Variable, Expression and Statement

Python Programming
CT108-3-1-PYP

Variables

- A variable is a container for storing a value.
- It is a placeholder where we can store data and later retrieve the data using the variable “name”.
- The variable names can be defined as we desire.

π Constants

- Is a type of variable whose value cannot be changed.
- Constants are like containers that hold information which cannot be changed later.
- Refer to names associated with values that never change during a program’s execution.
- Fixed values such as numbers, letters, and strings are called “constants” - because their value does not change.
- Constants are usually declared and assigned in a module

Declaring and Assigning a value to a Constant

Create python file name constant.py

```
PI = 3.14
```

Create python file name week4.py

```
import constant
print(constant.PI)
```

Rules and Naming Convention for Variables and constants

- Should have a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (_).
- Create a name that makes sense or with meaningful name
- Variable name that having two words, use underscore to separate them.
- Use capital letters to declare a constant.
- Never use special symbols like !, @, #, $, %, etc.
- Do not start a variable name with a digit(numbers).
- Cannot use keywords / reserve words.

Variable - Example

- **Good**: spam, eggs, spam23, _speed
- **Bad:** 23spam, #sign, var.12
- **Different:** spam, Spam, SPAM

Keywords or Reserved Words

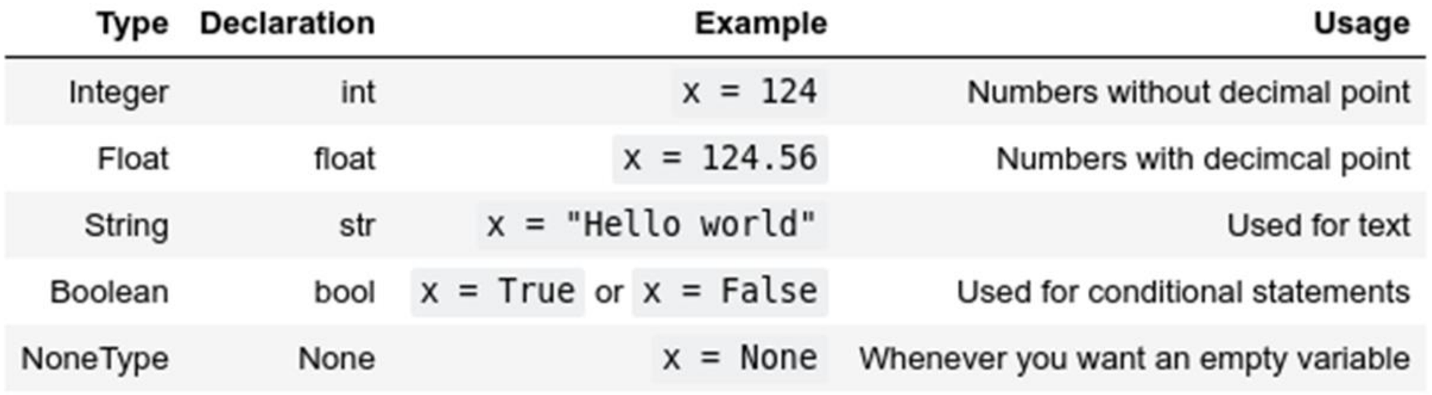
- Keywords are words reserved by python that have predefined meaning. You can not use reserved words as variable names / identifiers

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

Types

Variables have a type, which defines the way it is stored.

The basic types are:



Type	Declaration	Example	Usage
Integer	int	`x = 124`	Numbers without decimal point
Float	float	`x = 124.56`	Numbers with decimcal point
String	str	`x = "Hello world"`	Used for text
Boolean	bool	`x = True` or `x = False`	Used for conditional statements
NoneType	None	`x = None`	Whenever you want an empty variable

Type

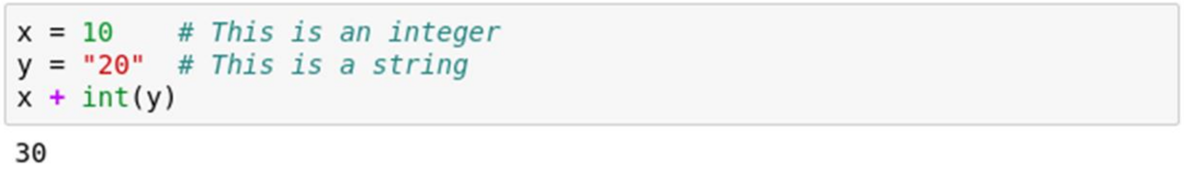
- In Python variables and constants have a “type”
- Python knows the difference between an integer number and a string
- For example, “+” means “addition” if something is a number and “concatenate” if something is a string
- We can't do arithmetic operations on variables of different types. Therefore, make sure that you are always aware of your variable's types

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

Casting types

Luckily, Python offers us a way of converting variables to different types!

Casting – the operation of converting a variable to a different type



```
x = 10    # This is an integer
y = "20"  # This is a string
x + int(y)
```

```
30
```

Similar methods exist for other data types:
int(), **float()**, **str()**

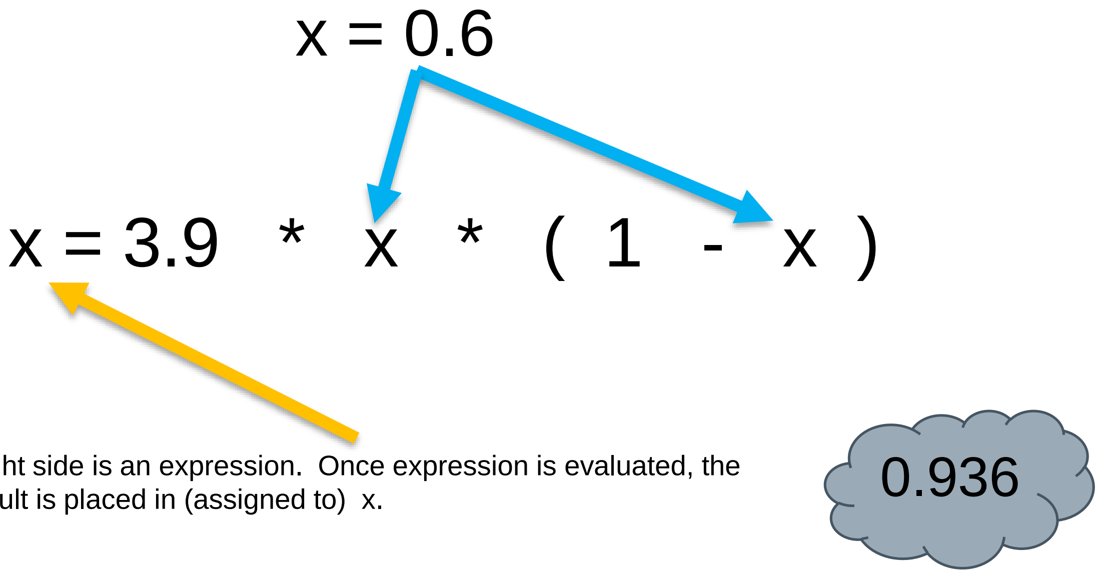
Assignment Statements

- We assign a value to a variable using the assignment statement (=)
- An assignment statement consists of an expression on the right-hand side and a variable to store the result

```
x = 3.9 * x * ( 1 - x )
```

Assignment Statements - Example

A variable is a memory location used to store a value (0.6).



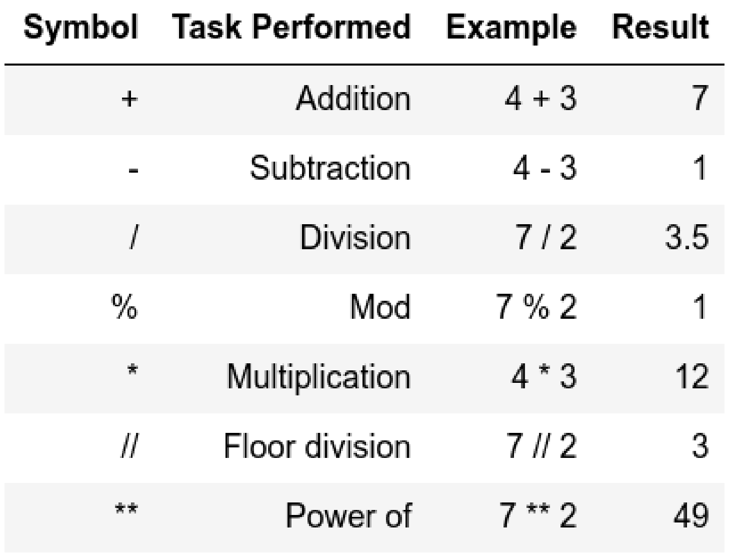
x = 0.6

x = 3.9 * x * (1 - x)

Right side is an expression. Once expression is evaluated, the result is placed in (assigned to) x.

Arithmetic / Numeric operations

- Like actual Mathematics.
- Order of precedence is the same as in Mathematics.
- We can also use parenthesis ()
- Because of the lack of mathematical symbols on computer keyboards - we use “computer-speak” to express the classic math operations
- Asterisk is multiplication
- Exponentiation (raise to a power) looks different from in math.



Symbol	Task Performed	Example	Result
+	Addition	4 + 3	7
-	Subtraction	4 - 3	1
/	Division	7 / 2	3.5
%	Mod	7 % 2	1
*	Multiplication	4 * 3	12
//	Floor division	7 // 2	3
**	Power of	7 ** 2	49

More Sample

- Exponentiation: 2**3 is   8
- Integer division (rounds down): 5//2   is   2
- Modulus (gives the remainder): 28 % 5   is   3

Expressions

- Expression: A data value or set of operations to compute a value.
  - Examples: 1 + 4 * 3
    - 13
- Precedence: Order in which operations are computed.
  - * / % ** have a higher precedence than + - 1 + 3 * 4 is 13
  - Parentheses can be used to force a certain order of evaluation.
    - (1 + 3) * 4 is 16

Order Precedence Rules

- Highest precedence rule to lowest precedence rule
  - Parenthesis are always respected
  - Exponentiation (raise to a power)
  - Multiplication, Division, and Remainder
  - Addition and Subtraction
  - Left to right
  - Top to bottom approach

Parenthesis
Power
Multiplication / division/
Modulus
Addition
Left to Right

Relational Operators

Table 1 Relational Operators

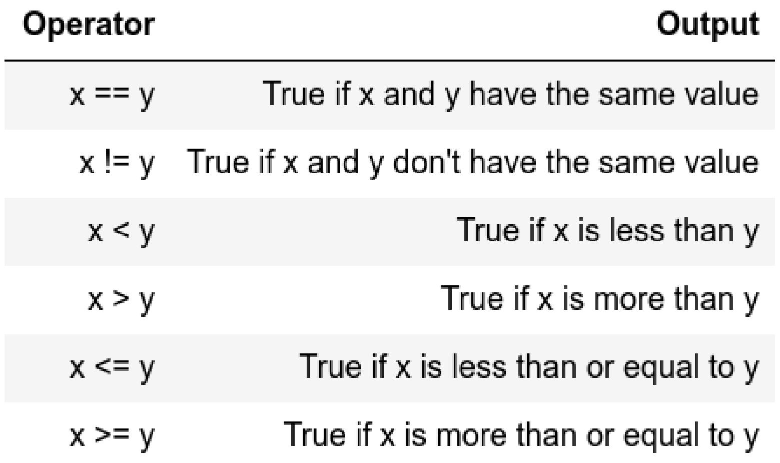
Python	Description
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
==	Equal
!=	Not equal

Table 2 Relational Operator Examples

Expression	Value
2 < 2	False
2 <= 3	True
5 > 6	False
5 >= (3-1)	True
5 == 6	False
5 != 6	True

π Relational operators

- Return Boolean values i.e., True or False
- Used extensively for conditional statements



Operator	Output
x == y	True if x and y have the same value
x != y	True if x and y don't have the same value
x < y	True if x is less than y
x > y	True if x is more than y
x <= y	True if x is less than or equal to y
x >= y	True if x is more than or equal to y

Relational examples

```
x = 5         # assign 5 to the variable x
x == 5        # check if value of x is 5
```

```
True
```

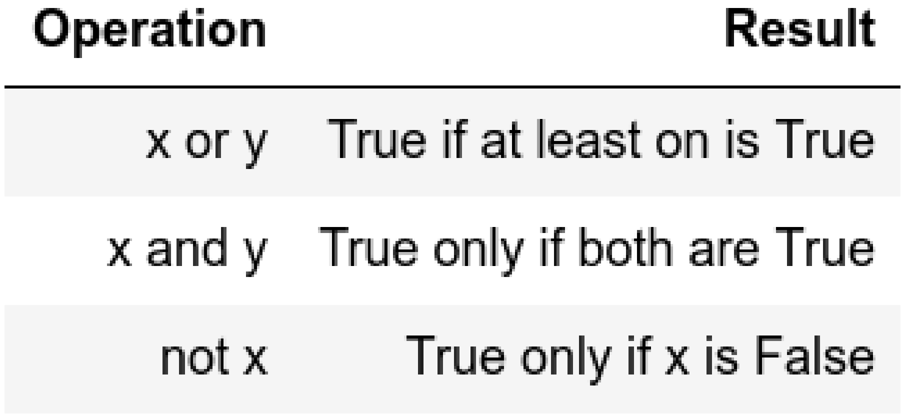
Note that `==` *is not the same as* `=`

```
x > 7
```

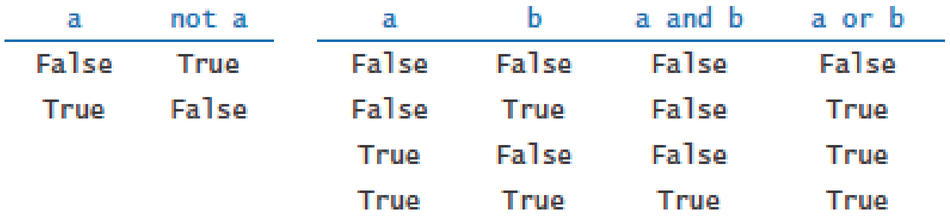
False

Logical operators

- Allows us to extend the conditional logic



Operation	Result
x or y	True if at least on is True
x and y	True only if both are True
not x	True only if x is False

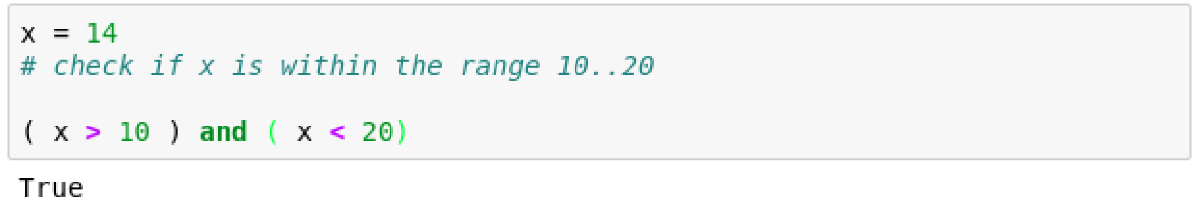


a	not a
False	True
True	False

a	b	a and b	a or b
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

Truth-table definitions of bool operations

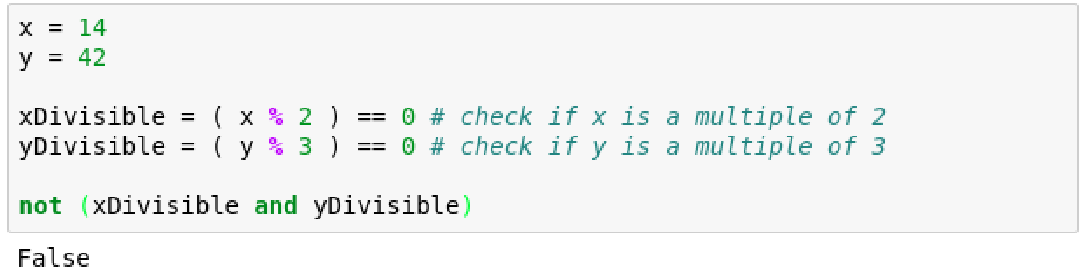
Combining both



```
x = 14
# check if x is within the range 10..20

( x > 10 ) and ( x < 20)
```

```
True
```



```
x = 14
y = 42

xDivisible = ( x % 2 ) == 0 # check if x is a multiple of 2
yDivisible = ( y % 3 ) == 0 # check if y is a multiple of 3

not (xDivisible and yDivisible)
```

```
False
```

Exercise

```
a = 10

b = a

c = "Your result is: "

print(b)
```

It will print out 10.

When you set one variable equal to another, they don’t become linked; b is set to 10 and no longer has anything else to do with a.