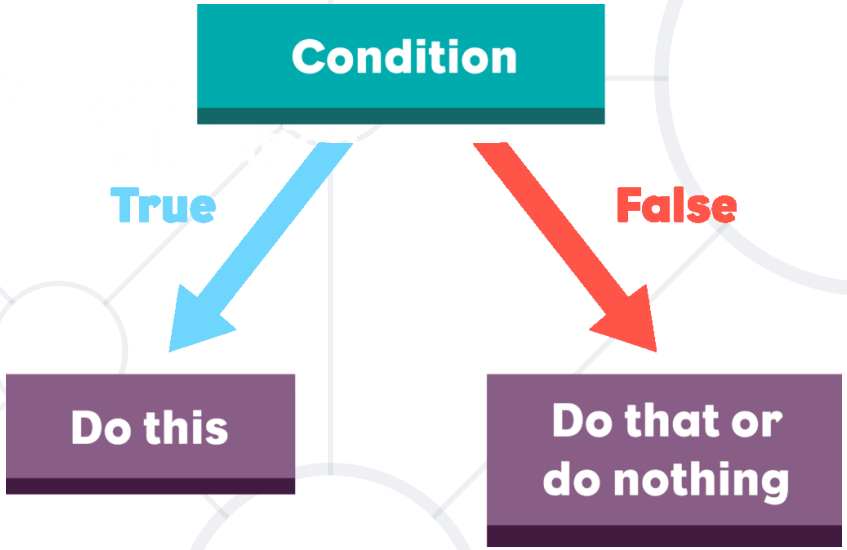
Conditional Statements



SoftUni

Training team

Software University

SoftUni

Software University

https://softuni.bg

Contents

Comparison Operators

Comparison Operators

Operator	Notation	Used for
Equality	==	numbers, dates, other comparable types
Difference	!=	
Bigger	>	
Bigger or equal	>=	
Less	<	
Less or equal	<=	

Comparing Values

- In programming, we can compare values
- The result of logical expressions is value: `True` or `False`

```
a = 5
b = 10
print(a < b)          # True
print(a > 0)          # True
print(a > 100)        # False
print(a < a)          # False
print(a <= 5)         # True
print(b == 2 * a)     # True
```

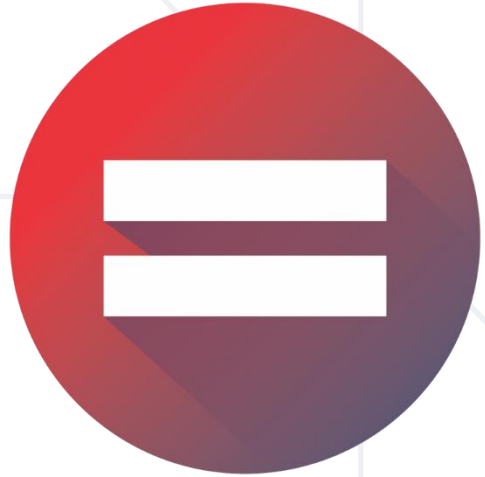
Comparing Values

- Comparing text using equality operator (==)

```
a = 'Example'
b = a
print(a == b)       # True
```

```
a = input()
b = input()
print(a == b)       # True
```

Entering the same value



Boolean Variable

- Has only the following two values: **True** or **False**

```
is_valid = True
```

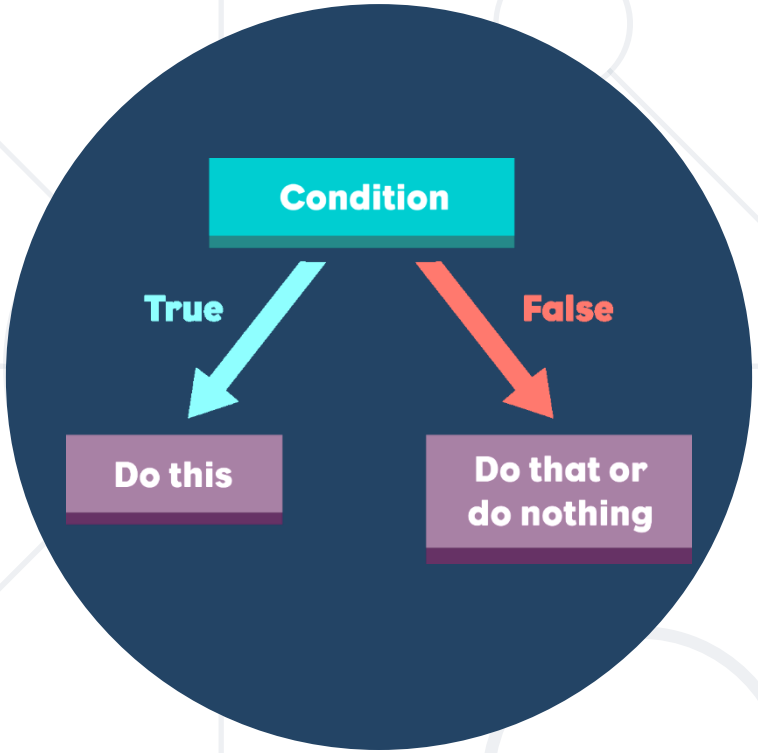
- Can also be created with a condition that evaluates to true or false

```
is_positive = a > 0
```

Boolean Variable - Example

```
a = 5
is_positive = a > 0
print(is_positive)  # True
```

```
a = -5
is_positive = a > 0
print(is_positive)  # False
```



Conditional Statements

Simple Conditional Statement

- We often check conditions and perform actions based on a result

```
if ...:
    # Code to execute
```

Condition (boolean expression)

Code to execute if the condition is true

- The result is `True` or `False`

Conditional Statement: `if-else`

- In case of **false condition**, we can perform other actions – through the `else` construction

```
if ...:
    # code to be executed
else:
    # code to be executed
```

Code to be executed when the condition is false

Block of Code

- **Tabs** form **block of code** (a group of commands)
  - The line that **matches** the condition is executed

```
color = 'red'
if color == 'red':
    print('Red')
else:
    print('Yellow')
    print('bye')
```

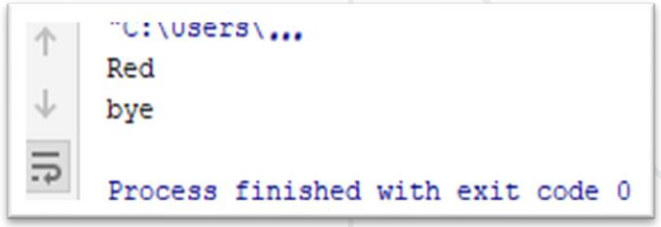
Block of Code

- Without tabs, the **last** line will also be executed

```
color = 'red'
if color == 'red':
    print('Red')
else:
    print('Yellow')
print('bye')
```

The lines that match the condition are executed

It is always executed – not part of the if / else construction



Rounding and Formatting

Working with Numbers

- In programming, we can round real numbers
  - Rounding up to the next (bigger) integer:

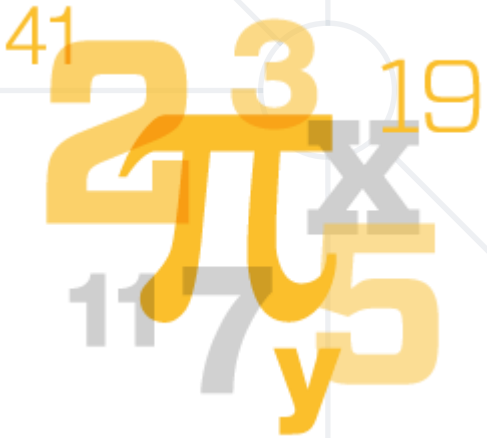
```
up = math.ceil(23.45)          # 24
```

  - Rounding down to the previous (smaller) integer:

```
down = math.floor(45.67)    # 45
```

- Finding the absolute value

```
example1 = abs(-50)          # 50
example2 = abs(50)           # 50
```



Rounding and Formatting

- Rounding to **2** decimal places:

```
rounded = round(45.67852, 2)  # 45.68
```

- Formatting to **2** decimal places:

```
print(f"{123.456:.2f}")          # 123.46
```

Number of characters after the decimal point

- Difference between formatting and rounding

```
print(round(45.60000, 4))        # 45.6
print(f"{45.60000:.4f}")         # 45.6000
```

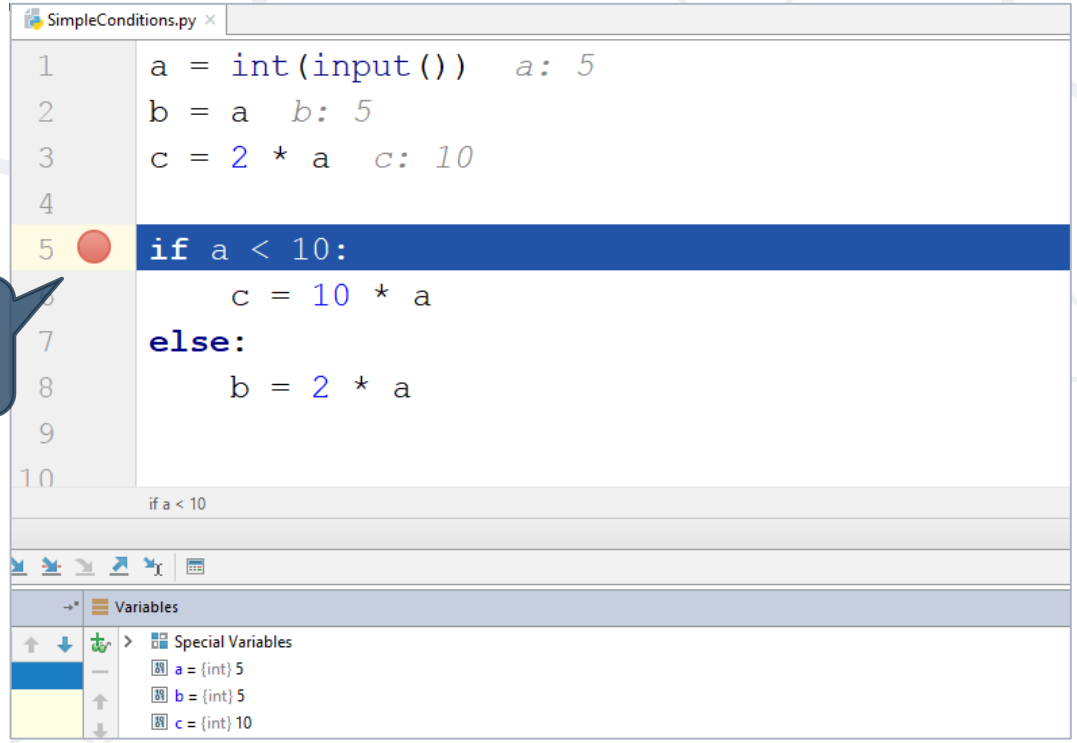
Debugging

Simple Operations with Debugger

Debugging

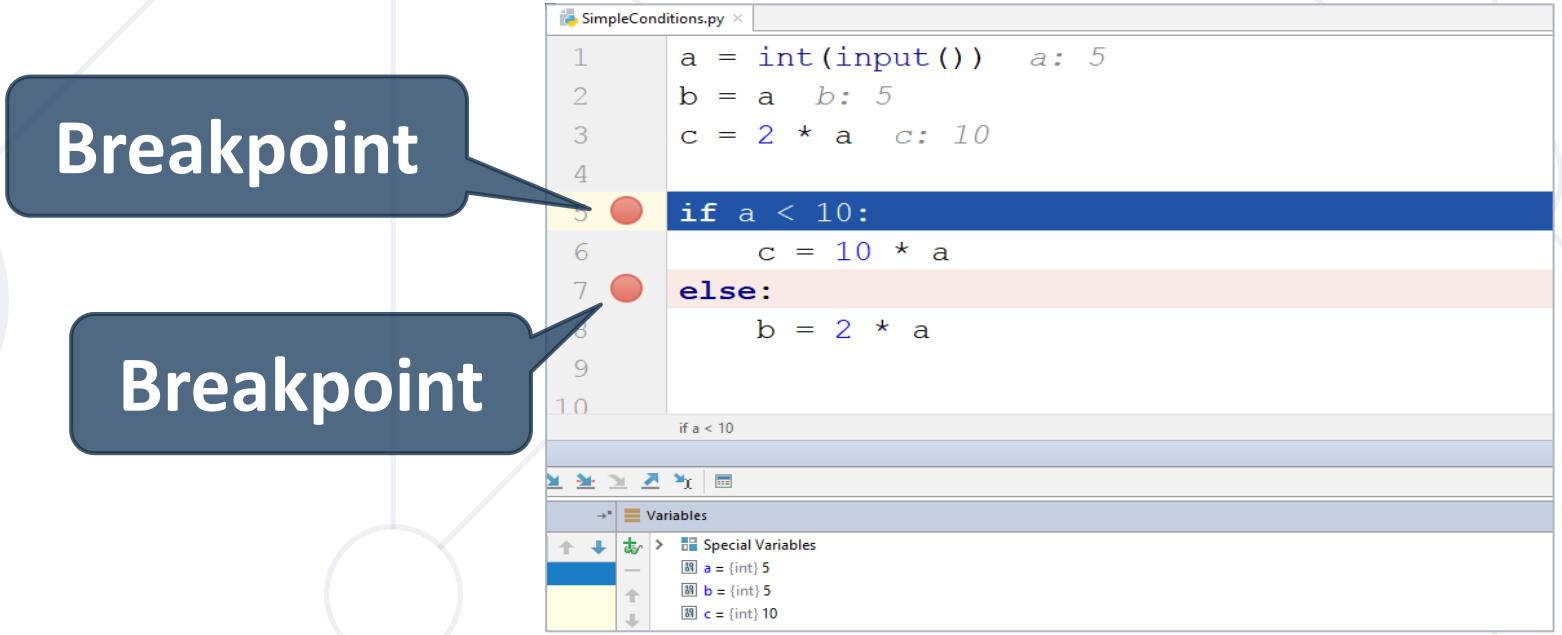
- Process of tracing the execution of the program
  - Allows us to detect errors (bugs)

Breakpoint



Debugging in PyCharm

- Pressing [Shift + F9] will start the program in debug mode
- We can move to the next step with [F8]
- We can create [Ctrl + F8] breakpoints
  - We can directly reach them using [F9]



Complex Conditional Statements

Series of Checks

- The construction **`if/else - if/else…`** is series of checks

```
if ...:
  # code
elif ...:
  # code
elif ...:
  # code
else:
  # code
```

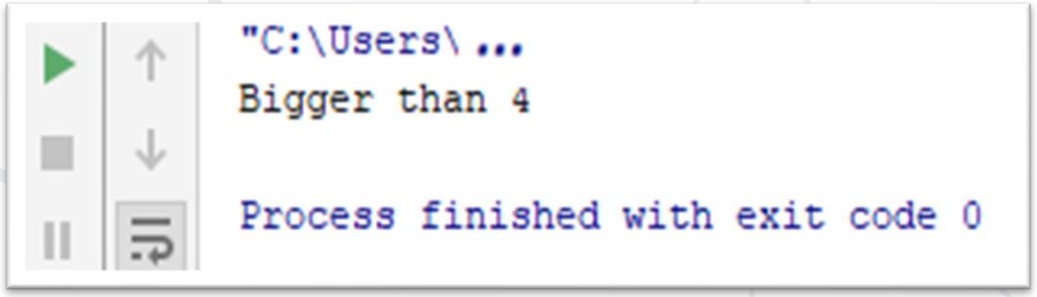
- When the condition is true, the evaluation of the next conditions is **not continued**

Series of Checks - Example

- The program checks the first condition, determines that it is true and ends

```
a = 7
if a > 4:
    print('Bigger than 4')
elif a > 5:
    print('Bigger than 5')
else:
    print('Equal to 7')
```

Only 'Bigger than 4' is displayed



Variable Lifespan

Scope of Usage

Variable Lifespan

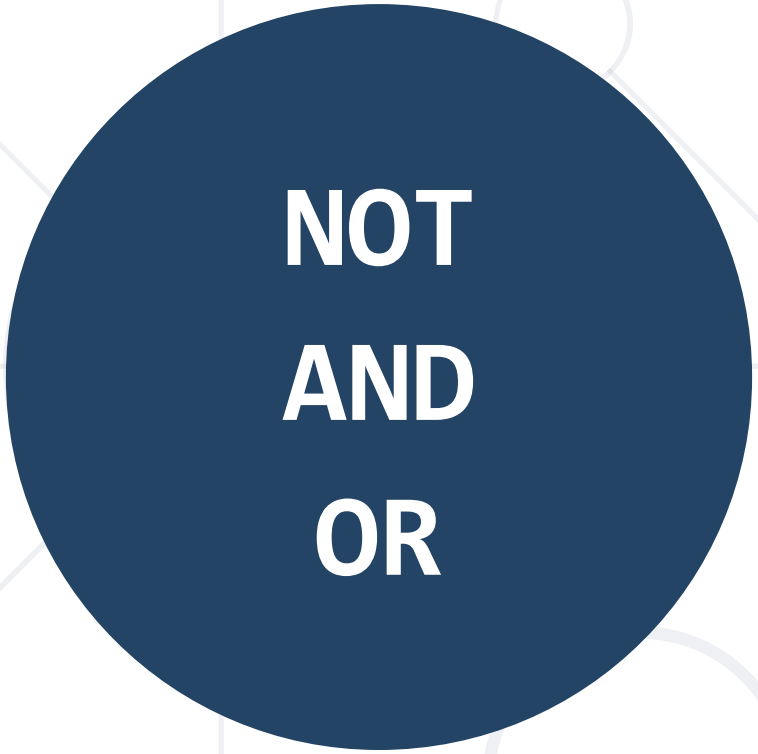
- Example: The variable **`salary`** will exist **only** if it is initialized somewhere in the program

```
current_day = "Monday"
if current_day == "Monday":
    salary = 1000
print(salary)  # 1000
```

Variable Lifespan

- Example: The variable **`salary`** will **not** exist if it is not initialized somewhere in the program

```
current_day = "Tuesday"
if current_day == "Monday":
    salary = 1000
print(salary)  # Error
```



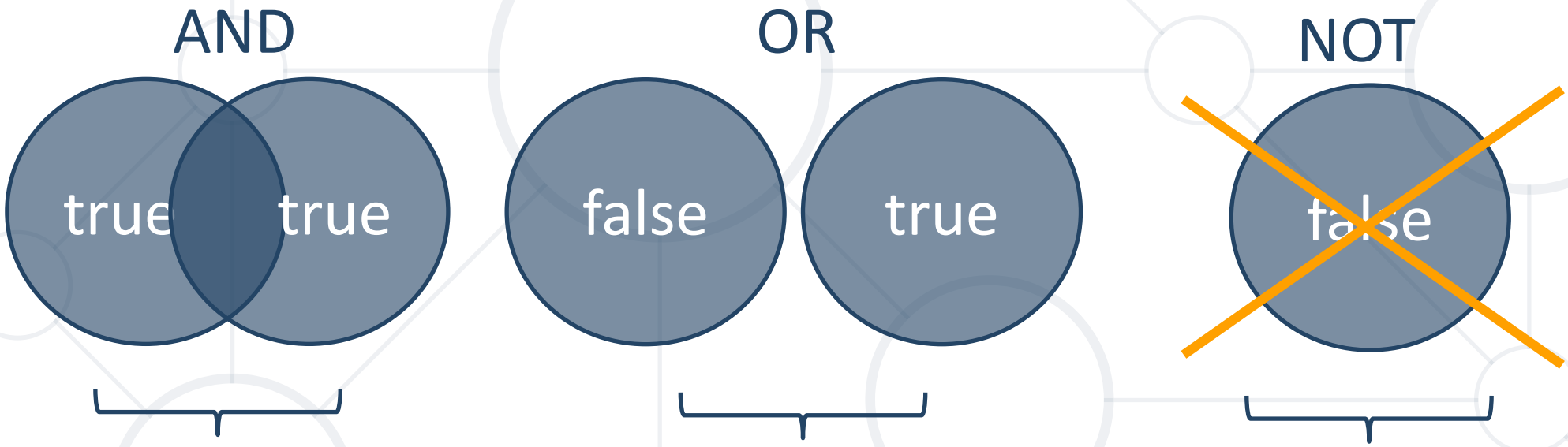
Logical Operators

Checking Complex Conditions

Logical Operators

- Logical checks are based on **logical conditions**
- The **logical operators** in Python are:
  - Logical **AND**
  - Logical **OR**
  - Logical **NOT**
- **Brackets** **( )** change the order

Logical Operators: Explanation



Both conditions must be true

One condition must be true

Logical negation

Logical AND

- Returns the Boolean value `true` if all of the operands are `true` and `false` otherwise
- Example: check **number** is in the following **range [100; 200]**

```
if number >= 100 and number <= 200:
    print("Number is in range")
```

Logical OR

- The result of the expression is `true` if one of the operands is `true`, otherwise the result is `false`

```
product == "tea" or product == "water"
```

- Problem: **check for food or drink**
  - Read single line and print "`drink`", "`food`" or "`unknown`"
  - Foods: `curry`, `noodles`, `sushi`, `spaghetti`
  - Drinks: `tea`, `water`, `coffee`
  - Everything else is `unknown`

Logical NOT

- Logical negation returns `true` when the operand is `false`, and `false` when the operand is `true`
- Example: **check for valid number**
  - A number is **valid** if is in the range [**100…200**] or is equal to **0**

```
isValid = (num >= 100 and num <= 200) or num == 0:
if not isValid:
    print("invalid")
```

Summary

- Comparison operators
- Condition statements
- Rounding and Formatting
- Series of checks
- Debugging
- Variable Lifespan
- Logical Operators

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity