(a+b)
- 900
- >>> (a+b)**2 == (a+b)*(a+b)
- True
- >>> (a-b)**2
- 100

Introduction to Python – 3.4

- How to exit python interpreter?
- quit() or
- Ctrl + z
- Python is case sensitive quit() IS NOT QUIT() or Quit()

Introduction to Python – 3.4

- Multiline Comment
- You can use triple-quoted strings.
- >>>''' This is
- …a
- …multiline comment. '''‘
- You can also use triple-Double Quotes
- >>>”””” This is
- …a
- …multiline comment. “””“

- Single Line comment
- >>>#this is single line comment
- …

Introduction to Python – 3.4

- >>> # this is the first comment
- ... spam = 1 # and this is the second comment
- >>> # ... and now a third!
- ... text = "# This is not a comment because it's inside quotes."
- >>> print(spam)
- 1
- >>> print(text)
- # This is not a comment because it's inside quotes.
- >>>

Introduction to Python – 3.4

- # note the differing behavior of the punctuation marks comma , and semicolon ;
- >>> 2, 3
- (2, 3) **# Comma-separated sequences are enclosed into a tuple, and displayed.**
- >>> 2; 3
- 2 # Semi-colon separated sequences are displayed on separate lines.
- 3
- #Try output as
- (2, 3, 4) # Execute this life first
- (5, 6, 7) # Execute this life first
- >>> 2, 3, 4; 5, 6, 7
- (2, 3, 4)
- (5, 6, 7)

Python – Data Types

- Commonly used data types include
- *int*,
- *float*,
- *str* and
- *bool*.
- To determine the data type of a variable:
- **>>>** a = 3
- **>>>** type(a)
- <type 'int'>

Introduction to Python – 3.4

- # Note the way the built-in function "type" is used:
- >>> type(2)
- <type 'int'>
- # Try to get following output
- <type 'int'>
- <type 'float'>
- >>> type(2); type(2.0);
- # Try
- <type 'bool'>
- <type 'tuple'>
- Solution:
- >>> type(2<3); type((2, 3, 4))

Introduction to Python – 3.4

- **# There is no char or character data type in Python.**
- **# There are strings of length 0 or more. A character is a string of length 1.**
- **# Strings are enclosed in single quotes or double quotes.**
- **>>> 'a'**
- **'a'**
- **>>> '1'**
- **'1'**
- **>>> "a"**
- **'a'**
- **>>> "1"**
- **'1'**
- **>>> ';'**
- **';'**
- **>>> 'a'; '1'**
- **'a'**
- **'1'**
- **>>> 'a', '1' #a pair**
- **('a', '1')**
- **>>> "Bhopal, M.P."**
- **'Bhopal, M.P.'**

Introduction to Python – 3.4

- **Exercise 1**
- print('"Isn\'t," she said.')
- **Exercise 2**
- s = 'First line.\nSecond line.' # \n means newline
- print(s)
- print('"Yes," he said.')
- **Exercise 3**
- print ('C:\some\name') # here \n means newline!
- **Output of Ex 1,2 and 3**
- "Isn't," she said.
- First line.
- Second line.
- "Yes," he said.
- >>>

Introduction to Python – 3.4

- Strings can be concatenated (glued together) with the + operator, and repeated with *:
- # String can be concatenated with + and repeated with *
- print(3 * 'Aa' + 'Bat')
- AaAaAaBat
- '''Two or more string literals (i.e. the ones enclosed between quotes) next to each other are automatically concatenated.'''
- print('Py','thon') # GIVES → Py thon
- print('Py' 'thon') #GIVES → Python
- s='KMIT Narayanguda - 500029'
- length = len(s)
- print(length)

Introduction to Python – 3.4

- With Python, it is possible to use the ** operator to calculate powers
- >>> 5 ** 2 # 5 squared
- 25
- >>> 25**.5
- 5.0 #Square Root of 25
- >>> pow(5,2)
- 25
- >>> a=complex(4,5)
- >>> b=complex(6,5)
- >>> a+b
- (10+10j)
- >>> a-b
- (-2+0j)

Introduction to Python – 3.4

- # Strings can be enclosed in pairs of one or three SINGLE or DOUBLE quotes, but not two quotes
- >>> '''abc'''; """abc"""
- 'abc'
- 'abc'
- >>> type('''abc'''); type("""abc""")
- <type 'str'>
- <type 'str'>
- # Strings cannot be enclosed in two single-quotes or two-double quotes
- >>> ''abc''
- SyntaxError: invalid syntax
- >>> ""abc""
- SyntaxError: invalid syntax