→ Python Data Types

. Variables can store data of different types, and different types can do different things

. In python, the data type is set when we assign a value to a variable.

. Python has the following data types built-in by default;

Text Type: str x = "Hello World"

Numeric Type: int x = 20
float x = 20.5
complex x = 1j

Sequence Type: list x = ["apple", "banana", "cherry"]
tuple x = ("apple", "banana", "cherry")
range x = range(6)

Mapping Type: dict x = {"name": "Jhon", "age": 36}

Set Type: set x = {"apple", "banana", "cherry"}
frozenset x = frozenset({"apple", "banana"})

Boolean Type: bool x = True / False

Binary Type: bytes x = b"Hello"
bytearray x = bytearray(5)
memoryview x = memoryview(bytes(5))

None Type: NoneType x = None

→ We can get the data type of any object by using the type() function:

```
x = 5
print(type(x)) # output: <class 'int'>
```

→ Python Casting

There may be times when we want to specify a type on to a variable. This can be done with casting:

- int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)

- float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)

- str() - constructs a string from a wide variety of data types, including strings, integers and float.

→ Python Strings

Strings in python are surrounded by either single quotation marks '', or double quotation marks " "

↳ Multiline String

we can assign a multiline string to a variable by using three quotes :

```
a = """This is a multiline
      String. assigned using
      three quotes """
```

↳ Strings are Arrays

Strings in python are arrays of bytes representing unicode characters.

Square brackets can be used to access elements of string

Example :

```
# Get the character at position 1
a = "Hello, World!"
print (a[1]) # output : e
```

↳ Looping Through a String

Since strings are arrays, we can loop through the characters in a string with a 'for' loop.

Example:

```
# loop through the letters in the word "python"
  for x in "python":
    print(x)  # output: p y t h o n
```

↳ String Length

To get the length of a string, use the 'len()' functio

Example:

```
a = "Python"
print(len(a))  # output: 6
```

↳ Check String

To check if a certain phrase or character is present in a string, we can use the keyword 'in'

Example:

```
# Check if "free" is present in the following text:
  txt = "The best things in life are free"
  print("free" in txt)  # output: True.
```

Format Strings

As we said before, we cannot combine strings and numbers with the '+' operator

But we can use the 'format()' method to solve the problem.

The "format()" methode takes the passed arguments, format them, and places them in the string where the placeholders "{}" are:

Example:

```
age = 36
txt = "My name is x, and I am {}"
print(txt.format(age)) # output: My name is x, and
                                 I am 36.
```

The "format()" method takes unlimited number of arguments, and are placed into the respective placeholders

Escape Character

\' single quote	\" double quote
\\ backslash	\n newline
\r Carriage return	\t Tab
\b backspace	

String Methods

Python has a set of built-in methods that we can use on strings

`capitalize()` : Converts the first character to uppercase

`casefold()` : Converts string into lower case

`count()` : Returns the number of time a specified value occurs in a string

`endswith()` : Return true if the string end with the specified value

`find()` : Searches the string for a specified value and
`index()` returns the position of where it was found

`format()` : Formats specified value in a string

`lower()` : Converts a string into lower case

`replace()` : Returns a string where a specified value is replaced with a specified value

`split()` : Split the string at the specified separator and return a list

startwith(): return true if the string start with the specified value

upper(): Converts a String to upper case

Strip(): Returns a trimmed version of the string (without spacing)

rsplit(): Splits the string at the specified separator and return a list

→ Python Booleans

Boolean represent one of two values : True or False

↳ Evaluate Values and Variables

The 'bool()' function allows us to evaluate any value, and give us True or False in return.

. almost any value is evaluated to True if it has some sort of content.

. Any String is True, except emty strings " "

. Any number is True, except 0

. Any list, tuple, set and dictionary are true except empty ones (), [], {}

→ Python Operater

Operators are used to perform operations on variables and values.

1. Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (remainder)
**	Power
//	Floor division (rounded result)

2. Assignment Operators

=	x = 5	
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

3. Comparison Operations

==	Equal
!=	Not Equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

4. Logical Operations

and	returns True if both are True
or	returns True if one of them is True
not	reverse the result

5. Identity Operators

is	returns true if both are the same object
isnot	returns true if both are not the same object

6. Membership Operators

in	returns True if the object is present in a sequence
notin	returns True if the object isn't present in a sequence

7. Bitwise Operators

&	AND
|	OR
^	XOR
~	NOT