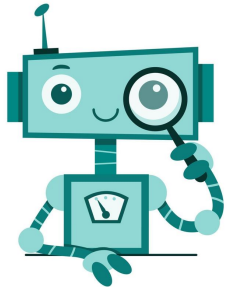
Artificial Intelligence (AI)



Week-2

AGENDA

- Python syntax
- Print statement
- Variables & Data types
- Python operators

Python syntax

- Set of rules that describe how a Python programme is written and understood **(by both the runtime system and by human readers)**.
- Shares a lot of similarities with Perl, C++, and Java.

Print Statement

- Print statement actually a **Call** to print something.
- Takes one argument (single parameter).
- **Print()** function sends a message to the screen or another standard output device.
- The message can be a string or another object, which will be converted to a string before being displayed on the screen.

Syntax:

print(*object(s)*, sep=*separator*, end=*end*, file=*file*, flush=*flush*)

Any Object

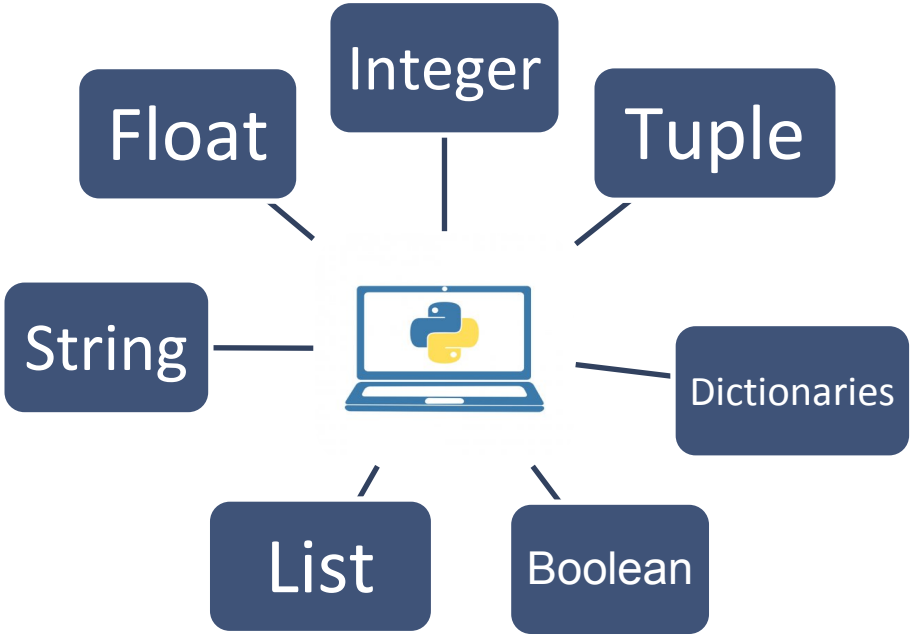
Separate the objects

Print at the end

object with a write method

if the output is flushed (True) or buffered (False). Default is False

Variables & Data type In Python



Variables:

- A variable is a value's container.
- When you first assign a value to a variable, it is created.
- Can store a variety of data, & different types can perform different tasks.

Data Types

A particular **kind** of data item, as defined by the **values** it can take. OR

An **attribute** of data which tells the compiler or interpreter how the programmer intends to use the data. OR

The **classification** or **categorization** of data items is known as Data types.

Data Types:

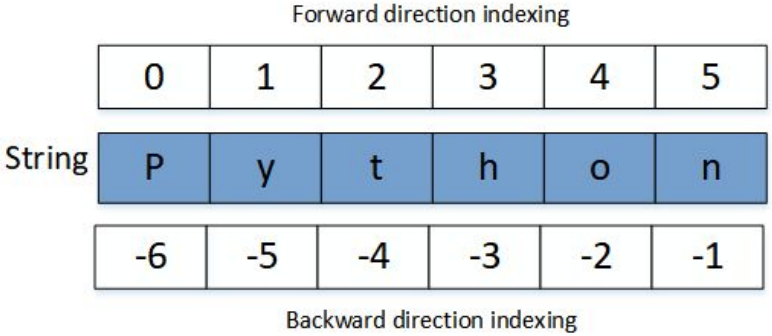
Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray

String: (str or text)

- Sequence of **characters** used to store text.
- It is a **collection** of one or more characters put in a single quote, double-quote or triple quote.
- **In Python** a character is a string of length one.
- A string can also include **digits** and **symbols;** however, it is treated as **text**.
- Also, there is **indexing** in python string to **access** the character of the word

Accessing character of a String:

- A character also called element of a string can be accessed with its index number.
- In Python, index number starts with 0 in forward direction and -1 in backward direction.



https://www.alphacodingskills.com/python/img/python-string.png

Example: Strings in Python using single quotes

```
#String with single Quotes
String1 = 'Welcome to the AI Class'
print("String with the use of Single Quotes: ")
print(String1)
#String with double Quotes
String1 = "I'm a Python"
print("\nString with the use of Double Quotes: ")
print(String1)
# String with triple Quotes
String1 = '''I'm a Python and I live in a world of "AI"'''
print("\nString with the use of Triple Quotes: ")
print(String1)
# String with triple Quotes allows multiple lines
String1 = '''AI
           With
          Python'''
print("\nCreating a multiline String: ")
print(String1)
```

Example: Accessing characters of String

```
#Access characters of String
String1 = "ArtificialIntelligence"
print("Initial String: ")
print(String1)

# Printing First character
print("\nFirst character of String is: ")
print(String1[0])

# Printing Last character
print("\nLast character of String is: ")
print(String1[-1])
```

String Slicing:

```
MyString = 'Learn Python'
print(MyString[0:5])
print(MyString[-12:-7],"\n")

print(MyString[6:])
print(MyString[:-7],"\n")

print(MyString[:])
```

Output: Learn
Learn

Python
Learn

Learn Python

String Concatenation:

We can join two strings by using **+** operator.

```
text_1 = 'Learn'
text_2 = 'Python'
MyString = text_1 + text_2
print(MyString)

MyString = text_1 + " " + text_2
print(MyString)
```

Output: LearnPython
Learn Python

Numeric:

- Numeric data type **represent** the data which has **numeric** value.
- Numeric value can be **integer**, **floating** number or even **complex** numbers.
- These values are **defined as** int, float and complex class in Python.

Integers: (int)

- It is the most common **numeric** data type used to store numbers (positive or negative **whole numbers**) without a fractional component.
- In Python there is **no limit** to how long an integer value can be.

Four forms of Integers:

We can represent int in 4 forms:

- **Decimal form**: it is by default and digits are from 0-9.
- **Binary form:** base-2, 0 and 1.
- **Octal form:** 0-7
- **Hexadecimal:** 0-9, A-F

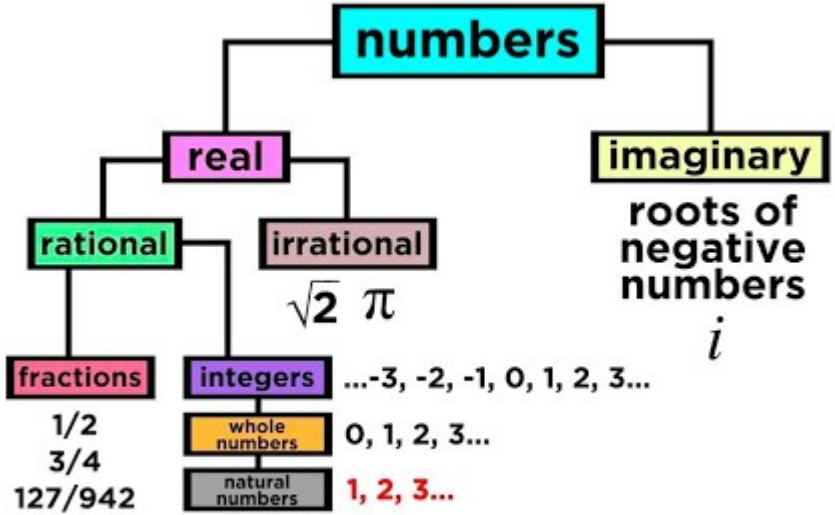
- By default the answer of these four forms are in decimal forms, so to change that we need to convert them. So, Here:
- For converting into **binary** we will do : bin(15) -.> 0b1111
- For converting into **octal** : oct(100) -> 0o144
- For converting into **hexadecimal** : hex (1000) -> 0x3e8

Float:

- Float is also a numeric data type used to store numbers that may have a **fractional component** (real number with floating point representation).
- In float, we can only have **decimal** no binary, oct, or hex. Eg: a=10.5 So when we see the type of “a” It will be float type.

Complex Numbers:

Complex number is represented by complex class. It is specified as (real part) + (imaginary part)j. Like **A+bj**



https://i.ytimg.com/vi/QUGmwPwtbpg/hqdefault.jpg

Complex Numbers:

- The real part can be binary, octal, or hex anything, but the imaginary should be decimal only.
- OB111 + 20j it is correct, But 14+ 0B101 it is incorrect.
- We can also take float values as well x=10.5+20.1j.
- Addition or subtraction of the complex numbers is simple. We simply do addition or subtraction of the real part of both numbers and imaginary parts of both numbers.

Example: Python program to demonstrate numeric value

```
a = 5
print("Type of a: ", type(a))

b = 5.0
print("\nType of b: ", type(b))

c = 2 + 4j
print("\nType of c: ", type(c))
```

Boolean:

Data type with one of the two built-in values, True or False. Sometimes a boolean value is also **represented** as 0 (for false) and 1 (for true).

True and **False** with capital **‘T’** and **‘F’** are valid booleans otherwise python will throw an **error**.

Non-Boolean objects can be evaluated in Boolean context as well and determined to be **true or false.**

Example: Python program to demonstrate boolean type

```
x = bool(5)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

Output:

```
True
<class 'bool'>
```

Examples

Print Integer Datatype :

```
x = 10
print(x)
print(type(x))
```

```
10
<class ‘int'>
```

Print Float datatype :

```
x = 10.5
print(x)
print(type(x))
```

```
10.5
<class 'float'>
```

Print String Datatype :

```
x = "Hello World"
print(x)
print(type(x))
```

```
Hello World
<class 'str'>
```

Print Boolean Datatype :

```
x = False
print(x)
print(type(x))
```

```
True
<class 'bool'>
```

Examples

Print List Datatype :

```
x = [“Lilly", “Rose", “Tulip“]
print(x)
print(type(x))
```

```
[Lilly’, Rose', ‘Tulip’]
<class ‘list'>
```

Print Dictionary datatype :

```
x = {"name" : “ali", "age" :40}
print(x)
print(type(x))
```

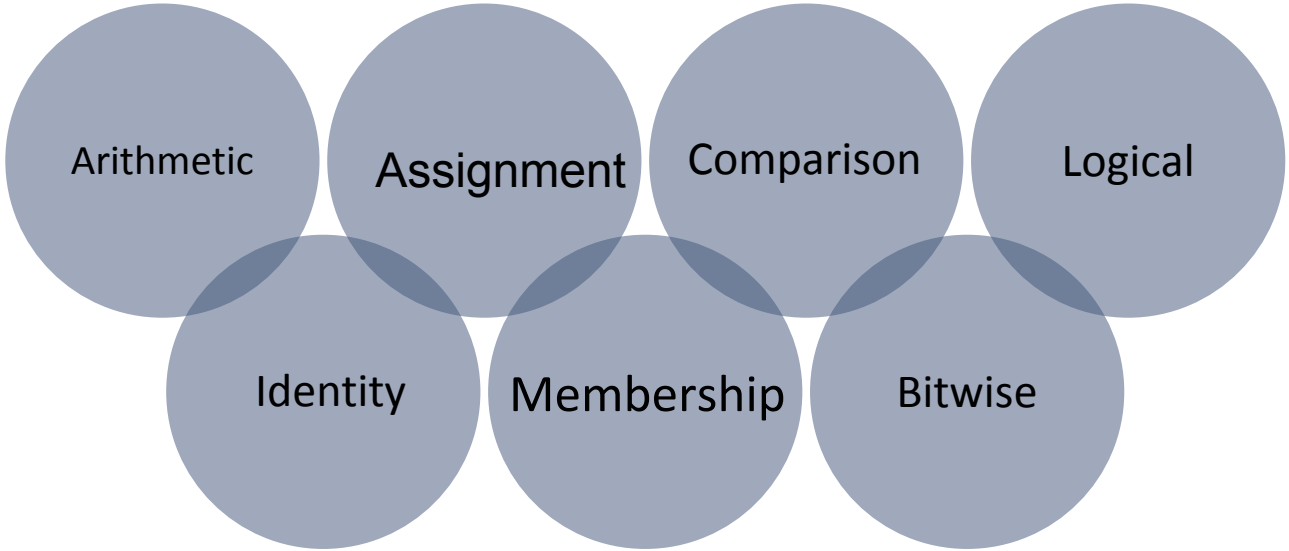
```
{'name': 'John', 'age': 36}
<class 'dict'>
```

Print Tuple Datatype :

```
x = (“Lilly", “Rose", “Tulip")
print(x)
print(type(x))
```

```
(Lilly’, Rose', ‘Tulip')
<class 'tuple'>
```

Python Operators



- Operators are special symbols in Python that perform some specific computation.
- Operators are used to perform operations on variables and values.

Arithmetic Operators

Addition:

```
x = 2
y = 3
print(x + y)
```

```
5
```

Subtraction:

```
x = 3
y = 2
print(x - y)
```

```
1
```

Modulus:

```
x = 5
y = 3
print(x % y)
```

```
2
```

Multiplication:

```
x = 2
y = 3
print(x * y)
```

```
6
```

Division:

```
x = 6
y = 3
print(x / y)
```

```
2
```

Assignment Operators

=:

```
x = 5
print(x)
```

```
5
```

+=:

```
x = 4
x += 3
print(x)
```

```
7
```

/=:

```
x = 25
x/= 5
print(x)
```

```
5
```

-=:

```
x = 7
x - = 3
print(x )
```

```
4
```

***=:**

```
x = 6
x*= 5
print(x )
```

```
30
```

Comparison Operators

Equal:

```
x = 5
y = 3
print(x == y)
```

False

Not Equal:

```
x = 5
y = 3
print (x == y)
```

True

Less than and greater than:

```
x = 25
y= 65
print(x <y)
print(x >y)
```

True
False

Greater than or Equal:

```
x = 7
y = 3
print(x > y)
```

True

Smaller than or Equal :

```
x = 8
y = 3
print(x > y)
```

False

Logical Operators

AND:

```
x = 2
print(x > 3 and x <6)
```

True

NOT:

```
x = 5
print(not(x > 3 and x < 10)
```

False

OR:

```
x = 7
print(x < 9 or x < 4)
```

True

Identity Operators

is:

```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(x is z)
print(x is y)
print(x == y)
```

```
True
False
True
```

Is not:

```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(x is not z)
print(x is not y)
print(x != y)
```

```
False
True
False
```

Membership Operators

in:

```
x = ["apple", "banana"]

print("banana" in x)
```

```
True
```

Not in:

```
x = ["apple", "banana"]

print("pineapple" not in x)
```

```
True
```

Lab Task -2

1. Open Jupyter Notebook.
2. Create a new Python 3 notebook.
3. Write a Python code that does the following:
    - Declare variables for your name, roll number, and CGPA.
    - Assign appropriate values to each variable (name as a string, roll number as an integer, and CGPA as a float).
    - Print your name, roll number, and CGPA.
    - Check and print the data type of each variable.

Expected Output of Task 2

```
Name: Your Name
Roll Number: 21
CGPA: 2.5
Data type of name: <class 'str'>
Data type of roll number: <class 'int'>
Data type of CGPA: <class 'float'>
```

```
[ ]:
```

Lab Task -3

Write a Python code snippet that defines two variables a and b with any numeric values of your choice and perform the following operations:

1. **Arithmetic Operators:**
   Perform basic arithmetic operations like addition, subtraction, multiplication, division, etc., on given numeric values.
2. **Comparison Operators:**
   Compare two numeric values using operators such as equal to, not equal to, greater than, less than, etc.
3. **Logical Operators:**
   Apply logical operators like AND, OR, and NOT on Boolean values.
4. **Membership Operators:**
   Check if a given element belongs to a sequence using membership operators (in and not in).
5. **Identity Operators:**
   Determine whether two variables refer to the same object in memory using identity operators (is and is not).

Back to main