Chapter Two
Python Fundamentals

By Kibret Zewdu

Department of Computer Science
College o Informatics

University of Gondar

Token

- Smallest individual unit in a program is known as **token**.
  1. Keywords
  2. Identifiers
  3. Literals
  4. Operators
  5. punctuators

Keywords

- Reserve word of the compiler/interpreter which can't be used as identifier.
- Are identifiers that cannot be used as ordinary identifiers. They must be spelled exactly as written here

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

Identifiers

- A Python identifier is a name used to identify a variable, function, class, module or other object.
  - An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
  - Python does not allow special characters
  - Identifier must not be a keyword of Python.
  - Python is a case sensitive programming language.
    - Thus, **Rollnumber** and **rollnumber** are two different identifiers in Python.
- Some valid identifiers : Mybook, file123, z2td, date_2, _no
- Some invalid identifier : 2rno, break, my.book, data-cs

Identifiers – (Continue…)

- Some additional naming conventions

1. **Class** names start with an uppercase letter. All other identifiers start with a lowercase letter.
2. Starting an identifier with a *single leading underscore* indicates that the identifier is **private**.
3. Starting an identifier with *two leading underscores* indicates a ***strong private identifier***.
4. If the identifier also ends with *two trailing underscores*, the identifier is a language-defined special name.

Literals

- Literals in Python can be defined as number, text, or other data that represent values to be stored in variables.
- Example of String Literals in Python

```
    name = 'Johni'    ,    fname ="johny"
```

- Example of Integer Literals in Python(numeric literal)

```
    age = 22
```

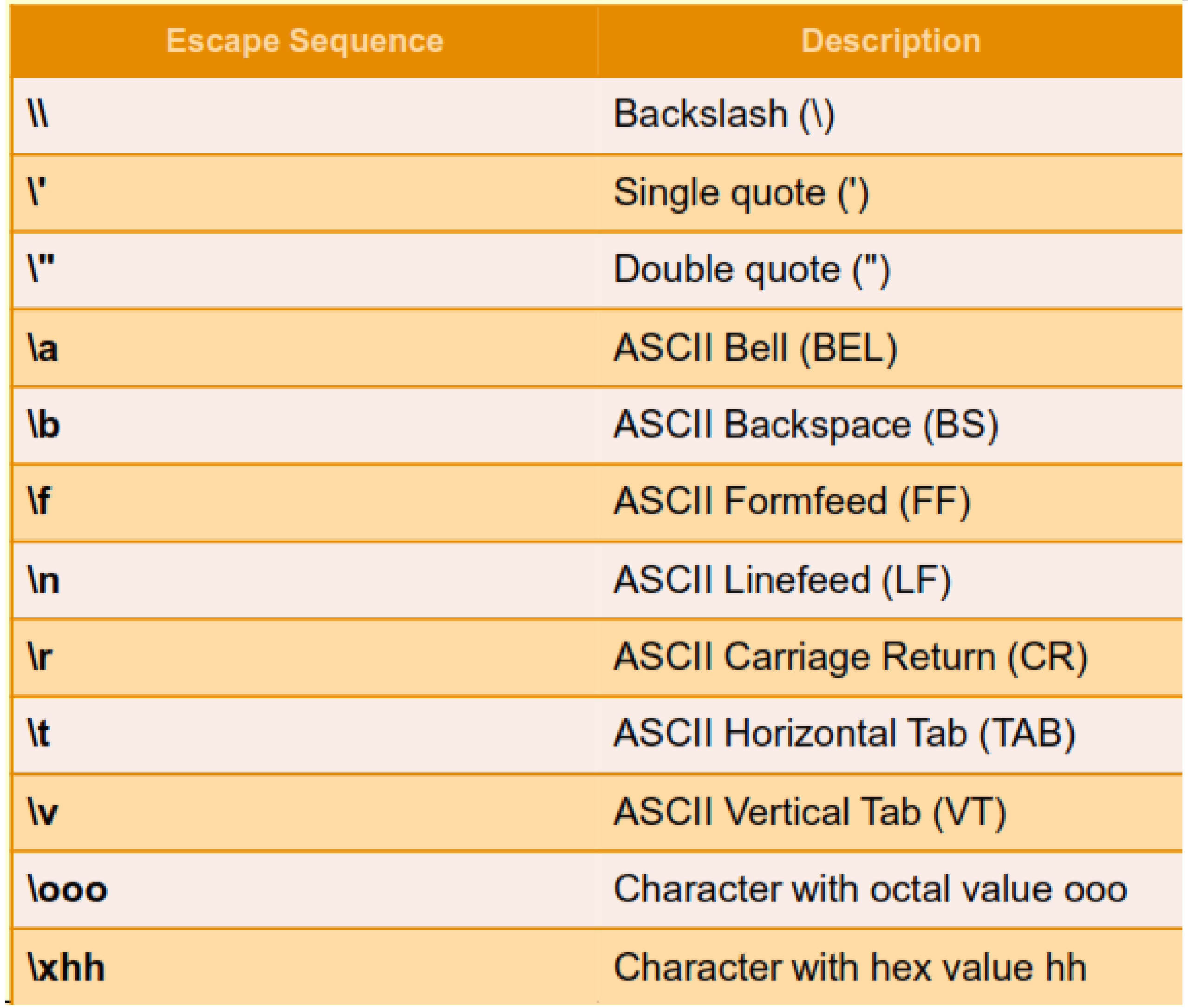
- Example of Float Literals in Python(numeric literal)

```
    height = 6.2
```

- Example of Special Literals in Python

```
    name = None
```

- Escape sequence



Escape Sequence	Description
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\v	ASCII Vertical Tab (VT)
\ooo	Character with octal value ooo
\xhh	Character with hex value hh

Operators

- symbols that are used to perform operations on operands.
- Types of Operators
    1. Arithmetic Operators.
    2. Relational Operators.
    3. Assignment Operators.
    4. Logical Operators.
    5. Bitwise Operators
    6. Membership Operators
    7. Identity Operators

Operators (continue...)

1. Arithmetic Operators

- used to perform arithmetic operations like addition, multiplication, division etc.

Operators	Description	Example
+	perform addition of two number	a + b
-	perform subtraction of two number	a – b
/	perform division of two number	a / b
*	perform multiplication of two number	a * b
%	Modulus = returns remainder	a % b
//	Floor Division = remove digits after the decimal point	a // b
**	Exponent = perform raise to power	a ** b

Operators (continue...)

2. **Relational Operators**

- Relational Operators are used to compare the values.

Operators	Description	Example
==	Equal to, return true if a equals to b	a == b
!=	Not equal, return true if a is not equals to b	a != b
>	Greater than, return true if a is greater than b	a > b
>=	Greater than or equal to , return true if a is greater than b or a is equals to b	a >= b
<	Less than, return true if a is less than b	a < b
<=	Less than or equal to , return true if a is less than b or a is equals to b	a <= b

Operators (continue...)

3. **Assignment Operators**

- Used to assign values to the variables.

Operators	Description	Example
=	Assigns values from right side operands to left side operand	a=b
+=	Add 2 numbers and assigns the result to left operand.	a+=b
/=	Divides 2 numbers and assigns the result to left operand.	a/=b
*=	Multiply 2 numbers and assigns the result to left operand.	a*=b
-=	Subtracts 2 numbers and assigns the result to left operand.	a-=b
%=	modulus 2 numbers and assigns the result to left operand.	a%=b
//=	Perform floor division on 2 numbers and assigns the result to left operand.	a//=b
**=	calculate power on operators and assigns the result to left operand.	a**=b

Operators (continue...)

4. Logical Operators

- Logical Operators are used to perform logical operations on thegiven two variables or values.

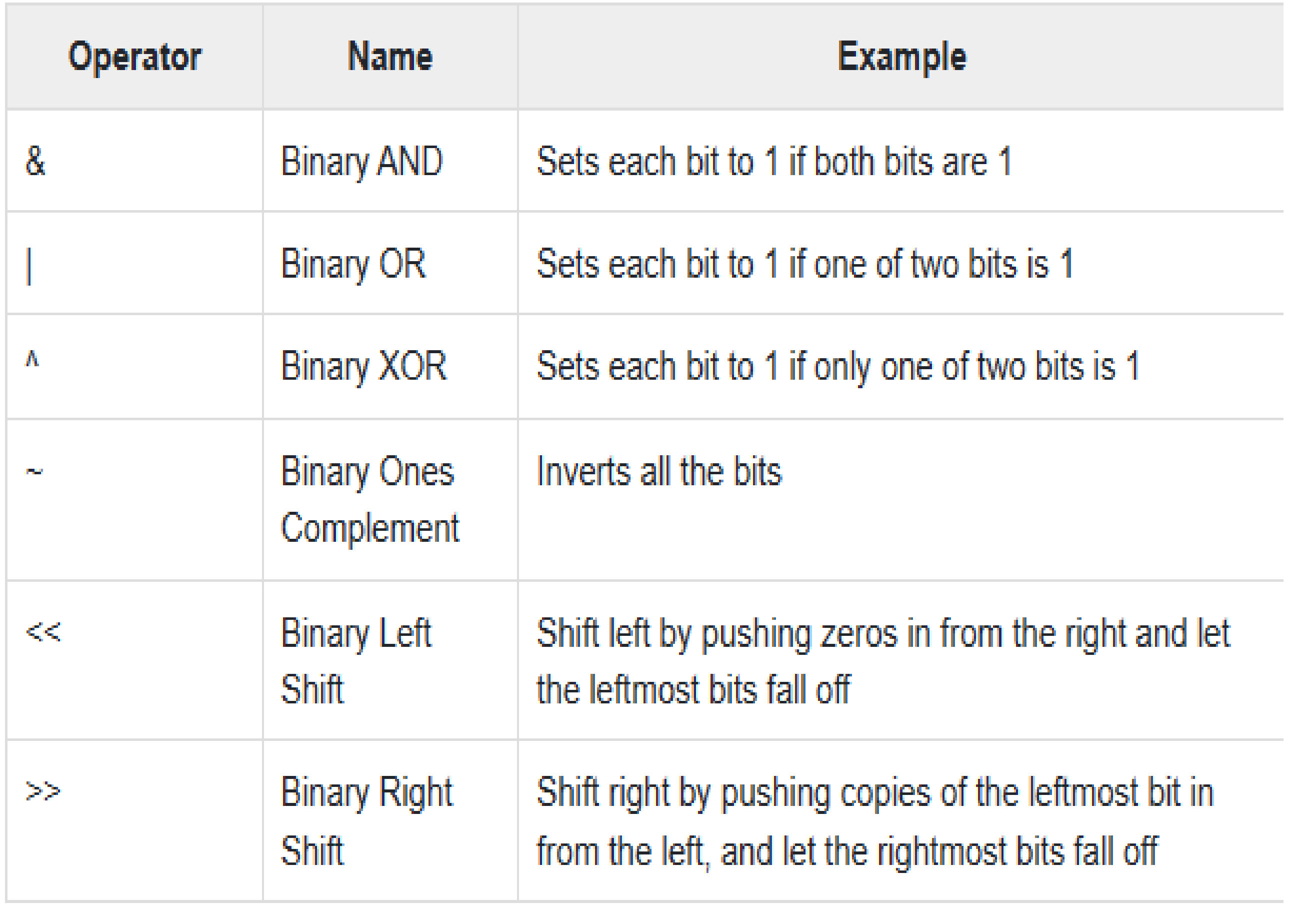
Operators	Description	Example
and	return true if both condition are true	x and y
or	return true if either or both condition are true	x or y
not	reverse the condition	not(a>b)

5. **Bitwise Operators**

- Bitwise operator works on bits and performs bit by bit operation.

- Assume if a = 60; and b = 13.

```
a = 0011 1100
b = 0000 1101
--------------------------
a&b = 12 (0000 1100)
a|b = 61 (0011 1101)
a^b = 49 (0011 0001)
~a  = -61 (1100 0011)
a << 2 = 240 (1111 0000)
a>>2 = 15 (0000 1111)
```



Operator	Name	Example
&	Binary AND	Sets each bit to 1 if both bits are 1
|	Binary OR	Sets each bit to 1 if one of two bits is 1
^	Binary XOR	Sets each bit to 1 if only one of two bits is 1
~	Binary Ones Complement	Inverts all the bits
<<	Binary Left Shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Binary Right Shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Operators (continue...)

6. **Membership Operators**

- The membership operators in Python are used to validate whether a value is found within a sequence such as such as strings, lists, or tuples.

Operators	Description	Example
in	return true if value exists in the sequence, else false.	a in list
not in	return true if value does not exists in the sequence, else false.	a not in list

Operators (continue...)

7. **Identity Operators**

- Identity operators in Python compare the memory locations of two objects.

Operators	Description	Example
is	returns true if two variables point the same object, else false	a is b
is not	returns true if two variables point the different object, else false	a is not b

Operators Precedence

Operators	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Structure of Python Programs

- A python program contain the following components
    a) Expression - which is evaluated and produce result. E.g. (20 + 4) / 4
    b) Statement - instruction that does something.
       E.g
       salary = 7500.0
       print(“Hello World!”)
    c) Comments - which is readable for programmer but ignored by python interpreter
    d) Function - a piece of reusable code
    e) Block & indentation - group of statements is block. ndentation at same level create a block.

Comments in Python

- A comment is a programmer-readable explanation or annotation in the Python source code.
- They are added with the purpose of making the source code easier for humans to understand, and are ignored by Python interpreter.
- There are three types of comments available in Python
    1. Single line Comments
    2. Multiline Comments
    3. Docstring Comments

Comments in Python (Continue…)

1. Single Line Comments

- A hash sign (#) that is not inside a string literal begins a comment.
- All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.
- Example:

```
# This is a single line comment in python
print ("Hello, World!")
```

Produces

```
Hello, World!
```

- can type a comment on the same line after a statement or expression

```
price = 23.75 # This is also another comment
```

Comments in Python (Continue...)

2. Multi-Line Comments

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

- **Triple-quoted** string is also ignored by Python interpreter and can be used as a multiline comments.

```
'''
This is a multiline comment.
''‘
print ("Hello, World!")
```

Comments in Python (Continue...)

3. Docstring Comments

- Python **docstrings** provide a convenient way to provide a help documentation with Python modules, functions, classes, and methods.
- The **docstring** is then made available via the __doc__ attribute.

```
def add(a, b):
   """Function to add the value of a and b"""
   return a+b

print(add.__doc__)
```

This produces the following result:

```
Function to add the value of  a and b
```

Variables

- Python variables are the *reserved memory locations used to store values* with in a Python Program.
- Based on the data type of a variable, Python interpreter allocates memory and decides what can be stored in the reserved memory.
  - Therefore, by assigning different data types to Python variables, you can store integers, decimals or characters in these variables.
- A Python variable is created automatically when you assign a value to it.
  - The equal sign (=) is used to assign values to variables.

```
counter = 100 # Creates an integer variable
miles = 1000.0 # Creates a floating point variable
```

- **Multiple Assignment**: assign a single value to many variables

```
a = b = c = 1 # single value to multiple variable. All three
              #variables are assigned to the same memory location

a,b = 1,2 # multiple value to multiple variable
a,b = b,a # value of a and b is swapped
```

Variables (Cont...)

- Every Python variable should have a *unique name* like a, b, c.
- A variable name can be meaningful like color, age, name etc.
- There are certain rules which should be taken care while naming a Python variable:
    - A variable name must start with a letter or the underscore character
    - A variable name cannot start with a number or any special character like $, (, * % etc.
    - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
    - Python variable names are case-sensitive which means Name and NAME are two different variables in Python.
    - Python reserved keywords cannot be used naming the variable.

Variables (Cont...)

- **Local Variable**
  - Python Local Variables are defined inside a function.
  - We cannot access variable outside the function.

```
def sum(x,y):
        sum = x + y

        return sum


print(sum(5, 10))
```

Variables (Cont...)

- **Global Variable**
  - Any variable created outside a function can be accessed within any function and so they have global scope.

```
x = 5
y = 10
def sum():
    sum = x + y
    return sum

print(sum())
```

Dynamic typing

- Data type of a variable depend/change upon the value assigned to a variable on each next statement.

```
X = 25  # integer type
X = “python” # x variable data type change to string on just next line
```

- Now programmer should be aware that not to write like this:

```
Y = X / 5  # error !! String cannot be divided
```