**Variables and Data Types**

**Variables**

Variables are like containers for storing values. Values in the variables can be changed.

**Values**

Consider that variables are like containers for storing information. In context of programming, this information is often referred to as value.

Data Type

In programming languages, every value or data has an associated type to it known as data type. Some commonly used data types

* String
* Integer
* Float
* Boolean

This data type determines how the value or data can be used in the program. For example, mathematical operations can be done on Integer and Float types of data.

**String**

A String is a stream of characters enclosed within quotes. Stream of Characters

* Capital Letters ( A – Z )
* Small Letters ( a – z )
* Digits ( 0 – 9 )
* Special Characters (~ ! @ # $ % ^ . ?,)
* Space

Some Examples

* "Hello World!"
* "some@example.com"
* "1234"

**Integer**

All whole numbers (positive, negative and zero) without any fractional part come under Integers. Examples

...-3, -2, -1, 0, 1, 2, 3,...

**Float**

Any number with a decimal point

24.3, 345.210, -321.86

**Boolean**

In a general sense, anything that can take one of two possible values is considered a Boolean. Examples include the data that can take values like -

True or False

-

Yes or No

-

0 or 1

-

On or Off

, etc.

As per the Python Syntax,

True

and

False

are considered as Boolean values. Notice that both start with a capital letter.

Assigning Value to Variable

The following is the syntax for assigning an integer value

10

to a variable

age



1

age = 10

PYTHON

Here the equals to

=

sign is called as **Assignment Operator** as it is used to assign values to variables.

# Sequence of Instructions

**Program** A program is a sequence of instructions given to a computer.

## Defining a Variable

A variable gets created when you assign a value to it for the first time.

#### Code



1

age = 10

PYTHON

### Printing Value in a Variable

#### Code



1

2

age = 10

print(age)

PYTHON

#### Output



10

#### Code



1

2

age = 10

print("age")

PYTHON

#### Output



age

Variable name enclosed in quotes will print variable rather than the value in it. If you intend to print value, do not enclose the variable in quotes.

## Order of Instructions

Python executes the code line-by-line.

#### Code



1

2

print(age)

age = 10

PYTHON

#### Output



NameError: name 'age' is not defined

Variable

age

is not created by the time we tried to print.

### Spacing in Python

Having spaces at the beginning of line causes errors.

#### Code



1

2

3

a = 10 \* 5

b = 5 \* 0.5

b = a + b

PYTHON

#### Output



File "main.py", line 3

b = a + b

^

IndentationError: unexpected indent

### Variable Assignment

Values in the variables can be changed.

#### Code



1

2

3

4

a = 1

print(a)

a = 2

print(a)

PYTHON

#### Output



1

2

**Examples of Variable Assignment**

#### Code



1

2

3

4

a = 2

print(a)

a = a + 1

print(a)

PYTHON

#### Output



2

3

#### Code



1

2

3

4

5

a = 1

b = 2

a = b + 1

print(a)

print(b)

PYTHON

#### Output



3

2

## Expression

An expression is a valid combination of values, variables and operators.

*Examples* a \* b a + 2 5 *2 + 3* 4

**BODMAS** The standard order of evaluating an expression - *Brackets* (B) - *Orders* (O) - *Division* (D) - *Multiplication* (M) - *Addition* (A) - *Subtraction* (S)

*Step by Step Explanation*



(5 \* 2) + (3 \* 4)

(10) + (12)

22

#### Code



1

2

print(10 / 2 + 3)

print(10 / (2 + 3))

PYTHON

#### Output

# Inputs and Outputs Basics

## Take Input From User

input()

allows flexibility to take the input from the user. Reads a line of input as a string.

#### Code



1

2

username = input()

print(username)

PYTHON

#### Input



Ajay

#### Output



Ajay

# Working with Strings

## String Concatenation

Joining strings together is called string concatenation.

#### Code



1

2

a = "Hello" + " " + "World"

print(a)

PYTHON

#### Output



Hello World

## Concatenation Errors

String Concatenation is possible only with strings.

#### Code



1

2

a = "\*" + 10

print(a)

PYTHON

#### Output



File "main.py", line 1

a = "\*" + 10

^

TypeError:

can only concatenate str (not "int") to str

## String Repetition

\*

operator is used for repeating strings any number of times as required.

#### Code



1

2

a = "\*" \* 10

print(a)

PYTHON

#### Output



\*\*\*\*\*\*\*\*\*\*

#### Code



1

2

3

s = "Python"

s = ("\* " \* 3) + s + (" \*" \* 3)

print(s)

PYTHON

#### Output



\* \* \* Python \* \* \*

## Length of String

len()

returns the number of characters in a given string.

#### Code



1

2

3

username = input()

length = len(username)

print(length)

PYTHON

#### Input



Ravi

#### Output



4

## String Indexing

We can access an individual character in a string using their positions (which start from 0) . These positions are also called as index.

#### Code



1

2

3

username = "Ravi"

first\_letter = username[0]

print(first\_letter)

PYTHON

#### Output



R

### IndexError

Attempting to use an index that is too large will result in an error:

#### Code



1

2

username = "Ravi"

print(username[4])

PYTHON

#### Output

# Type Conversion

## String Slicing

Obtaining a part of a string is called string slicing.

#### Code



1

variable\_name[start\_index:end\_index]

PYTHON

* Start from the

start\_index

 and stops at

end\_index

* end\_index

 is not included in the slice.

#### Code



1

2

3

message = "Hi Ravi"

part = message[3:7]

print(part)

PYTHON

#### Output



Ravi

### Slicing to End

If end index is not specified, slicing stops at the end of the string.

#### Code



1

2

3

message = "Hi Ravi"

part = message[3:]

print(part)

PYTHON

#### Output



Ravi

### Slicing from Start

If start index is not specified, slicing starts from the index 0.

#### Code



1

2

3

message = "Hi Ravi"

part = message[:2]

print(part)

PYTHON

#### Output



Hi

## Checking Data Type

Check the datatype of the variable or value using

type()

**Printing Data Type**

#### Code



1

2

3

print(type(10))

print(type(4.2))

print(type("Hi"))

PYTHON

#### Output



<class 'int'>

<class 'float'>

<class 'str'>

## Type Conversion

Converting the value of one data type to another data type is called *Type Conversion* or *Type Casting*. We can convert

* String to Integer
* Integer to Float
* Float to String and so on.

### String to Integer

int()

converts valid data of any type to integer

#### Code



1

2

3

4

a = "5"

a = int(a)

print(type(a))

print(a)

PYTHON

#### Output



<class 'int'>

5

### Invalid Integer Conversion

#### Code



1

2

3

a = "Five"

a = int(a)

print(type(a))

PYTHON

#### Output



ValueError:

invalid literal for int() with base 10: 'Five'

#### Code



1

2

3

a = "5.0"

a = int(a)

print(type(a))

PYTHON

#### Output



invalid literal for int() with base 10: '5.0'

### Adding Two Numbers

#### Code



1

2

3

4

5

6

a = input()

a = int(a)

b = input()

b = int(b)

result = a + b

print(result)

PYTHON

#### Input



2

3

#### Output



5

### Integer to String

str()

converts data of any type to a string.

#### Code

1

2

3

4

5

6

a = input()

a = int(a)

b = input()

b = int(b)

result = a + b

print("Sum: " + str(result))

PYTHON

#### Input

2

3

#### Output

Sum: 5

## Summary

1. int()

 -> Converts to integer data type

1. float()

 -> Converts to float data type

1. str()

 -> Converts to string data type

1. bool()

 -> Converts to boolean data type