

Python Programming



Session Content

Variables

- Definition & Working with it
- Variable naming

Data types

- Numbers
- Arithmetic operations
- Boolean & Comparison
- Logical & Bitwise operators
- Strings



Session Content

Variables

- Definition & Working with it
- Variable naming

Data types

- Numbers
- Arithmetic operations
- Boolean & Comparison
- Logical & Bitwise operators
- Strings



Variables

- Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value.

Identifier

Memory

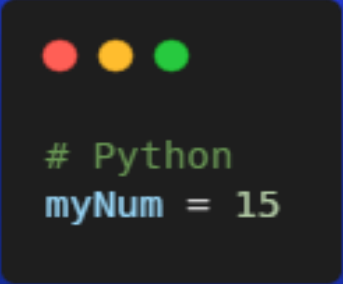
myNumber



Address	Value
0012CCGWH80	23

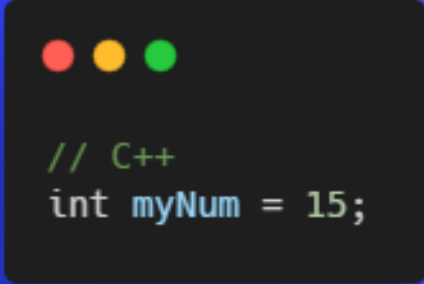
Variables

- In Python, we don't need to specify the type of variable because Python is an inferred language and smart enough to get variable type.

A dark-themed code editor window with three colored window control buttons (red, yellow, green) at the top left. It contains two lines of Python code: a comment line starting with a hash and the word 'Python', followed by an assignment statement.

```
# Python  
myNum = 15
```

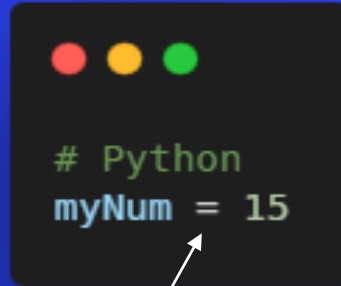
VS

A dark-themed code editor window with three colored window control buttons (red, yellow, green) at the top left. It contains two lines of C++ code: a comment line starting with two slashes and the text 'C++', followed by a C++ declaration and assignment statement.

```
// C++  
int myNum = 15;
```

Variables

- Assignment is done with a single equals sign (=).



```
# Python  
myNum = 15
```


Assignment operator



Variables

- Python allows us to assign a value to multiple variables in a single statement, which is also known as multiple assignments.

1. Assigning single value to multiple variables:




```
x=y=z=50  
print(x) # x=50  
print(y) # y=50  
print(z) # z=50
```

Variables

- Python allows us to assign a value to multiple variables in a single statement, which is also known as multiple assignments.

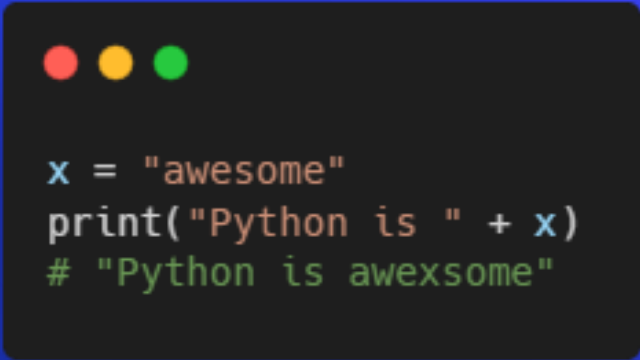
1. Assigning multiple values to multiple variables:



```
a,b,c=5,10,15  
print(a) # a=5  
print(b) # b=10  
print(c) # c=15
```


Variables

- The Python print statement is often used to output variables.



```
x = "awesome"  
print("Python is " + x)  
# "Python is awexsome"
```

Variables

- We can delete the variable using the del keyword. The syntax is given below.

```
● ● ●  
  
# Assigning a value to x  
x = 6  
print(x)  
# deleting a variable.  
del x  
print(x)
```

```
● ● ●  
  
# Output  
  
...  
6  
NameError: name 'x' is not defined  
...
```

Quiz

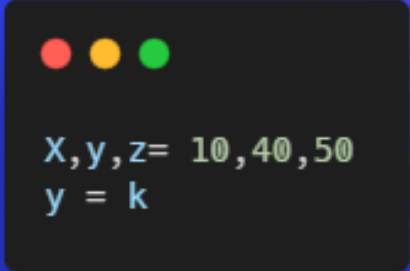
Q. What is the value of the variable **K** ?

A - 10

B - 100

C - 40

D - 50



```
X,y,z= 10,40,50  
y = k
```



Multiple Choice

Session Content

Variables

- Definition & Working with it
- Variable naming

Data types

- Numbers
- Arithmetic operations
- Boolean & Comparison
- Logical & Bitwise operators
- Strings



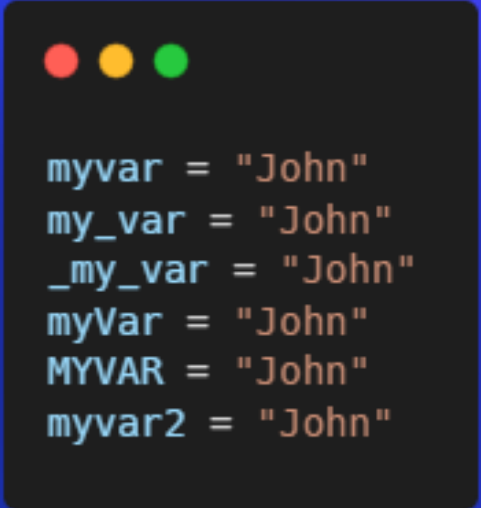
Variable naming

- The rules to name an identifier are given below.

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
- Identifier name mustn't contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language (print , if, for).
- Identifier names are case sensitive; for example, my name, and MyName is not the same.
- Use easy and meaningful name and related to the problem.

Variable naming

- Correct variable names:



```
myvar = "John"  
my_var = "John"  
_my_var = "John"  
myVar = "John"  
MYVAR = "John"  
myvar2 = "John"
```

Variable naming

- Wrong variable names:



```
2myvar = "John"  
my-var = "John"  
my var = "John"
```

Quiz

Q. Stores a piece of data, and gives it a specific name

A - variable

B - whitespace

C - interpreter

D - modulus



Multiple Choice

Quiz

Q. Is the name of this variable, correct? → `V@riable1` = 15

A - True

B - False



Multiple Choice

Session Content

Variables

- Definition & Working with it
- Variable naming

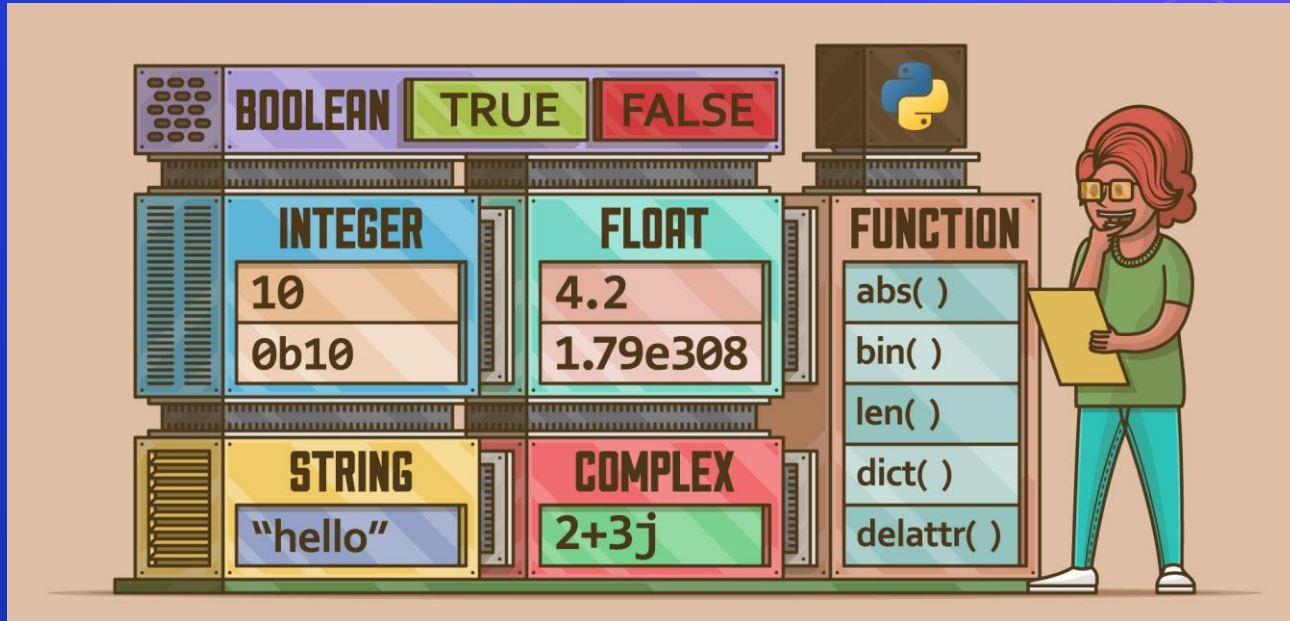
Data types

- Numbers
- Arithmetic operations
- Boolean & Comparison
- Logical & Bitwise operators
- Strings



Data types

- Variables can hold values, and every value has a data-type.
- hence, we do not need to define the type of the variable while declaring it.
- The interpreter implicitly binds the value with its type.



Session Content

Variables

- Definition & Working with it
- Variable naming

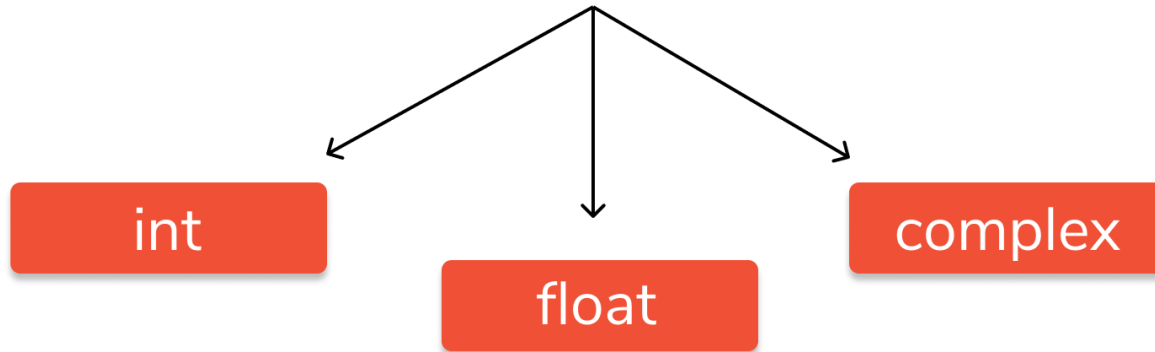
Data types

- Numbers
- Arithmetic operations
- Boolean & Comparison
- Logical & Bitwise operators
- Strings



Numeric Types

Python Numbers



Integer :

An integer (more commonly called an int) is a number without a decimal point.



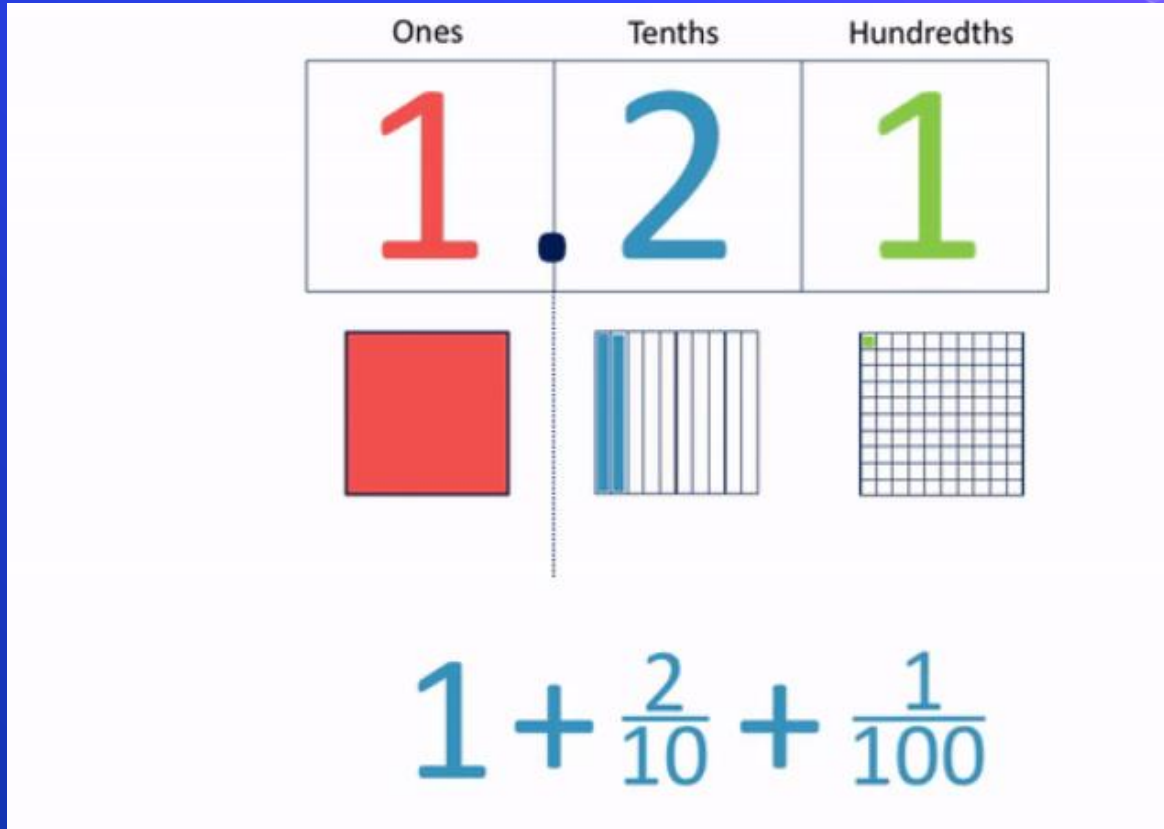
an Integer

is a whole number from the set of negative, non-negative, positive and 0 numbers.

- **Negative** : -1, -2, -3, -4, -5 ...
- **Non-negative** : 0, 6, 7, 8, 9 ...
- **Positive** : 1, 2, 3, 4, 5 ...
- **Zero** : 0 all by itself

Float :

- A float is a floating-point number, which means it is a **number that has decimal place.**



complex number :

-A complex number is the sum of a real number and an imaginary number.

Complex Numbers

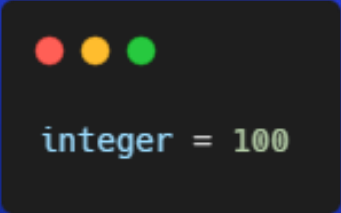
$$Z = a + ib$$

Real part

Imaginary part

Imaginary number

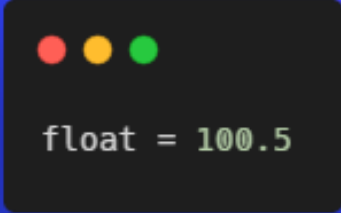
Integer :



```
integer = 100
```

A terminal window with a dark background and three colored dots (red, yellow, green) at the top. It displays the code `integer = 100`.

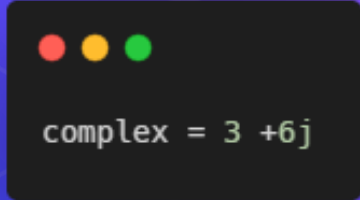
Float :



```
float = 100.5
```

A terminal window with a dark background and three colored dots (red, yellow, green) at the top. It displays the code `float = 100.5`.

Complex :



```
complex = 3 + 6j
```

A terminal window with a dark background and three colored dots (red, yellow, green) at the top. It displays the code `complex = 3 + 6j`.

- We can use integer, float and complex To perform arithmetic operations
But first, We must learn about arithmetic operators.

Quiz

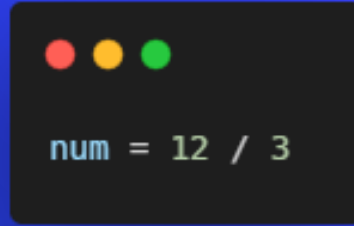
Q. What is the type of Output ?

A – int

B – float

C – complex

D – None of them

A terminal window with a black background and three colored window control buttons (red, yellow, green) at the top left. It displays the code `num = 12 / 3` in a light green monospace font.

```
num = 12 / 3
```



Multiple Choice

Session Content

Variables

- Definition & Working with it
- Variable naming

Data types

- Numbers
- Arithmetic operations
- Boolean & Comparison
- Logical & Bitwise operators
- Strings



arithmetic operators :

- An operator that performs arithmetic operations on groups and numbers

Operators	Meaning	Example	Result
+	Addition	$4 + 2$	6
-	Subtraction	$4 - 2$	2
*	Multiplication	$4 * 2$	8
/	Division	$4 / 2$	2
%	Modulus operator to get remainder in integer division	$5 \% 2$	1
**	Exponent	$5 ** 2 = 5^2$	25
//	Integer Division/ Floor Division	$5 // 2$ $-5 // 2$	2 -3

arithmetic operators :

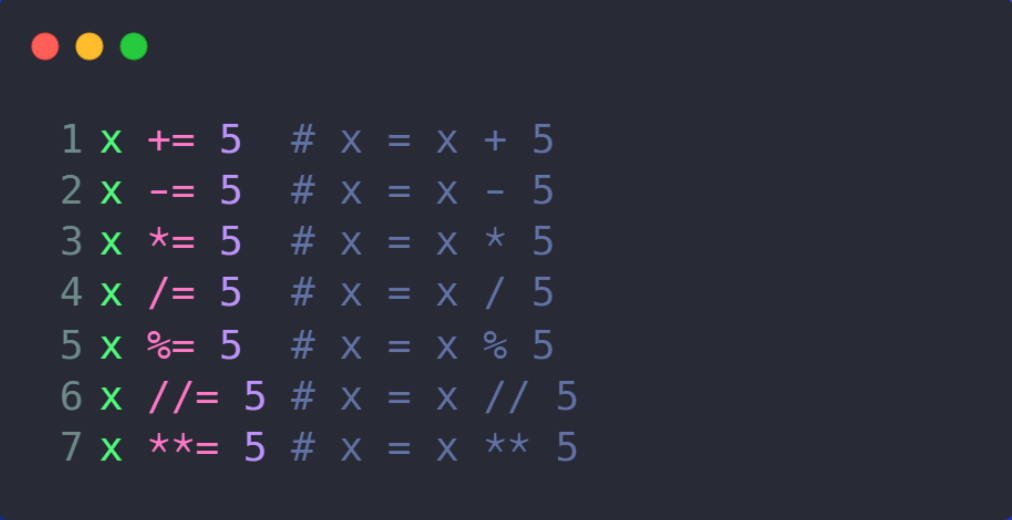
- An operator that performs arithmetic operations on groups and numbers



```
1 3 + 5      # result is 8
2 10 - 7     # result is 3
3 2 * 5      # result is 10
4 15 / 5     # result is 3
5 3 / 2      # result is 1.5
6 3 // 2     # result is 1
7 32 % 3     # result is 2
8 2 ** 3     # result is 8
9 4 ** 0.5   # result is 2
```

arithmetic operators :

- An operator that performs arithmetic operations on groups and numbers



```
1 x += 5 # x = x + 5
2 x -= 5 # x = x - 5
3 x *= 5 # x = x * 5
4 x /= 5 # x = x / 5
5 x %= 5 # x = x % 5
6 x //= 5 # x = x // 5
7 x **= 5 # x = x ** 5
```

A different way to
write the equation

Quiz

Q. Arithmetic Operators, What is a % ?

A - Multiplication

B - Modulo

C - Division

D - Increment



Multiple Choice

Quiz

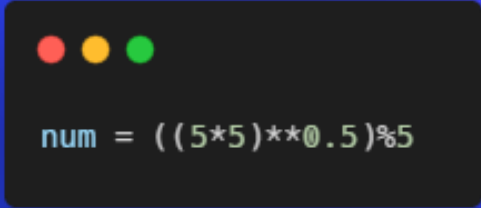
Q. What is the value of **num** ??

A - 0

B - 1

C - 5

D - 0.0



```
num = ((5*5)**0.5)%5
```



Multiple Choice

Session Content

Variables

- Definition & Working with it
- Variable naming

Data types

- Numbers
- Arithmetic operations
- **Boolean & Comparison**
- Logical & Bitwise operators
- Strings



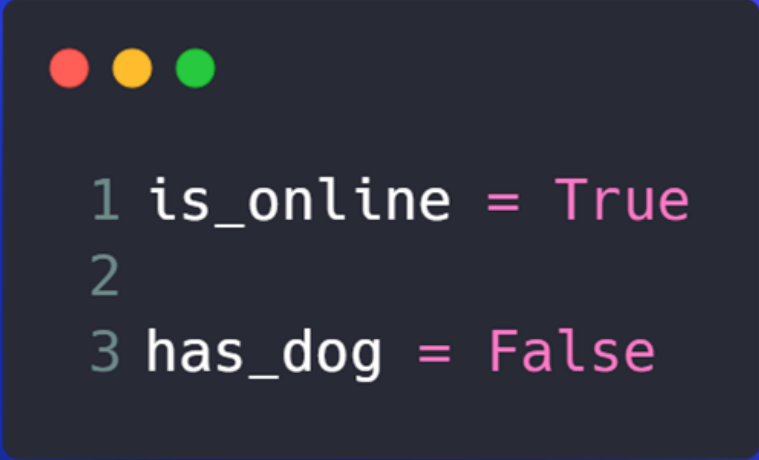
Boolean :

- Boolean refers to a system of logical thought that is used to create true/false statements.



Boolean :

- Boolean refers to a system of logical thought that is used to create true/false statements.



```
1 is_online = True
2
3 has_dog = False
```

Comparison operator :

- operators that compare values and return true or false .

Relational Operators

Operators	Meaning	Example	Result
<	Less than	$5 < 2$	False
>	Greater than	$5 > 2$	True
<=	Less than or equal to	$5 <= 2$	False
>=	Greater than or equal to	$5 >= 2$	True
==	Equal to	$5 == 2$	False
!=	Not equal to	$5 != 2$	True

Comparison operator :

- operators that compare values and return true or false .



```
1 5 == 5      # result is True
2 5 != 5      # result is False
3 10 > 7       # result is True
4 2 >= 5      # result is False
5 15 < 5      # result is False
6 3 <= 3      # result is True
```

Quiz

Q. What is the expected Output ?

A – 3.0

B – 2.0

C – 1.0

D – 4.0



```
Bool = (10%9) * (5//2)  
print(Bool)
```



Multiple Choice

Session Content

Variables

- Definition & Working with it
- Variable naming

Data types

- Numbers
- Arithmetic operations
- Boolean & Comparison
- Logical & Bitwise operators
- Strings



Logical operator :

- Logical operators are used on conditional statements (either True or False).

Python - Logical Operators

- not

x	not x
False	True
True	False

- and

x	y	x and y
False	False	False
False	True	False
True	False	False
True	True	True

- or

x	y	x or y
False	False	False
False	True	True
True	False	True
True	True	True

Operator Priority

Logical operator :

- Logical operators are used on conditional statements (either True or False).

```
1 # AND
2 1 < 2 and 2 < 3           # Result is True
3 1 != 1 and 2 < 3          # Result is False
4 1 != 1 and 2 > 3          # Result is False
5
6
7 # OR
8 1 < 2 or 2 < 3            # Result is True
9 1 != 1 or 2 < 3          # Result is True
10 1 != 1 or 2 > 3         # Result is False
11
12
13 # NOT
14 not 1 == 1               # Result is False
15 not 1 > 10              # Result is True
```

Bitwise operator :

- The bitwise operator takes two numbers as operands and does it on every bit of two numbers.

Types of Bitwise Operators

Operator	Name	Example	Result
&	Bitwise AND	6 & 3	2
	Bitwise OR	10 10	10
^	Bitwise XOR	2 ^ 2	0
~	Bitwise 1's complement	~9	-10
<<	Left-Shift	10 << 2	40
>>	Right-Shift	10 >> 2	2

Bitwise operator :

- The bitwise operator takes two numbers as operands and does it on every bit of two numbers.



```
a = 10
b = 4

# Print bitwise AND operation
print("a & b =", a & b) # 0

# Print bitwise OR operation
print("a | b =", a | b) # 14

# Print bitwise NOT operation
print("~a =", ~a) # -11

# print bitwise XOR operation
print("a ^ b =", a ^ b) # 14
```



```
a = 10
b = -10

# print bitwise right shift operator
print("a >> 1 =", a >> 1) # 5
print("b >> 1 =", b >> 1) # -5
```

Bitwise operator :

1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---

AND (&)

0	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---

OR (|)

0	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---

XOR (^)

0	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

1	0	0	1	1	1	0	1
---	---	---	---	---	---	---	---

Bitmasked Left Shift (<<)

0	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

NOT (~)

1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---

Bitmasked Right Shift (>>)

Quiz

Q. What is the expected Output ?

A – True

B – False

```
a = 10
b = 10

bool = (a > 0) and (b > 0)
print(bool)
```



Multiple Choice

Operators Precedence :

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+X, -X, ~X	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
		Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, Identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

Session Content

Variables

- Definition & Working with it
- Variable naming

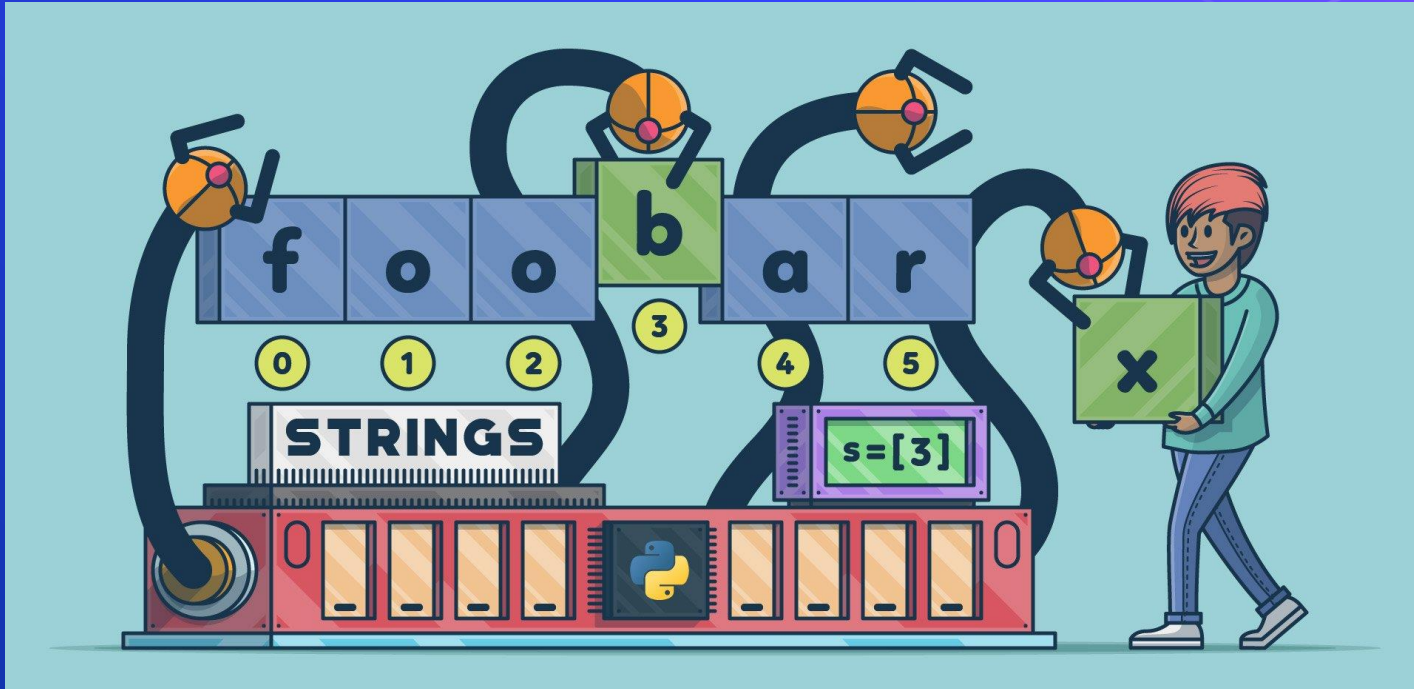
Data types

- Numbers
- Arithmetic operations
- Boolean & Comparison
- Logical & Bitwise operators
- Strings



String :

- Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes.
- Each character is encoded in the ASCII or Unicode character. So, we can say that Python strings are also called the collection of Unicode characters.



String :

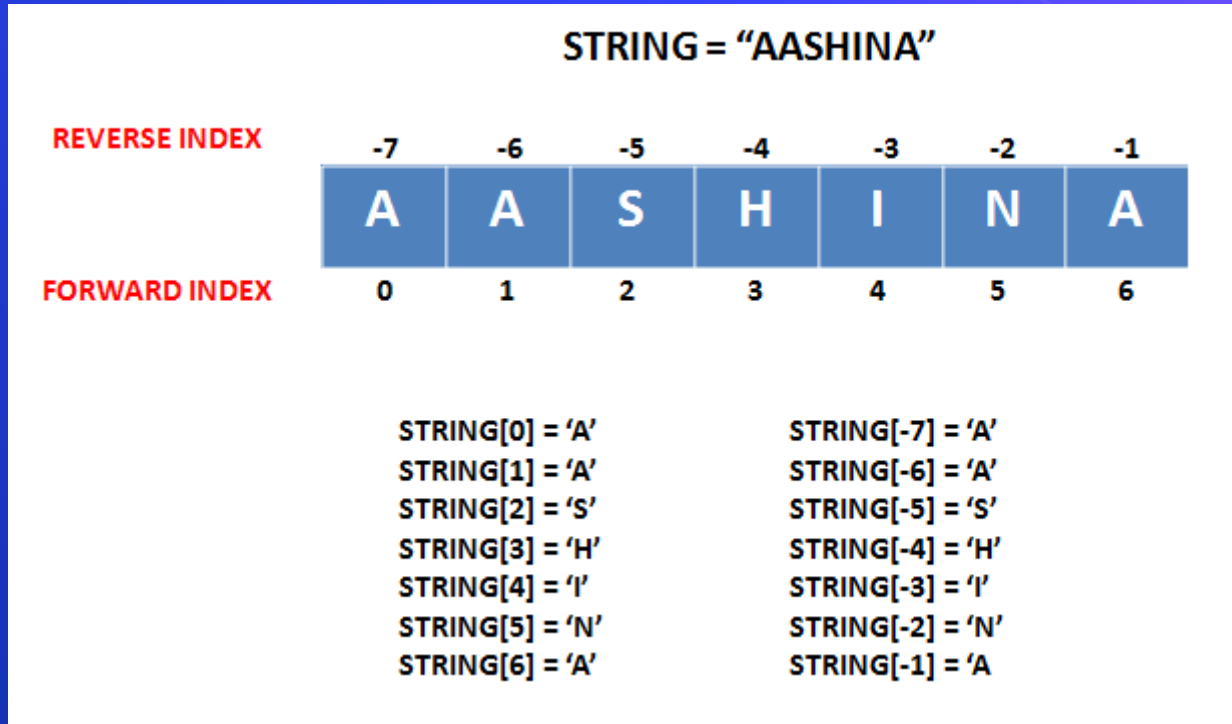
- Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes.
- Each character is encoded in the ASCII or Unicode character. So, we can say that Python strings are also called the collection of Unicode characters.

```
#Using single quotes
str1 = 'Hello Python'
print(str1)
#Using double quotes
str2 = "Hello Python"
print(str2)

#Using triple quotes
str3 = '''Triple quotes are generally used for
        represent the multiline or
        docstring'''
print(str3)
```

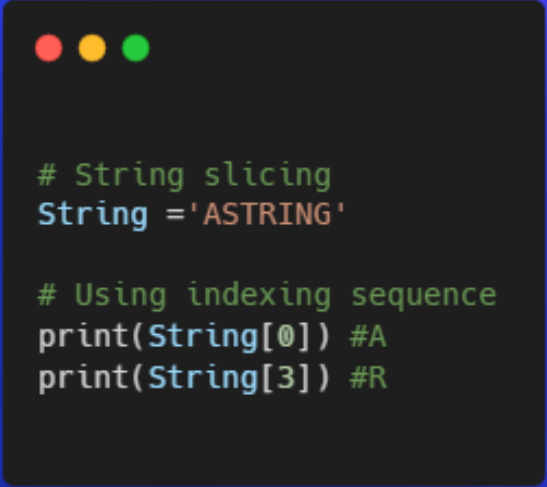
String indexing :

- Indexing allows you to access individual characters in a string directly by using a numeric value.



String indexing :

- Indexing allows you to access individual characters in a string directly by using a numeric value.



```
# String slicing
String = 'ASTRING'

# Using indexing sequence
print(String[0]) #A
print(String[3]) #R
```

String Slicing :

- You can return a range of characters by specify the start index and the end index, separated by a colon, to return a part of the string.

str = "HELLO"				
H	E	L	L	O
-5	-4	-3	-2	-1
str[-1] = 'O'		str[-3:-1] = 'LL'		
str[-2] = 'L'		str[-4:-1] = 'ELL'		
str[-3] = 'L'		str[-5:-3] = 'HE'		
str[-4] = 'E'		str[-4:] = 'ELLO'		
str[-5] = 'H'		str[::-1] = 'OLLEH'		

String Slicing :

- You can return a range of characters by specify the start index and the end index, separated by a colon, to return a part of the string.

```
# String slicing
String ='ASTRING'

# Using indexing sequence
print(String[:3]) #AST
print(String[1:5:2]) #SR
print(String[-1:-12:-2]) #GITA
```

Quiz

Q. What is the expected Output ?

- A - worl
- B - llo,
- C - world
- D - Hello



```
b = "Hello, World!"  
print(b[:5])
```



Multiple Choice

Modify Strings:

- Python has a set of built-in methods that you can use on strings.
 - `lower()`: Converts all uppercase characters in a string into lowercase.
 - `upper()`: Converts all lowercase characters in a string into uppercase.
 - `title()`: Convert string to title case.
- ✓ To see all string function : https://www.w3schools.com/python/python_strings_methods.asp

Modify Strings:

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

```
# Python3 program to show the
# working of upper() function
text = 'Epsilon AI'

# upper() function to convert
# string to upper case
print("\nConverted String:")
print(text.upper())

# lower() function to convert
# string to lower case
print("\nConverted String:")
print(text.lower())

# converts the first character to
# upper case and rest to lower case
print("\nConverted String:")
print(text.title())

# original string never changes
print("\nOriginal String")
print(text)
```

```
#OUTPUT
'''
Converted String:
EPSILON AI

Converted String:
epsilon ai

Converted String:
Epsilon Ai

Original String
Epsilon AI
'''
```


Quiz

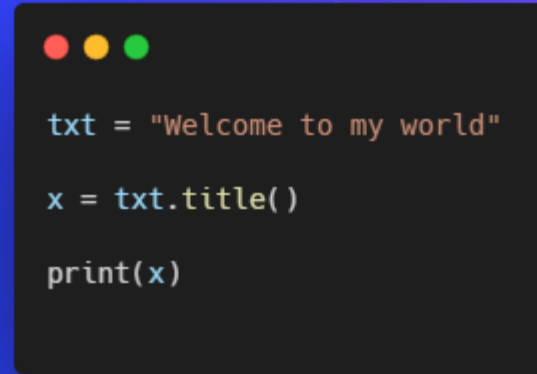
Q. What is the expected Output ?

A - WELCOME TO MY WORLD

B - Welcome To My World

C - welcome to my world

D – Welcome To my world

A dark-themed terminal window with three colored window control buttons (red, yellow, green) at the top left. It contains three lines of Python code: `txt = "Welcome to my world"`, `x = txt.title()`, and `print(x)`.

```
txt = "Welcome to my world"
x = txt.title()
print(x)
```



Multiple Choice

Modify Strings:

- Python has a set of built-in methods that you can use on strings.
 - `replace()`: returns a copy of the string where all occurrences of a substring is replaced with another substring.
 - `split()`: return a list of the words of the string, If the optional second argument `sep` is absent or `None`
 - `strip()`: It returns a copy of the string with both leading and trailing white spaces removed
- ✓ To see all string function : https://www.w3schools.com/python/python_strings_methods.asp

Modify Strings:

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

```
Python3 program to demonstrate the  
# use of replace() method  
  
string = "I love pyhon"  
  
# Prints the string by replacing all  
print(string.replace("pyhon", "Java"))
```

```
# OUTPUT  
...  
  
I love Java  
  
...
```

Modify Strings:

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

```
• • •  
  
# Python3 program to demonstrate the  
# use of split() method  
  
line = "I Love Python"  
print(line.split())
```

```
• • •  
  
# OUTPUT  
...  
  
['I', 'Love', 'Python']  
  
...
```

Modify Strings:

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

```
# Python3 program to demonstrate the use of
# strip() method

string = "    I love python    "

# prints the string without stripping
print(string)

# prints the string by removing leading and trailing whitespaces
print(string.strip())
```

```
# OUTPUT
'''
    I love python
I love python
'''
```

Quiz

Q. A function used to return a trimmed version of the string

A – strip()

B – upper()

C – split()

D – replace()



Multiple Choice

String Concatenation:

- To concatenate, or combine, two strings you can use the + operator.



```
# Python3 program to demonstrate the  
# use of Concatenation
```

```
a = "Hello"  
b = "World"  
c = a + b  
print(c)
```



```
# OUTPUT  
...
```

```
HelloWorld
```

```
...
```

String Format :

- To concatenate, or combine, two strings you can use the + operator.

```
# Python3 program to demonstrate  
# the string format  
  
Age = 19  
# formatting a string using a numeric constant  
  
print(f"Hello, I am {Age} years old !")
```

```
# OUTPUT  
...  
  
Hello, I am 19 years old !  
  
...
```


Escape Characters:

Escape Sequence	DESCRIPTION
<code>\newline</code>	Backslash and newline ignored
<code>\\</code>	Backslash
<code>\'</code>	Single quote
<code>\"</code>	Double quote
<code>\a</code>	ASCII Bell
<code>\b</code>	ASCII Backspace
<code>\f</code>	ASCII Form feed
<code>\n</code>	ASCII Linefeed
<code>\r</code>	ASCII Carriage Return
<code>\t</code>	ASCII Horizontal Tab
<code>\v</code>	ASCII Vertical Tab
<code>\ooo</code>	Character with octal value ooo
<code>\xHH</code>	Character with hexadecimal value HH

Escape Characters:

- To concatenate, or combine, two strings you can use the + operator.

```
# sample string
s = "I love to use \t instead of using 4 spaces"

# normal output
print(s)

# doubling line backslashes
s = "I love to use \\t instead of using 4 spaces"

# output after doubling
print(s)
```

```
# OUTPUT
...

I love to use      instead of using 4 spaces
I love to use \t  instead of using 4 spaces

...
```

Quiz

Q. What is the expected Output ?

A – I Love Python

B – Ilove python

C – Python I love

D - ilovepython



```
a = "I"  
b-= "Love"  
c= "Python"
```

```
st = a+b+c  
print(st)
```



Multiple Choice

Questions ?!



Thanks!

