Lecture 2:
Variable Names, Keywords, Operators and Operands

Objectives

- After completing the lesson, the student will be able to:
  - Identify Python reserved keywords from a list of names.
  - Define the term variable. Understand the rules that should follow when naming identifies.
  - Write simple programs using variables.
  - Explain how expressions are formed and its parts.
  - List and use arithmetic operators, relational operators and logical operators in programming statements.

Reserved keywords

- Some identifiers are part of Python itself – called *reserved words*

and	del	from	not
while	as	elif	global
or	with	assert	else
if	pass	yield	break
except	import	print	class
exec	in	raise	continue
finally	is	return	def
for	lambda	try	

Identifiers

- Identifiers refer to the names of variables, functions,
- arrays etc. created by the programmer.

- Rules for naming identifiers
  - Only alphabetic characters, digits and underscores are permitted.
  - The name cannot start with a digit.
  - Uppercase and lowercase letters are distinct.
  - A declared keyword cannot be used as a variable name.
  - An identifier cannot contain any spaces.

Variables

- A variable is a data name that may be used to store a data value.
- Variable names may consist of letters, digits and the underscore character.
- The variable name should be short and meaningful.
- Example:
  - `x = 56`

Variables - Example

- Calculate the sum of two numbers

```
a = 23
b = 3
print("Sum of the two variables = ", a + b)
```

Sum of the two variables = 26

- The above programming statements assigns integer values to A and B and then calculates the sum.

Variables - Example

```
value =4
print(value)
value = 10
print(value)
```

Assign *value*

Replace *value*

Numeric Data Types

- The data type of an object determines what values it can have and what operations can be performed on it.
- Whole numbers are represented using the integer data type (int)
- Numbers that have fractional parts are represented by using float, long and complex data type.
- Type function tells us the data type of any value.
- Example:
  - type(3)
    - <type 'int'>

Numeric Data Types

- The four distinct numeric types:
  - Plain integers
  - Long integers
  - Floating point numbers
  - Complex numbers.

- Plain integers have 32 bits of precision.
- Floating points have 64 bits of precision.
- Long integers have unlimited precision.

Type Conversion

- Combining an `int` with an `int` produces an `int` (except for division).
  - Example: `5 + 2` ➔ `7`

- Combining a `float` with a float creates another `float`. Dividing two numbers also produce a `float`
  - Example:
  - `5.0 + 2.0` ➔ 7.0
  - `5 / 2` ➔ 2.5

- In mixed-typed expressions, `int` will be automatically converted to `float` and perform floating point operations to produce a `float` result.
  - Example: `5.0 + 2` ➔ 7.0

Type Conversion

- Explicit type conversion – convert data to a specific type by using `int()`,`long()` and `float()`.

- Example:
  - `sum, n = 22 , 4`
  - `average = sum / n`
  - `print(average)`
  - `average = int(average)`
  - `print(average)`

Type Conversion: Examples

- `int(4.5)`
- `long(3.9)`
- `float(int(3.3))`
- `int(float(3.3))`
- `int(float(3))`

Type Conversion: Examples

- `int(4.5)`
  4
- `long(3.9)`
  3L
- `float(int(3.3))`
  3.0
- `int(float(3.3))`
  3
- `int(float(3))`
  3

Operators and Operands

- An operator is a symbol that tells the computer to perform certain mathematical or logical operations.

- An operand may be a constant, a variable or combination of them.

- Example:
  - `2 + a`
  - `b / c + 4`
- In the above examples 2, a, b, c and 4 are operands and + and / are operators

Arithmetic Operators

- Basic arithmetic operators:

Operator	Name	Example	Result
+	Addition	2+3	5
-	Subtraction	5-4	1
*	Multiplication	3*7	21
/	Division	5/2	2.5
//	Integer Division	5//2	2
%	Remainder (division)	8%5	3
**	exponentiation	2**8	256

Arithmetic Expressions

- If both the operands in an arithmetic expression are integers, the result is an integer, except for division.
- Example:
  - 7 + 3 = 10
  - 7 – 3 = 4
  - 7 * 3 = 21
  - 7 // 3 = 2
  - 7 / 3 = 2.3333333333333335
  - 7 % 3 = 1

- If both the operands in an arithmetic expression are float, the result is always of type float.
- Example:
  - 7.5 + 3.2 = 10.7
  - 7.5 - 3.2 = 4.3
  - 7.5 * 3.2 = 24.0
  - 7.5 / 3.2 = 2.34375
  - 7.5 % 3.0 = 1.5

Arithmetic Expressions

- All the arithmetic operators can accept a mix of integer and float operands.
- If one or both of the operands are float then the result will be float.
- Example:
  - 2.5 + 3 = 5.5
  - 2.5 + 3.5 = 6.0
  - 5 * 3.2 = 16.0
  - 3 / 2.5 = 1.2
  - 3 % 2.5 = 0.5

Relational Operators

- Used to compare numeric quantities and it evaluates True or False.

Operator	Name	Example	Result
<	Less than	5<2	False
<=	Less than or equal to	5<=2	False
==	Equal to	5==2	False
>	Greater than	5>2	True
>=	Greater than or equal to	5>=2	True
!=	Not equal to	5!=2	True

Relational Expressions

- General syntax:
  - expression relational operator expression
  - Expression can be a constant, variable or combination of them.
- Example:
  - `3 < 4`
  - `3*4 < 3+4`
- Characters are valid operands since they are represented by numeric values.
  - Example: `'A' < 'C'`

Logical Operators

- Logical operators are used to combine relational expressions and it evaluates to True or False.

Operator	Name	Example	Result
and	Logical AND	10>5 and 2>1	True
or	Logical OR	10>5 or 2>12	True
not	Logical NOT	not (5>2)	False

Logical Operators

- `not` - if the result of its single operand is 1 it produces 0, otherwise it produces 1.

- `and` – produces 0 if one or both of its operands evaluate to 0, otherwise 1 is the result.

- `or` - produces 0 if both of its operands evaluate to 0, otherwise it produces 1.

Logical Expressions

- Example:
  - `marks = 40`
  - `attendance = 81`
  - `marks>45 and attendance>80`
- The result of the above expression is false.

- Example:
  - `marks = 45`
  - `attendance = 81`
  - `marks>=45 or attendance>=80`
- The result of the above expression is true

Assignment Statements

- Equal (=) is used to assign values to variables
- General Syntax:
  - `variable = expression`
- Example: `a = x + 3`
  - Here a is the variable and x + 3 is an expression.
- Example:
  - `b = 4`
  - `c = 6`
  - `result = b + c`
- In this case, first it evaluates the expression on right and then the value is assigned to the variable on left.

Assignment Statements

- A variable can be assigned many times.
- Example:
  - `a = 5`
  - `print(a)`

  The above statement prints **5** as the value of a.

  - `a = 10`
  - `print(a)`

  The above statement prints **10** as the value of a.

Assigning Input

- input statement is used to get data from the user and store it into a variable.
- General Syntax:
  - `variable = input("Prompt/message: ")`
- Example:
  - `username = input("Enter your name: ")`

- In the above example, it will prompt to enter the user's name and the input value will be assigned to the variable `username`.

Assigning Input – Numbers (Integers)

- Example:
  - `num1 = int(input("Enter an first number: "))`
  - `num2 = int(input("Enter an second number: "))`
  - `print(num1 + num2)`

- In the above example, the first line will:
  - prompt the user to enter the first number.
  - Convert it to an integer (int) and assign it to variable num1.
- The second line does the same for the second number.
- Then the sum of the two numbers typed at the keyboard are printed on the screen.

Simultaneous Assignment

- Assign several values all at the same time.
- General Syntax:
  - `variable1, variable2 = expression1, expression2`

- Evaluate all the expressions on the right-hand side and then assign these values to the corresponding variables on the left-hand side.
- Example:
  - `add, difference = 5+3, 5-3`

Simultaneous Assignment

- Example: swapping two values
  - `a = 5`
  - `b = 15`

  - `temp = a`
  - `a = b`
  - `b = temp`

Simultaneous Assignment

- Example: swapping two values
  - `a = 5`
  - `b = 15`

variables	*a*	*b*	*temp*
initial values	5	15	No value yet

  - `temp = a`

values now	5	15	5

  - `a = b`

values now	15	15	5

  - `b = temp`

values now	15	5	5

- The above code can be replaced by the following statement
  - `a, b = b, a`