Parul® University

Python Programming

Prof. Swasti Patel, Assistant Professor
Information Technology

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-2

Python Data Types

Comments in Python

- Single Line Comments
- Comments are non-executable statements.
- Python compiler nor PVM will execute them.

```
#To find the sum of two numbers  a = 10 # store 10 into variable a
```

Comments in Python

- **When we want to mark several lines as comment, then we can write the previous block of code inside """ (triple double quotes) or ''' (triple single quotes) in the beginning and ending of the block as.**

```
"""
Write a program to find the total marks scored by a
student in the subjects
"""
```

```
'''
Write a program to find the total marks scored by a
student in the subjects
'''
```

Parul® University

DIGITAL LEARNING CONTENT

Comments in Python

- Python supports only single line comments.
- Multiline comments are not available in Python.
- Triple double quotes or triple single quotes are actually not multiline comments but they are regular strings with the exception that they can span multiple lines.
- Use of ’’’ or ”””” are not recommended for comments as they internally occupy memory.

Variables in Python

- In many languages, the concept of variable is connected to memory location.
- In python, a variable is seen as a tag that is tied to some value.
- Variable names in Python can be any length.
- It can consist of uppercase and lowercase letters (A-Z,
- a-z), digits (0-9), and the underscore character (_).
- A restriction is that, a variable name can contain digits, the first character of a variable name cannot be a digit.

Datatypes in Python

- A datatype represents the type of data stored into a variable or memory.
- The datatypes which are already avaiable in Python language are called Built-in datatypes
- The datatypes which are created by the programmers are called User-Defined datatypes.

Built-in Datatypes

- None Type
- Numeric Types
- Sequences
- Sets
- Mappings

None Type

- ‘None’ datatype represents an object that does not contain
- any value.
- In Java, it is called ‘Null’ Object’, but in Python it is called ‘None’ object.
- In Python program, maximum of only one ‘None’ object is
- provided.
- ‘None’ type is used inside a function as a default value of the arguments.
- When calling the function, if no value is passed, then the
- default value will be taken as ‘None’.
- In Boolean expressions, ‘None’ datatype is represents ‘False’

Parul®
University

DIGITAL LEARNING CONTENT

Numeric Type

- int
- float
- complex

Int DataType

- The int datatype represents an integer number.
- An integer number is a number without any decimal point or fraction part.
- int datatype supports both negative and positive integer numbers.
- It can store very large integer numbers conveniently.

```
a = 10
b = -15
```

Float DataType

- The float datatype represents floating point numbers.
- A floating point number is a number that contains a decimal point.

```
#For example : 0.5, -3.4567, 290.08,0.001
num = 123.45

x = 22.55e3
#here the float value is 22.55 x 10^3
```

Bool DataType

- The bool datatype in Python represents
- boolean values.
- There are only two boolean values True or False that can be represented by this datatype.
- Python internally represents True as 1 and
- False as 0.

Bool DataType

- The bool datatype in Python represents
- boolean values.
- There are only two boolean values True or False that can be represented by this datatype.
- Python internally represents True as 1 and
- False as 0.

Print Statement

```
>>>print “Python"   Python
```

```
>>>print   “Python Programming"  Python Programming
```

```
>>> print "Hi, I am Learning Python Programming."  Hi, I am Learning Python Programming.
```

Print Statement

```
>>> print 25
25
```

```
>>> print 2*2  4
```

```
>>> print 2 + 5 + 9
16
```

Print Statement

```
>>> print "Hello,","I","am",“Python"   Hello, I am Python
```

```
>>> print “Python","is",27,"years","old"   Python is 27 years old
```

```
>>> print 1,2,3,4,5,6,7,8,9,0
1 2 3 4 5 6 7 8 9 0
```

String DataType

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator. For example –

```
str = 'Hello World!'
print str # Prints complete string
print str[0] # Prints first character of the string
print str[2:5] # Prints characters starting from 3rd to 5th
print str[2:] # Prints string starting from 3rd character
print str * 2 # Prints string two times
print str + "TEST" # Prints concatenated string
```

String Operations

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric

Parul®
University

DIGITAL LEARNING CONTENT

String Operations

Method	Description
isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Converts the elements of an iterable into a string

String Operations

Method	Description
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string

String Operations

Method	Description
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

Assignment Operations

- Operators are used to perform operations on values and variables. These are the special symbols that carry out arithmetic, logical, bitwise computations. The value the operator operates on is known as Operand.
- Here, we will cover Assignment Operators in Python. So, Assignment Operators are used to assigning values to variables.

Operator	Description	Syntax
=	Assign value of right side of expression to left side operand	x = y + z
+=	Add and Assign: Add right side operand with left side operand and then assign to left operand	a += b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand: True if both operands are equal	a -= b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a *= b

Assignment Operations

Operator	Description	Syntax
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a /= b
%=	Modulus AND: Takes modulus using left and right operands and assign result to left operand	a %= b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a //= b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a **= b
&=	Performs Bitwise AND on operands and assign value to left operand	a &= b

Assignment Operations

Operator	Description	Syntax
|=	Performs Bitwise OR on operands and assign value to left operand	a |= b
^=	Performs Bitwise xOR on operands and assign value to left operand	a ^= b
>>=	Performs Bitwise right shift on operands and assign value to left operand	a >>= b
<<=	Performs Bitwise left shift on operands and assign value to left operand	a <<= b

Parul® University

DIGITAL LEARNING CONTENT

Expressions

- Constant Expressions
- Arithmetic Expressions
- Integral Expressions
- Floating Expressions
- Relational Expressions
- Logical Expressions
- Bitwise Expressions
- Combinational Expressions

Constant Expressions

- These are the expressions that have constant values only.

```
# Constant Expressions
x = 15 + 1.3

print(x)
```

Output

```
16.3
```

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. Here are some arithmetic operators in Python:

Operators	Syntax	Functioning
+	x + y	Addition
-	x - y	Subtraction
*	x * y	Multiplication
/	x / y	Division
//	x // y	Quotient
%	x % y	Remainder
**	x ** y	Exponentiation

Arithmetic Expressions

```
x = 40
y = 12

add = x + y
sub = x - y
pro = x * y
div = x / y

print(add)
print(sub)
print(pro)
print(div)
```

Output

```
52
28
480
3.3333333333333335
```

Integral Expressions

- These are the kind of expressions that produce only integer results after all computations and type conversions.

```
# Integral Expressions
a = 13
b = 12.0

c = a + int(b)
print(c)
```

Output

```
25
```

Floating Expressions

- These are the kind of expressions that produce only These are the kind of expressions which produce floating point numbers as result after all computations and type conversions. integer results after all computations and type conversions.

```
# Floating Expressions
a = 13
b = 5

c = a / b
print(c)
```

Output

```
2.6
```

Relational Expressions

- In these types of expressions, arithmetic expressions are written on both sides of relational operator (> , < , >= , <=). Those arithmetic expressions are evaluated first, and then compared as per relational operator and produce a boolean output in the end. These expressions are also called Boolean expressions.

```
a = 21
b = 13
c = 40
d = 37

p = (a + b) >= (c - d)
print(p)
```

Output

```
True
```

Logical Expressions

- These are kinds of expressions that result in either True or False. It basically specifies one or more conditions. For example, (10 == 9) is a condition if 10 is equal to 9. As we know it is not correct, so it will return False. Studying logical expressions, we also come across some logical operators which can be seen in logical expressions most often. Here are some logical operators in Python:

Operator	Syntax	Functioning
and	P and Q	It returns true if both P and Q are true otherwise returns false
or	P or Q	It returns true if at least one of P and Q is true
not	not P	It returns true if condition P is false

Logical Expressions

```
P = (10 == 9)
Q = (7 > 5)

# Logical Expressions
R = P and Q
S = P or Q
T = not P

print(R)
print(S)
print(T)
```

Output

```
False
True
True
```

Bitwise Expressions

- These are the kind of expressions in which computations are performed at bit level.

```
a = 12

x = a >> 2
y = a << 1

print(x, y)
```

Output

```
3 24
```

Combinational Expressions

- We can also use different types of expressions in a single expression, and that will be termed as combinational expressions.

```
a = 16
b = 12

c = a + (b >> 1)
print(c)
```

Output

```
22
```

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in