Chapter 7. Selection

In computer science, *selection* refers to the ability of a program to make decisions and follow different paths of execution depending on certain conditions. Python achieves selection through the use of **boolean values**, **boolean expressions**, **logical operators**, and various forms of **conditional statements**.

7.1 Boolean Values and Boolean Expressions

- A **boolean value** has only two possibilities: True or False. These are the fundamental values of logic in programming and are of type bool.
- A **boolean expression** is any expression that evaluates to a boolean value. Examples include comparisons like 5 > 3 (which evaluates to True) or x == 10.
- **Comparison operators** are used to form boolean expressions. They include:
    - == (equal to)
    - != (not equal to)
    - > (greater than)
    - < (less than)
    - >= (greater than or equal to)
    - <= (less than or equal to)

Boolean expressions allow programs to ask questions and make decisions based on the answers.

7.2 Logical Operators

- A **logical operator** combines boolean expressions to form more complex conditions. Python supports:
    - and – True if both expressions are true.
    - or – True if at least one expression is true.

- not – Reverses the value of a boolean expression (True becomes False, False becomes True).

7.2.1 Logical Opposites

Logical opposites help simplify conditions. For example:

- not (x > 10) is equivalent to x <= 10.
- not (x == y) is equivalent to x != y.

7.3 Order of Operators

Just like arithmetic operators, logical and comparison operators have rules of precedence.

- Comparisons (<, >, ==, etc.) are evaluated before logical operators.
- Among logical operators, not has the highest precedence, followed by and, and finally or.
- Parentheses () can be used to group expressions.

7.4 Conditional Execution: if and else

A **conditional statement** controls the flow of execution depending on a condition.

- In Python, this is written using if and else.
- The **condition** is the boolean expression being tested.
- The **body** is the block of statements indented under the conditional header.
- Each possible outcome is called a **branch**.

Example:

```
if x > 0:
   print("Positive")   # body for True branch
else:
   print("Non-positive")  # body for False branch
```

7.6 Nested Conditionals

Nesting means placing one program structure inside another. A **nested conditional** is when an if statement is inside another if or else branch.

Example:

```
if x > 0:
   if x % 2 == 0:   # modulus operator checks even/odd
      print("Positive and even")
   else:
      print("Positive and odd")
```

Here, the modulus operator % is used to compute the remainder, allowing the program to check if a number is even or odd.

7.7 Chained Conditionals

A **chained conditional** allows more than two possible flows of execution using if ... elif ... else.

Example:

```
if x > 0:
   print("Positive")
elif x == 0:
   print("Zero")
else:
   print("Negative")
```

7.8 Boolean Functions

A **boolean function** is a function that returns a boolean value (True or False). They are useful for making code reusable and readable.

Example:

```
def is_even(n):
   return n % 2 == 0
```

Calling is_even(4) returns True, while is_even(5) returns False.

7.8.1 More Unit Testing

Since boolean functions return only two possible outcomes, they are easy to test. **Unit testing** ensures that functions behave correctly for various input cases. For example:

```
print(is_even(2))  # Expected True
print(is_even(3))  # Expected False
```

Testing boolean functions helps verify program correctness and reduce errors.