

Introduction to Python

What is Python?

- A Python program is a script, which is vaguely a short program that can be executed in Python interactive script.
- A statement is a line of code or command or instruction for computers to perform initialize a number, perform some computations, and do some control flow.
- A Function is a collection of statements that perform an intended action and returns a Value.
- There are two types of statements: Simple and Compound.

Simple Statement: Performs a single simple task. For Ex., Expressions and Assignment Statements.

- Compound Statements: This is a group of statements that spans multiple lines and that control the execution of other statements of that program.
- For example, if, while and function are examples of the compound statement.
- Example: Illustration of swapping of variables in Interactive mode

```
>>> x = 10
```

```
>>> y = 20
```

```
>>> t = x
```

```
>>> x = y
```

```
>>> y = t
```

```
>>> x, y
```

```
(20, 10)
```

```
>>>
```

Differences Between Python 2.x and 3.x

No	Python 2.x	Python 3.x
1.	Old version of Python.	New and updated language
2.	The syntax of the print statement is print "hello"	The syntax of Python 3. x print is changed. Print is a function and it should be print("Hello").
3.	Uses ASCII as default for string representation.	Uses UNICODE (UTF-8) to represent strings. So foreign languages and more characters are supported.
4.	Python 2.x is not forward compatible.	Python 3.x is compatible compared with Python 2.x.
5.	The division is awkward in Python 2.x. Division of 7/2 gives 3.	The division of 7 by 2 results in 3.5; hence, calculations are more predictable.

Elements of Python Language

- The set of valid characters that a programming language can recognize is called the character set of the language. Two types of character sets are
- Source Characters. (Alphabets uppercase and lowercase, underscore, Special characters blank –(+,-, *, /, ^, ~, %, !, &, |, (), {}, [], ?', ;, \), blank(“”))
- The Token is the smallest lexical unit in a program. The Types of tokens are listed below.

1. Identifiers

2. Keywords

3. Literals

4. Punctuations

Identifiers

- Identifiers- In Python, all variables, objects, and functions are given a name. These names are called identifiers.
- Certain rules govern how a name can be given. The guidelines are given below.
 - 1) Every identifier must start with a letter or underscore ("_")
 - 2) The letter or underscore can be followed by any sequence of letters, digits, or underscores.
 - 3) Identifier cannot have a space
 - 4) Identifiers are space sensitive. So the names celsius, Celsius, and CELSIUS are different

Keywords

- Python has many inbuilt functions like input(). So Python does not allow a program to use these function names liberally.

Keywords in Python:

and	as	assert	del	for	False	break	class	finally	is
return	None	continue	def	del	else	if	in	global	not
local	except	in	raise	with	or	else	import	True	Float

Literals

- Literal is one way of specifying data values. These will be the values that never modify while the program is executing.
- Literals in Python are numbers, text, or other fixed data, unlike variables whose values change during the action.
- Some of the literals are listed below:
 1. String literals
 2. Numerical literals
 3. Boolean literals

Literals

- String Literals

```
>>> # string literals
>>> # in single quote
>>> single_quote = 'stringliterals'
>>> # in double quotes
>>> double_quotes = "stringliterals"
>>> # multi-line string
>>> multiline = ''' string
... literals'''
>>> print(single_quote)
stringliterals
>>> print(double_quotes)
stringliterals
>>> print(multiline)
string
literals
>>>
```

- Numerical Literals

Example of handling constant as a variable

```
>>>
>>> PI = 3.14
>>> TEMPERATURE = 9.0
```

Python Block Structure

Left Margin

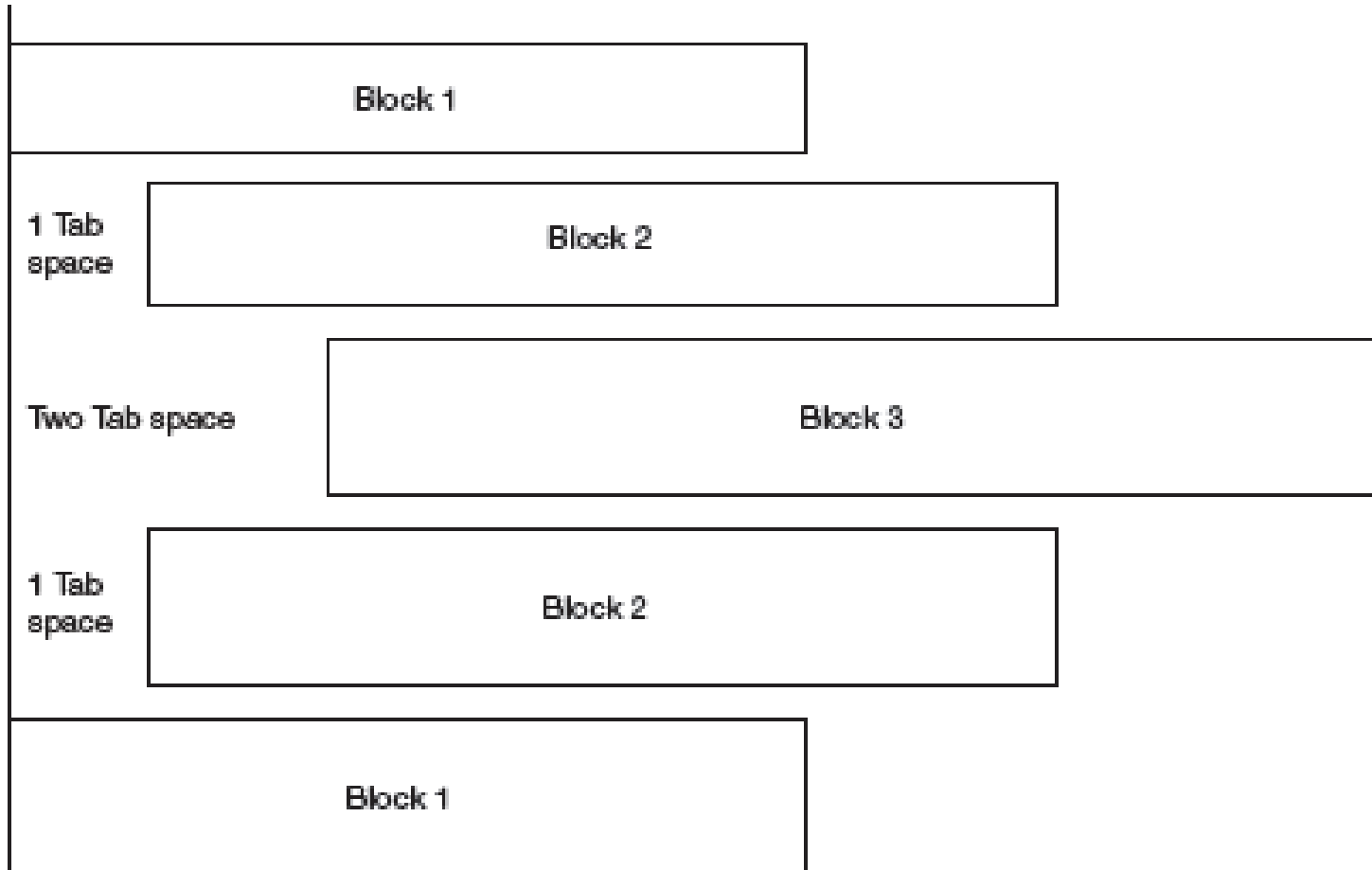


Illustration of Blocks in Python

```
x = 10 # Block 1 starts
if x% 2 == 0: # Block 1 continues
    print("The number is even" ) # Block 2
else: # Block 1
    print("The number is odd") # Block 2
print("Program Ends' ) # Block 1 continues
```

Comments in Python

Block Comments illustration	Inline comments illustration	Multiple quotes illustration
<pre>>>> # This is a single-line comment >>> # This is a block comment >>> # This is a block comment</pre>	<pre>>>> a = 10 # a is initialized to 10 >>> a = 10 # a is initialized to 10 >>> b = 10 # b is initialized to 10 >>> c = a + b # c = a + b >>></pre>	<pre>>>> # Multiline Quote >>> """ ... This is a multiline quote ... using Triple quotes ... """ >>></pre>

Variables and Assignment Statement

- In Python everything is an object, therefore a variable also is an object. Thus, the rules for forming a variable are the same as the identifiers.

No.	Variable names	Valid/Invalid	Remarks
1	y	Valid	All rules are followed
2	x1	valid	All rules are followed
3	Id_number	Valid	Underscore sign is permitted
4	While	Valid	While is different from a while, which is a keyword
5	"g"	Invalid	The quote is not permitted
6	123y	Invalid	A variable cannot start with a number
7	works	Invalid	A variable cannot have a space
8	while	Invalid	Keyword
9	Id!	Invalid	The special symbol! is not allowed

Assignment Statement

- One of the most critical statements in Python is the assignment statement. The syntax of the introductory assignment statement is given as follows:



- An alternative form of an assignment statement is to assign/compute several values simultaneously. The syntax of simultaneous is given as follows:



Name Spaces

- Python interpreter uses a unique structure called namespace using which it maintains a list of names and their associated values.
- Python updates the list when a new variable is created and it is deleted in the namespace when the variable is deleted.

Python Objects

- In Python all are objects, variables, functions or even values.

An object is linked with three concepts-Identify, Zero or more names, and a set of attributes.

1. Identify the Object

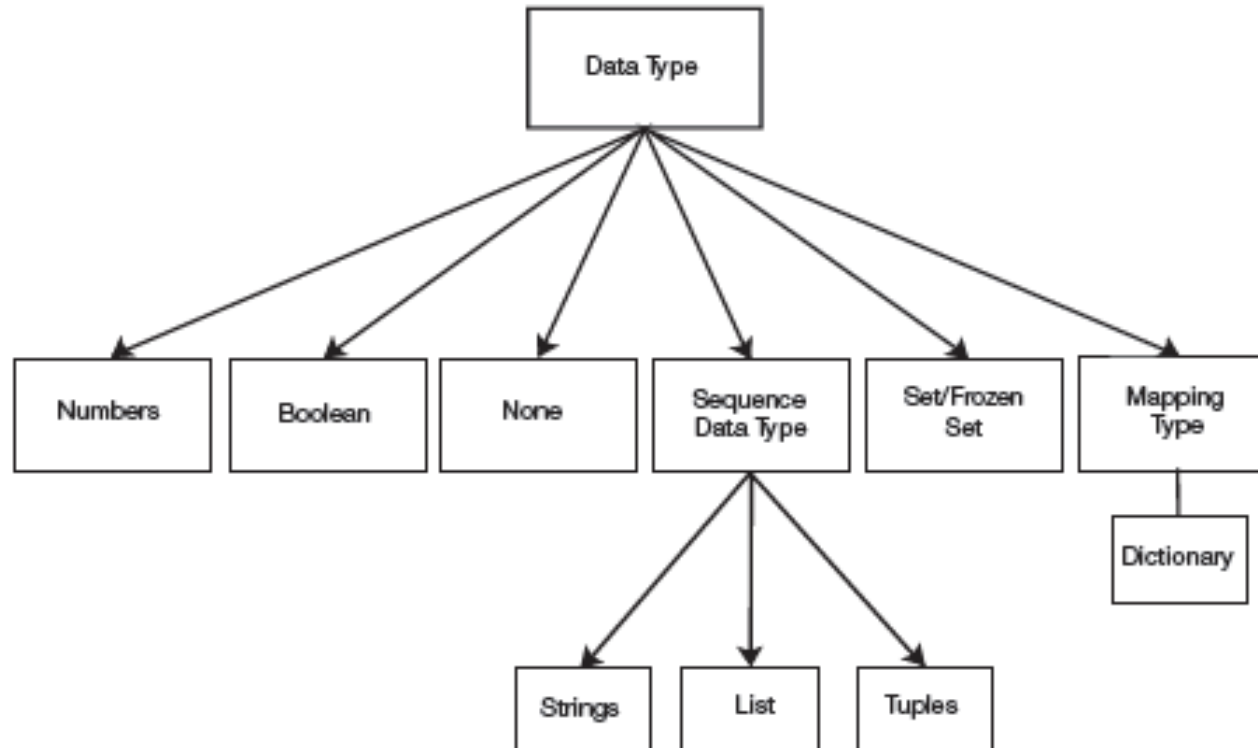
When the object is assigned to a variable, some memory address is allocated to the object.

The Python function to get the memory address is called `id()` function.

```
>>> j = 10
>>> i = i + 7
>>> id(i)
1952232768464
>>> id(j)
1952232768016
```


Data Types in Python

- A data type indicates what values a variable can hold, and an operator indicates the kind of operations performed on the values.
- Python supports numbers and floating numbers. Classification of data types as follows,



Collection data Types

- Two main kinds of collection types are sequence and another type called mapping.
- A sequence is an ordered collection of values. These sequence types are string, list and tuples.
 - 1) A string is a sequence of characters
 - 2) A list is a sequence of values
 - 3) A tuple is a sequence of values like lists but a tuple is immutable

The string is an example of a sequence collection data type. It is a collection of characters delimited by single, double, and triple quotes. It can include letters, numbers, punctuations, and many special/unprintable characters.

String Operations	Operator	Description	Example
Concatenation	+	The process of combining two strings is called concatenation.	<pre>>>> "Hello" + "Hello" 'helloWorld' >>> "hello" + ""+"World" 'helloWorld' >>> "Hello" + " "+"world" 'hello world'</pre>
Repetition	*	If a string is multiplied by 'n' times, the string is repeated 'n' times.	<pre>>>> "Hello" * 5 'Hello Hello Hello Hello Hello'</pre>
Slicing	string[begin:end]	Operation of extracting a subset of characters [begin:end] from a string.	<pre>>>> s = 'hello' >>> s = 'Hello' >>> s[1:4] 'ell' >>></pre>
Length	len(string)	Finds the length of the string	<pre>>>> s = "America" >>> len(s) 7 >>></pre>
Deletion	del string	Delete the string	<pre>>>> s = "America" >>> len(s) 7 >>> del s</pre>

Lists

A list is a set of items or elements. All these items are separated by commas and enclosed within square brackets. In all aspects, lists are similar to arrays but can have non-homogenous content. To create a list, one can just enclose the sequence of objects in square brackets.

```
>>> lst1 = ['hello', "hello", 13, 13.4]
>>> lst1[0]
'hello'
>>> lst1[1]
'hello'
>>> lst1[2]
13
>>> lst1[3]
13.4
>>> lst1[4]
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Illustration of Slicing in Lists

```
>>> lst1[3:]
[13.4]
>>> lst1[1:3]
['hello', 13]
>>>
```

Tuples

- A tuple is a collection of items or elements. A tuple is also a sort of list but the main difference is that tuples are immutable compared to mutable lists.
- The tuple items are enclosed in round brackets and separated by a comma. The empty tuple is (). The tuple with a single element is called a singleton.
- Some of the examples are,

```
>>> tupl = ('a', 'b', 'c', 'd', 'k', 'm')
>>> print(tupl)
('a', 'b', 'c', 'd', 'k', 'm')
```

Dictionary

- A dictionary is an unordered collection of items in the form of keys and values. A dictionary is called a mapping type.
- A map type consists of a set of element pairs. The first element of the pair is called a key, and the second element is called a value.
- To find the corresponding equivalent values, one can search for a key or value.

```
>>> pwd = {'abc':'123', 'def':'456'}
>>> pwd
{'abc': '123', 'def': '456'}
>>>
```

Sets/Frozen Sets

- A set is an unordered collection of zero or more elements with no duplicates. As in mathematics, a set is an unordered collection of items with no duplicate items that use curly braces for sets- {}.
- The set is immutable and unique.

```
>>> lst = {"red", "red", "green"}
>>> lst1 = set(lst)
>>> print(lst1)
{'red', 'green'}
>>>
```

Operations

- One of the other attributes of the objects is operations. Operations are performed on operands. For example, in the expression, 'a+b', 'a' and 'b' are called operands, and '+' is the operator. Operations depend on the data type.

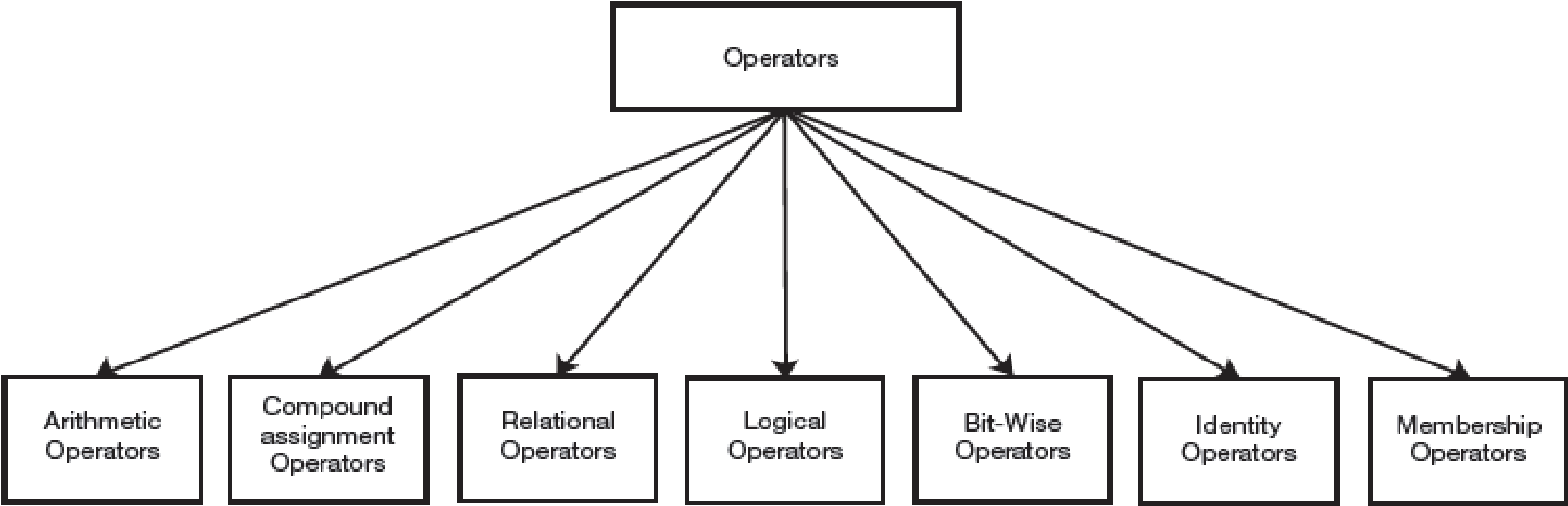


Operations

An operator applies an operation over operands to give a result as output. The operator works over operands. For example, in expression 'x * y', where, 'x' and 'y' are operands and '*' is an operator. There are two types of operators. They are

- 1) Unary operator
- 2) Binary operator
- 3) Ternary operator that works with three operands

The classification of operators



Type Conversions

- There are certain limitations to the data types. For example, one cannot do 'Hello'+4, because one is a string type and another is an integer. For example, input() returns a string, so one needs to convert that to an integer to perform operators. This concept is called type conversion.
- There are two types of data conversion. They are
 - 1) Implicit conversion (or Type Coercion)
 - 2) Explicit conversion (or Type Casting)

Implicit conversion

Python implicitly converts a data type during compilation or runtime. Python automatically converts an integer to a floating point to preserve the fractional part.

```
>>> 20 + 3.33
23.33
```

Explicit conversion

The explicit conversion of a value from one data type into another data type is called typecasting. There are many built-in functions available for type conversion

Functions	Description	Remarks
int(x)	Convert x to integers	>>> int(3.4) 3
round(x)	Convert x into whole integers	>>> round(7.33) 7 >>> round(13.3333, 2) 13.33
oct(x)	Convert x to octal numbers	>>> oct(8) '0o10' >>> oct(14) '0o16'
hex(x)	Convert x to hexadecimal numbers	>>> hex(10) '0xa' >>> hex(49) '0x31'

Function	Description	Remarks
float(x)	Convert x to float numbers	>>> x = 8 >>> float(x) 8.0 >>> float('8') 8.0

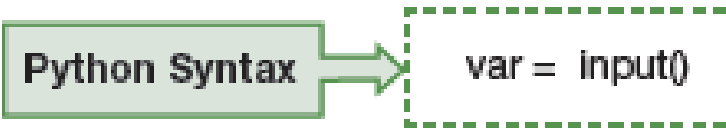
The type conversions for integers and floats to strings are given below in Table 3.25.

Table 3.25: Functions for conversion to string type

Functions	Description	Remarks
str(x)	Converts x into a string	>>> chr(97) 'a'
ord(x)	returns the ASCII code of the character. It is called ordinal.	>>> ord('d') 100

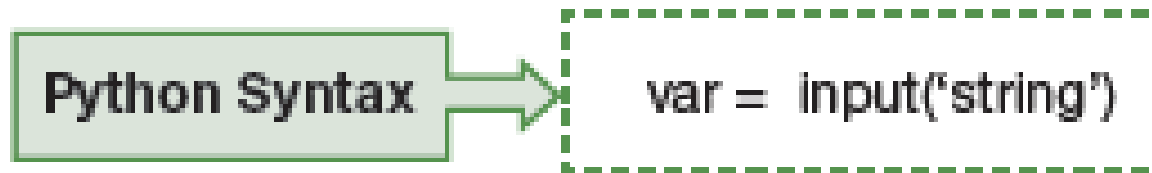
Simple input/output Statements

- Reading Numbers from the Keyboard. – one can read the input from the console using the input() function.
- The syntax of the statement is given as



A diagram consisting of a solid green rectangular box on the left containing the text "Python Syntax". A green arrow points from the right side of this box to a dashed green rectangular box on the right containing the code `var = input()`.

- One can also accept a string using the input() function. The syntax is given as

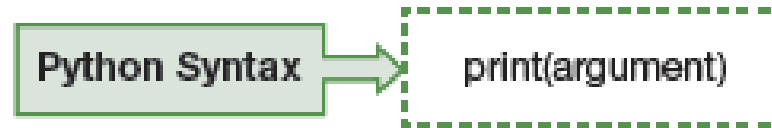


A diagram consisting of a solid green rectangular box on the left containing the text "Python Syntax". A green arrow points from the right side of this box to a dashed green rectangular box on the right containing the code `var = input("string")`.

Illustration of Input statement	Illustration of Type Error	Illustration of input statement	Illustration of eval
<pre data-bbox="191 606 624 871">>>> str = input("Enter String") Enter String Hello >>> str 'Hello'</pre>	<pre data-bbox="736 606 1248 1270">>>> a = input("Enter Number ") Enter Number 10 >>> a = a + 10 Traceback (most recent call last): File "<stdin>", line 1, in <module> TypeError: can only concatenate str (not "int") to str >>></pre>	<pre data-bbox="1286 606 1758 928">>>> a = int(input("Enter Number ")) Enter Number 10 >>> a = a + 10 >>> print(a) 20</pre>	<pre data-bbox="1837 606 2331 928">>>> x = eval(input("Enter the number")) Enter the number10 >>> x = x + 10 >>> print(x) 20</pre>

Print() statement

The contents can be displayed on the screen using a print statement. A python print statement converts all python objects into a string and writes it on in general i/o stream.



- Illustration of Print

```
>>> print(x)
20
>>> print('Hello')
Hello
>>> print("This is correct")
This is correct
>>> print(100)
100
>>> print(1000)
1000
>>> x = 100
>>> print(x)
100
```

Formatting using Print Statement

Many times a simple printing of values is not sufficient. Many projects may require fancy printing as per the requirements. It is called formatting. All variables and constants need to be converted to string type for printing purposes.

- Built-in Methods

Escape Character	Used To Print
<code>\\</code>	A backslash (\)
<code>\"</code>	Double-quote (")
<code>\a</code>	ASCII bell (BEL)
<code>\b</code>	ASCII backspace (BS)
<code>\f</code>	ASCII Form feed (FF)
<code>\n</code>	ASCII linefeed (LF)
<code>\N{name}</code>	Character named name in the Unicode database (Unicode only)

String Formatting

- String formatting is the second way of formatting; here, there will be a format string and arguments.
- The format of this string formatting is given as



- Formatting Illustration

```
>>> "The name".format()  
'The name'
```

Built-in functions

Function name	Purpose	Example
abs()	Returns the absolute value of a number	>>> abs(-50) 50
round()	returns the rounded value to the nearest integer of a given number	>>> round(12.3) 12
sqrt()	returns the square root of the given number	>>> sqrt(256) 16.0
exp()	Returns the exponentiation of a number	>>> exp(10) 22026.465794806718
ceil()	Returns the highest integer by rounding off the given number	>>> ceil(4.2) 5
floor()	Returns the round off the given number to the lowest integer	>>> floor(3.2) 3
log()	Returns the logarithm of a given number	>>> log(100) 4.605170185988092
log10()	Returns the logarithm to base 10 of a given number	>>> log10(100) 2.0
pow()	Returns the result of the power of x to y.	>>> pow(3,2) 9.0 >>>