Programming

Machine

Translator

(Compiler / Interpreter)

Code

What is Python?

- Python is simple & easy
- Free & Open Source
- High Level Language
- Developed by Guido van Rossum
- Portable

Our First Program

```
print("Hello World")
```

Python Character Set

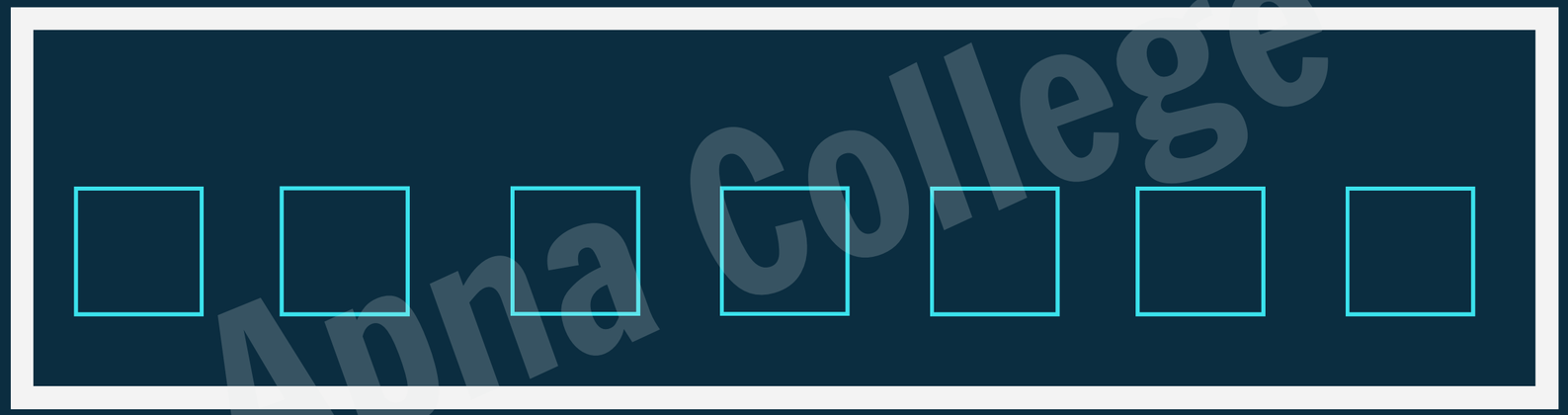
- Letters - A to Z, a to z
- Digits - 0 to 9
- Special Symbols - + - * / etc.
- Whitespaces - Blank Space, tab, carriage return, newline, formfeed
- Other characters - Python can process all ASCII and Unicode characters as part of data or literals

Variables

A variable is a name given to a memory location in a program.

```
name = "Shradha"
age = 23
price = 25.99
```

Memory



name = "Shradha"

age = 23

price = 25.99

Rules for Identifiers

1. Identifiers can be combination of uppercase and lowercase letters, digits or an underscore(_). So **myVariable**, **variable_1**, **variable_for_print** all are valid python identifiers.
2. An Identifier can not start with digit. So while **variable1** is valid, **1variable** is not valid.
3. We can't use special symbols like !,#,@,%,$ etc in our Identifier.
4. Identifier can be of any length.

Data Types

- Integers
- String
- Float
- Boolean
- None

in Python, the primitive (basic) data types include:

- int: Integer values (e.g., 10, -3)
- float: Floating-point numbers (e.g., 3.14, -0.01)
- bool: Boolean values (True or False)
- str: String (sequence of characters, e.g., "hello")
- NoneType: Represents the absence of a value (None)

These are the fundamental types that Python uses to build more complex structures.

The complex data type is a primitive data type in Python.

It represents complex numbers and is used to store numbers in the form a + bj, where:

a is the real part (e.g., 5 in 5 + 3j)
b is the imaginary part (e.g., 3 in 5 + 3j)
j is the imaginary unit
For example:

python
Copy code

```
z = 2 + 3j
print(z.real)  # Output: 2.0 (real part)
print(z.imag)  # Output: 3.0 (imaginary part)
```

Data Types

```
print(type(age))
print(type(pi))
print(type(complex_num))
print(type(A))
print(type(name))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'bool'>
<class 'str'>
```


Non-primitive (or composite) data types in Python are those that can hold multiple values or are more complex structures. These include:

- List: Ordered and mutable collection (e.g., [1, 2, 3])
- Tuple: Ordered and immutable collection (e.g., (1, 2, 3))
- Set: Unordered collection of unique elements (e.g., {1, 2, 3})
- Dictionary (dict): Key-value pairs (e.g., {"name": "Niketan", "age": 25})
- Bytes: Immutable sequence of bytes (e.g., b"hello")
- Bytearray: Mutable sequence of bytes (e.g., bytearray(b"hello"))
- Frozenset: Immutable version of a set (e.g., frozenset({1, 2, 3}))

These data types allow for the storage and manipulation of more complex data structures in Python.

Keywords

Keywords are reserved words in python.

*False should be uppercase

and	else	in	return
as	except	is	True
assert	finally	lambda	try
break	false	nonlocal	with
class	for	None	while
continue	from	not	yield
def	global	or	
del	if	pass	
elif	import	raise	

Print Sum

Comments in Python

```
# Single Line Comment

"""
Multi Line
Comment
"""
```

Types of Operators

An operator is a symbol that performs a certain operation between operands.

- Arithmetic Operators (+ , - , * , / , % , **)
- Relational / Comparison Operators (== , != , > , < , >= , <=)
- Assignment Operators (= , +=, -= , *= , /= , %= , **=)
- Logical Operators (not , and , or)

Type Conversion

```
a, b = 1, 2.0
sum = a + b
```



```
#error
a, b = 1, "2"
sum = a + b
```

Type Casting

```
a, b = 1, "2"
c = int(b)
sum = a + c
```

Type Casting

Function	Description
int(y [base])	It converts *y* to an integer, and Base specifies the number base. For example, if you want to convert the string in decimal numbers then you'll use 10 as base.
float(y)	It converts *y* to a floating-point number.
complex(real [imag])	It creates a complex number.
str(y)	It converts *y* to a string.
tuple(y)	It converts *y* to a tuple.
list(y)	It converts *y* to a list.
set(y)	It converts *y* to a set.
dict(y)	It creates a dictionary and *y* should be a sequence of (key, value) tuples.
ord(y)	It converts a character into an integer.
hex(y)	It converts an integer to a hexadecimal string.
oct(y)	It converts an integer to an octal string

Input in Python

input( ) statement is used to accept values (using keyboard) from user


```
input( )  #result for input( ) is always a str

int ( input( ) )  #int

float ( input( ) )   #float
```

give code eg of all 3

Let‘s Practice

Write a Program to input 2 numbers & print their sum.

Let‘s Practice

WAP to input side of a square & print its area.

Let‘s Practice

WAP to input 2 floating point numbers & print their average.

Let‘s Practice

WAP to input 2 int numbers, a and b.
Print True if a is greater than or equal to b. If not print False.