LECTURER: TAI LE QUY

Introduction to Programming with Python

Modules and Packages

6

UNIT 3

Statements

STUDY GOALS

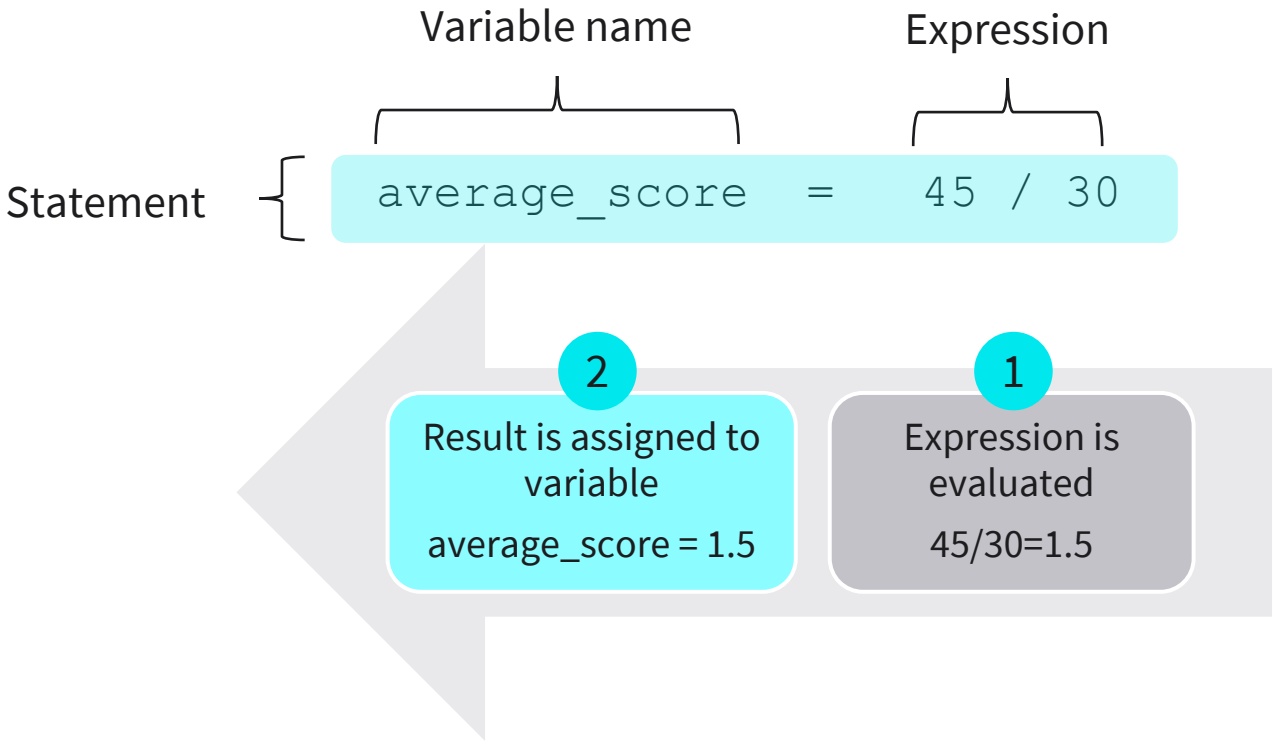
- Use basic assignments and expressions
- Learn how to use various conditional statements
- Implement loops
- Apply iterators and comprehensions

EXPLAIN SIMPLY

1. What is a condition statement?
2. What is a loop statement for?
3. What are comprehensions?

Expression & statement

In Python, an expression consists of values and operators and evaluates to a single value. In a statement, an expression sits on the right-hand side of the '=' sign and is evaluated first. The result is then assigned to a variable.



Expression evaluation

Expressions are evaluated in the following order: Parentheses, Exponents, Multiplication, Division, Addition and Subtraction - PEMDAS.

Example: 2**4*(3+5)+5 = 2**4*8+5 =16*8+5=128+5=133

Assignment operators

Operator	Syntax	Meaning
=	`a = 28`	Assign 28 to variable a
+=	`a += 5`	Add 5 to variable a and assign the result back to a i.e., `a=a+5`
-=	`a -= 5`	Subtract 5 from variable a and assign the result back to a i.e., `a=a-5`
*=	`a *= 5`	Multiply variable a by 5 and assign the result back to a i.e., `a=a*5`
/=	`a /= 5`	Divide variable a by 5 and assign the result back to a i.e., `a=a/5`
%=	`a %= 5`	Get the remainder of a divided by 5 and assign the result to a i.e., `a=a%5`
**=	`a **= 5`	Raise a to the power of 5 and assign the result back to a i.e., `a=a**5`

Chaining

A chained statement or assignment assigns value to multiple variables simultaneously.

Example: The chained statement `a=b=c=d=5` simultaneously assigns 5 to variables a, b, c and d so that `a=5`, `b=5`, `c=5` and `d=5`.

The `print` statement can chain outputs together. The outputs can be strings, expressions which are separated by commas.

Example:
```
>>> print('a =', 2*5, ' is a python statement')
>>> a = 10 is a python statement
```

Definition

A conditional statement tells a program what to do if a certain condition is met (i.e., evaluates to True).

Example:

```
if a <= b:
    a*=2
```

If a is less than or equal to b ← condition, True or False

double the value of a and assign the result to a ← what to do if True

Comparison operators

Operator	Expression	Meaning	Example a=5, b=8
==	`a == b`	Check if a is equal to b. Return True or False	False
!=	`a != b`	Check if a is not equal to b. Return True or False	True
>	`a > b`	Check if a is greater than b. Return True or False	False
>=	`a >= b`	Check if a is greater than or equal to b. Return True or False	False
<	`a < b`	Check if a is less than b. Return True or False	True
<=	`a <= b`	Check if a is less than or equal to b. Return True or False	True

if statements

Syntax	**Example**	**Explanation** (take a = 5, b = 8)
if condition: Statement	`if a <= b:` `  a*=2`	Since a(5) is less than b(8), condition `a <= b` evaluates to `True`. Statement `a*=2` is therefore executed, and the result is `a=10, b=8`

if condition.1:
  Statement.1
else:
  Statement.2
	`if a > b:`
	`  a*=2`
	`else:`
	`  a-=b`
		Since a(5) is less than b(8), condition.1 `a > b` evaluates to `False`. Statement.1 `a*=2` is therefore ignored and instead Statement.2 `a-=b` is executed, and the result is `a=-3, b=8`

if condition.1:
  Statement.1
elif condition.2:
  Statement.2
else:
  Statement.3
	`if a > b:`
	`  a*=2`
	`elif a < b:`
	`  b/=a`
	`else:`
	`  a-=b`
		Since a(5) is less than b(8), condition.1 `a > b` evaluates to `False`. Statement.1 `a*=2` is therefore ignored and condition.2 `a < b` is tried and evaluated to `True`. So, Statement.2 `b/=a` is executed and yields `b=1.6`. Now that Statement.2 has been executed, Statement.3 `a-=b` is ignored. The end result: `a=5, b=1.6`

Source of the text: Lutz, 2013.

Range

The range function creates a range of numbers based on a set of parameters: start, stop and step.

Syntax: `range(start, stop, step)`

Example: `range(3,9,2)` creates a range of values starting from 3, ending at the number preceding 9 in steps of 2 i.e., the values are 3, 5, 7.

Other variants:

`range(7)` is equivalent to `range(0,7,1)` and produces 0, 1, 2, 3, 4, 5, 6

`range(2,7)` is equivalent to `range(2,7,1)` and produces 2, 3, 4, 5, 6

For loop

The for loop iterates a code block until the end of a range.

Syntax: `for <variable> in <range>:`

Example:

```
for x in range(7):
    print(x)
```

This code block iterates until the end of `range(7)` i.e., prints the value of x: 0,1, 2, 3, 4, 5, 6

Source of the text: Lutz, 2013.

While loop

The while loop iterates a code block until an expression is evaluated to False.

Syntax: `while <expression>:`

Example:

```
x = 0
while x < 7:
    print(x)
    x+=2
```

This code block iterates until the expression `x < 7` is False i.e., until x is greater or equal to 7 starting with x=0. So, this code prints the value of x: 0, 2, 4, 6

Important key words

Key words	Example	Description

`continue`
`break`
	`for x in range(10):`
	`  if x == 4:`
	`  continue`
	`  elif x > 6:`
	`  break`
	`  print(x)`
		Iterate the code block with x in range(10) i.e., x runs from 0, 1, 2, …, 9. If x is equal to 4, the `continue` statement skips the remaining codes and continues with the next number in the range i.e. 5. If x is greater than 6, `break` statement forces the code block to exit. The result of print(x): 0, 1, 2, 3, 5, 6

Source of the text: Lutz, 2013.

Iterators

One can also loop through values in lists, sets, tuples and dictionaries.

Example:

```
my_list = [1, 3, 5, 7]
for x in my_list:
    print(x)
```

This code block iterates through `my_list` i.e., prints the values in the list: 1, 3, 5, 7

Comprehension

Comprehensions are container objects (e.g., lists, sets, tuple, dictionaries) that are generated from existing container objects.

Syntax: `[<expression> for n in <existing_list>]`

Example:

```
my_list = [1, 3, 5, 7]
my_new_list = [2*n for n in my_list]
```

← an existing list

← a new list

Note that tuple comprehensions work slightly differently.

Example:

```
my_tuple = (1, 3, 5, 7)
my_new_tuple = tuple(2*n for n in my_tuple)
```

Source of the text: Lutz, 2013.

REVIEW STUDY GOALS

- Use basic assignments and expressions
- Learn how to use various conditional statements
- Implement loops
- Apply iterators and comprehensions

SESSION 3

TRANSFER TASK

TRANSFER TASK

Write a python code for a bingo game: Start off with setting a secret number between 1 and 20. The player has 5 chances to guess what that secret number is. So, at the prompt the player makes a guess. If the guess is lower (higher) than the secret number, the program returns a hint: ‘higher’ (‘lower’) and lets the player make another guess. If the player makes a correct guess at the prompt, the program returns ‘bingo!’ and exits. If the player does not manage a correct guess within the allowed 5 chances, the program returns ‘Oops, game over!’ and tells the player the secret number.

TRANSFER TASK
PRESENTATION OF THE RESULTS

Please present your results.

The results will be discussed in plenary.

1. What is the value of the variable c after the following statement: a=b=c=d=20?

2. What is the code that will generate a range of numbers from 11 to 99 with increment of 2?

3. Why will the following code not run?

```
my_num = 0
int(input())
if my_num < 0
  print('negative')
elif my_num > 0
  print('positive')
else
  print('zero')
```

LIST OF SOURCES

Lutz, M. (2013). Introducing Python Object Types. *Learning Python* (5th ed.). O'Reilly.