L4 / Expressions and Statements / Understanding Expressions

An `expression` is a combination of **values(Constants)**, **variables** and **operators**.

An **expression** may also include **call to functions** and **objects**. We will learn about `functions` and `objects` in the later sections.

Operators are **symbols** that represent particular actions. **Operands** participate in the action performed by the operator.

```
Some of the arithmetic operators present in Python are :
Operator    Meaning                     Expression      Value
+           Arithmetic addition         10+20           30  (addition of 10 and 20)
-           Arithmetic subtraction      40-20           20  (subtraction of 20 from
40)
*           Arithmetic Multiplication   5*4             20  (multiplication of 5 and
4)
/           Arithmetic Division         8/4             2   (quotient when 8 is
divided by 4)
%           Modulus (Remainder)         5%3             2   (remainder when 5 is
divided by 3)
More operators will be discussed in the later sections.
```

Any expression evaluates to a single value, which becomes the value of the expression.

Instructions that a Python interpreter can execute are called **statements**. For example, `a = 1` is an assignment statement.

ExpressionExample1.py

Submit

```python
number1 = 20.50
number2 = 38.25
res=number1*number2
print(res)
# Print the multiplication of number1 and number2 using * operator.

string1 = "Amazon"
string2 = "River"
con=string1+string2
print(con)
# Print the concatenated value of string1 and string2 using + operator.

```

Close Reset Submit

L4/Expressions and Statements/Understanding Statements - Print statements

A **statement** in **Python** is a logical instruction which can be read and executed by **Python interpreter**.

In **Python**, we have different kinds of statements :

1. Print statements
2. Assignment statements
3. Selective statements
4. Iterative statements
5. Control Flow statements
6. Function declaration statements

The purpose of Python's **assignment statement** is to associate **variables** with **values** in your program.

It is the only statement that does not start with a keyword.

An assignment statement contains atleast one equal sign (=), to the left hand side of = we have a `variable` and to the right hand side we have an `expression`.

The general form of an **assignment statement** is as follows :

```
target0 = target1 = ... = expression

Here target0, target1, target2 ... are variables and Python will evaluate the
expression to a single value and that single value is assigned to the variables.
This association of a value to a variable is called as binding.
```

For example, Let us consider the following code:

- [x] In **Python**, Statements are executed by Interpreter.
- [ ] 2 = a, is a valid assignment statement in **Python**.
- [x] The association of a value to a variable is called as Binding.
- [ ] In **Python**, assignment statement starts with the keyword assign.
- [x] a = b + a * b is a valid statement in **Python**.

Close Reset Submit

Validation OTP for CodeTantra | CodeTantra Edu | Download file | iLovePDF

csmssengg.codetantra.com/secure/topic-details1.jsp?cid=65969c70422c5e79de0c160e&tid=65969c71422c5e79de0c1655&bd=AMTc3M19jdF9jaA==

Q3

100%

L4 / Expressi... / Types of A...

There are two types of **assignment statements** in **Python**. They are:

1. **Basic assignment statements**
2. **Augmented assignment statements**

The simple syntax for a basic assignment statement is:

```
variable_name = expression
```

In the example given below we are assigning a string value "CodeTantra" to a variable called `name`.

```
name  =  "CodeTantra"
```

Statement1.py

Submit

```
1  message = "Welcome to Python"
2  print(message)
3  # print message
4  number = int(input())
5  # add 5 to the number
6  # print number
7
8  res=number+5
9  print(res)
```

Close | Reset | Submit

Type here to search

39°C Sunny

ENG US 4:46 PM 07/05/2024

L4 / Expressi... / Augmente...

Augmented assignment is a combination of an arithmetic or a binary operation and an assignment operation in a single statement.

We can combine arithmetic operators in assignments to form an **augmented assignment** statement.

The combined operations of the augmented assignments are represented using the following operators : **+=**, **-=**, ***=**, **/=**, **%=**

Let us consider a simple example:

```
a += b
```

The above statement is a shorthand for the below simple

- [ ] **x *= y** means **x * x = y**
- [x] We can perform assignment with '=' operator.
- [x] **a -= b** and **a = a-b** are gives same result.

Close Reset Submit

Validation OTP for CodeTantra | CodeTantra Edu | Download file | iLovePDF

csmssengg.codetantra.com/secure/topic-details1.jsp?cid=65969c70422c5e79de0c160e&tid=65969c71422c5e79de0c1655&bd=AMTc3M19jdF9jaA==

Q1

100%

L4/Indenta... /

Magnifier 100%

Python utilizes **white spaces** to define control flow blocks. This concept is inherited from its predecessor, ABC. Instead of relying on punctuation or keywords, Python uses indentation to signify the scope of a block of code.

The first line of the function definition is called the header, the rest is called the body.

The header has to end with a colon `(:)` and the body has to be indented.

By convention, the indentation is always **four spaces** or **one Tab space**.

The body can contain any number of statements

Indentation1.py

Submit

```
1 if 100.0 > 10.0:
2     print("Small Value:", 10.0)
3
4 else:
5     print("Large Value:", 100.0)
```

Close Reset Submit

Type here to search

39°C Sunny ENG US 4:47 PM 07/05/2024

Validation OTP for CodeTantra | CodeTantra Edu | Download file | iLovePDF

csmssengg.codetantra.com/secure/topic-details1.jsp?cid=65969c70422c5e79de0c160e&tid=65969c71422c5e79de0c1655&bd=AMTc3M19jdF9jaA==

Q2

100%

L4/Indenta... /A simple ...

We can run a **Python** program in two ways. They are:

1. Interactive mode
2. Scripting mode

1. Interactive mode

- Interactive mode in Python offers a command-line environment that provides instant feedback for every statement. It also retains the memory of previously entered statements, allowing them to remain active.
- With each new line of code entered, the interpreter evaluates both the partial code and the entire fed program.
- Interactive mode is particularly useful for swiftly and conveniently executing individual lines or chunks of code. It's a handy platform for experimentation, trying out

Sample.py

Submit

```
1  print("Hello Python")
```

Close Reset Submit

Type here to search

39°C Sunny

ENG US 4:47 PM 07/05/2024