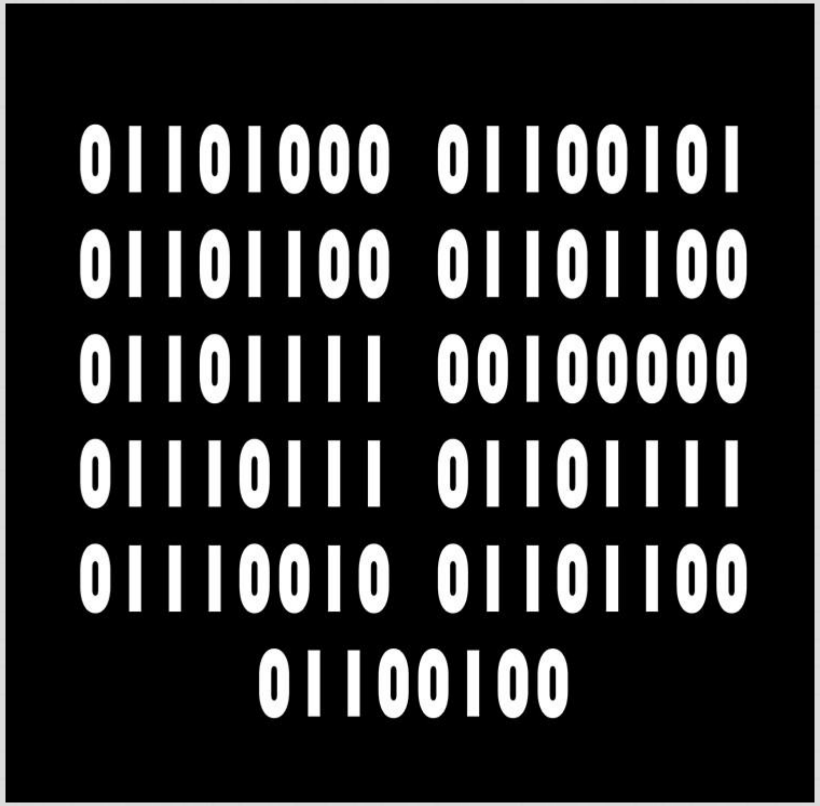
CYDEO

Day01 Presentation Slisde

Contents

Why do we need Programming Languages?

- To Communicate with computers
- Computer ONLY understands machine language and machine language is in binary code

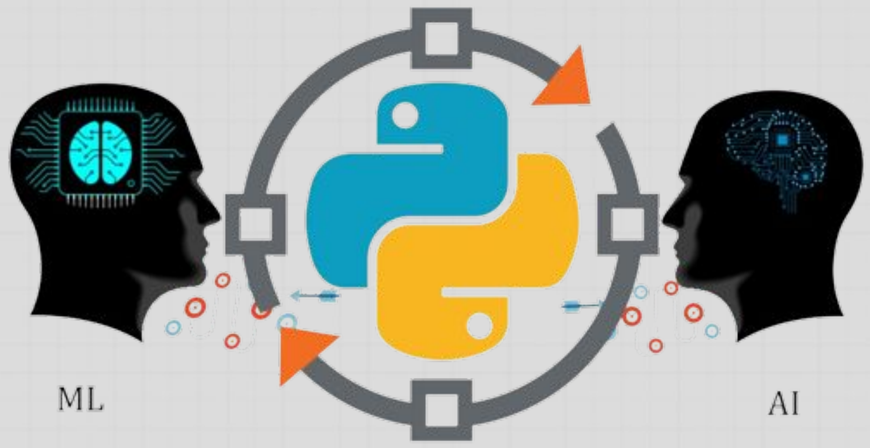


```
01101000 01100101
01101100 01101100
01101111 00100000
01110111 01101111
01110010 01101100
    01100100
```

Why Learn Python?

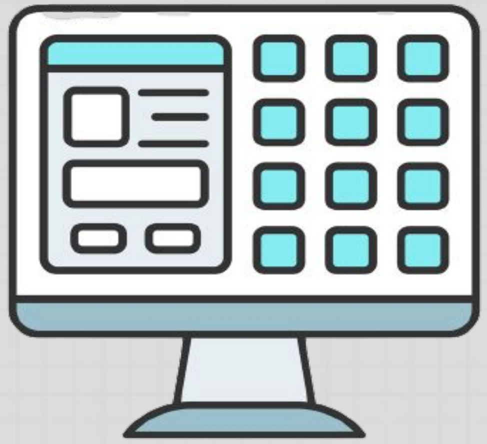
- The most popular programming language
- The easiest programming language
- The demand is high
- Extremely versatile

Where Is Python Mostly Used At?



WEB DEVELOPMENT

GAME DEVELOPMENT

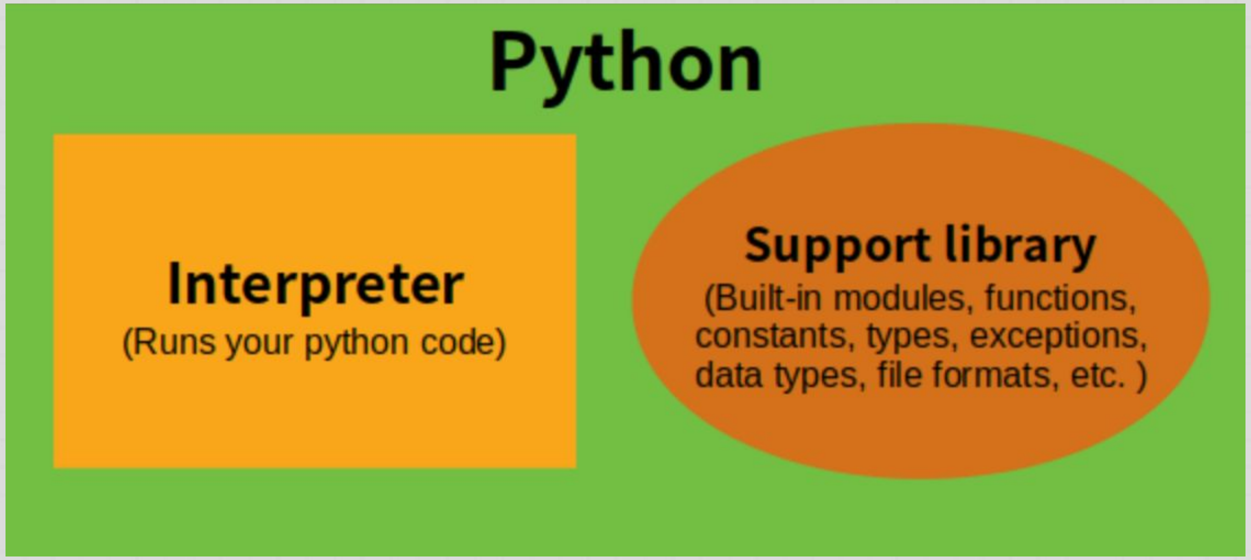

GUI

Components in Python

- Python software includes the

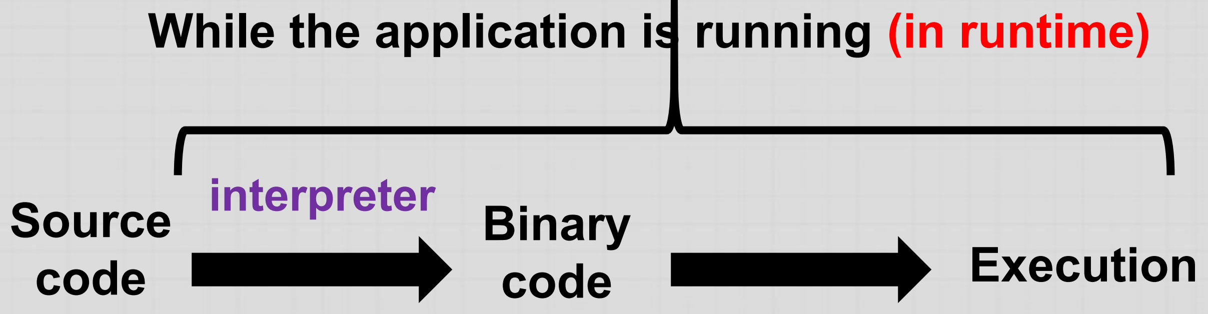
  following components:

  - Interpreter
  - Support Library

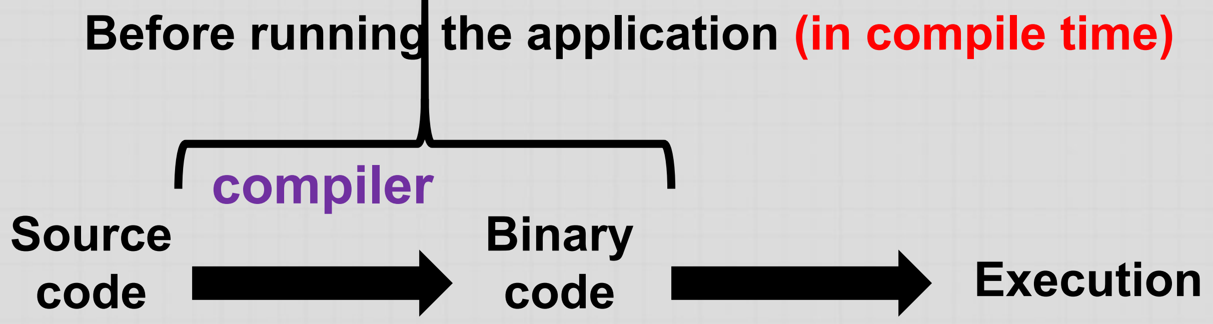

Python
Interpreter
(Runs your python code)
Support library
(Built-in modules, functions, constants, types, exceptions, data types, file formats, etc.)

Interpreter vs Compiler

An interpreter is a program reading the source code line by line, translates them to machine code in runtime.



A compiler is a tool that translates the entire source code of a program into a form that can be executed by the computer's hardware



Different IDEs for Python

- An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development

PC

spyder

Visual Studio Code

eclipse®

WING PYTHON IDE

THE INTELLIGENT DEVELOPMENT ENVIRONMENT FOR PYTHON

Why PyCharm IDE?

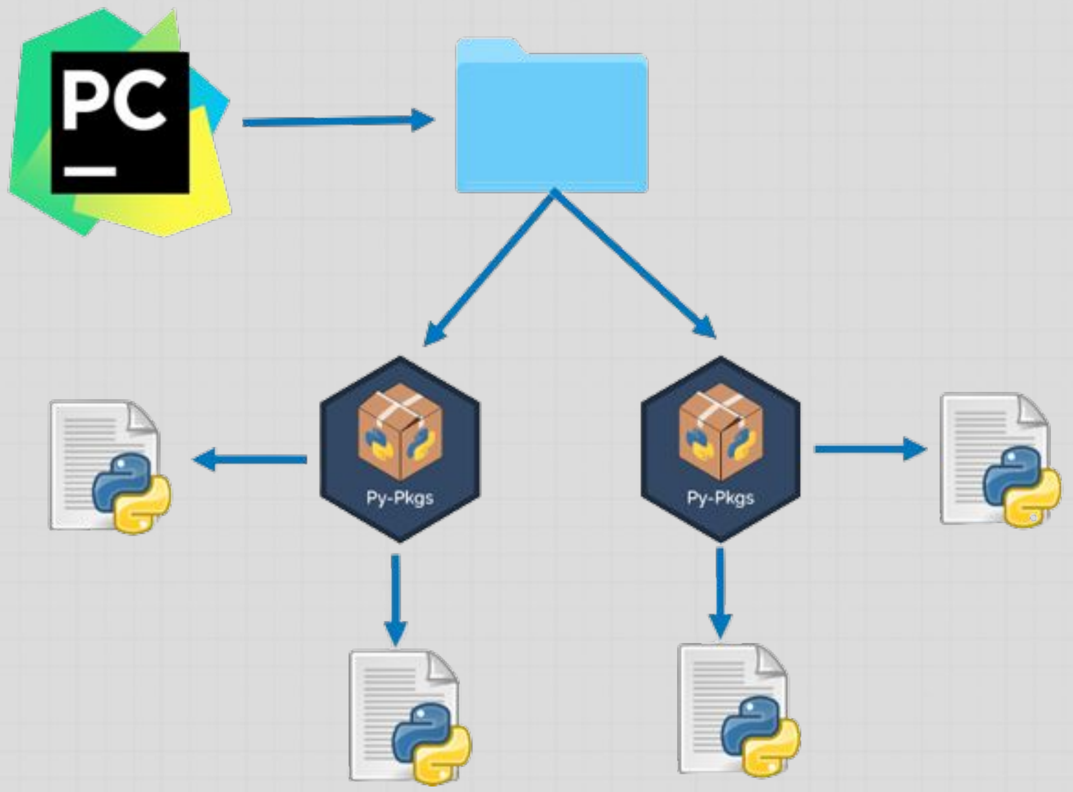
- The most user-friendly IDE
- easy and interactive user interface
- Support for a lot of developer tools
- Improves Productivity with smart code completion
- Different plugins for even more customization

Steps of Creating Python Project

- Step 1: Open your PyCharm application
- Step2: Go to the New Project window
- Step 3: Choose the project location
- Step 4: Set the Virtualenv environment
- Step 5: Set the Base interpreter
- Step 6: Click the Create button

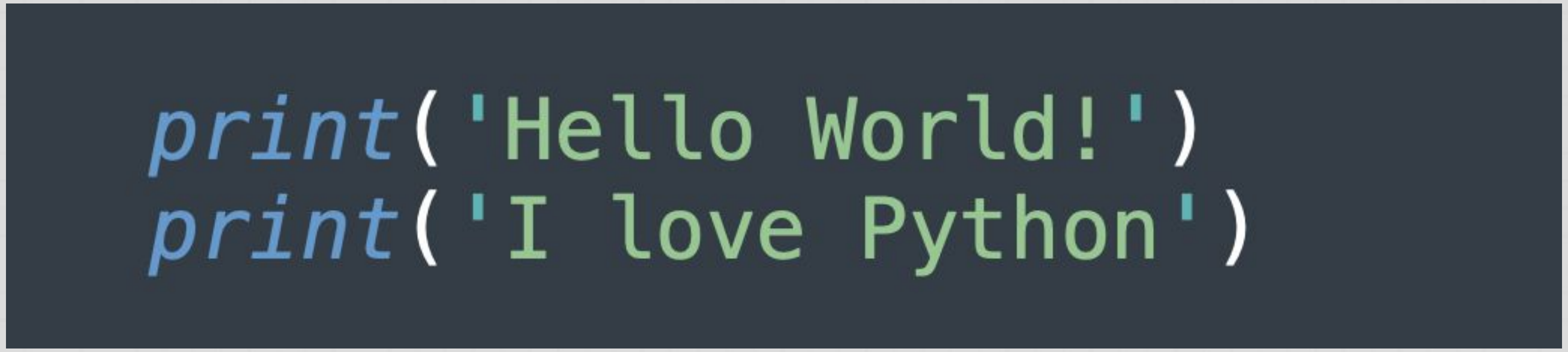
First Python Programming

- Project: The root directory of all our python files and packages
- Py: python file is where we write our source codes
- Py-Pkgs: Collection of modules. Modules that are related to each other



The Print() method

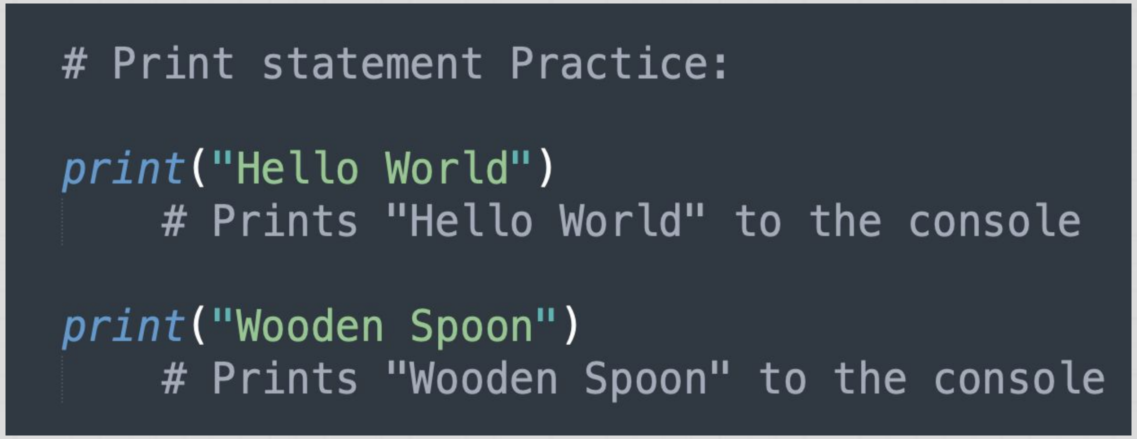
- Used for printing data on the console
- Appends a newline at the end of the data output



```python
print('Hello World!')
print('I love Python')
```

Comments

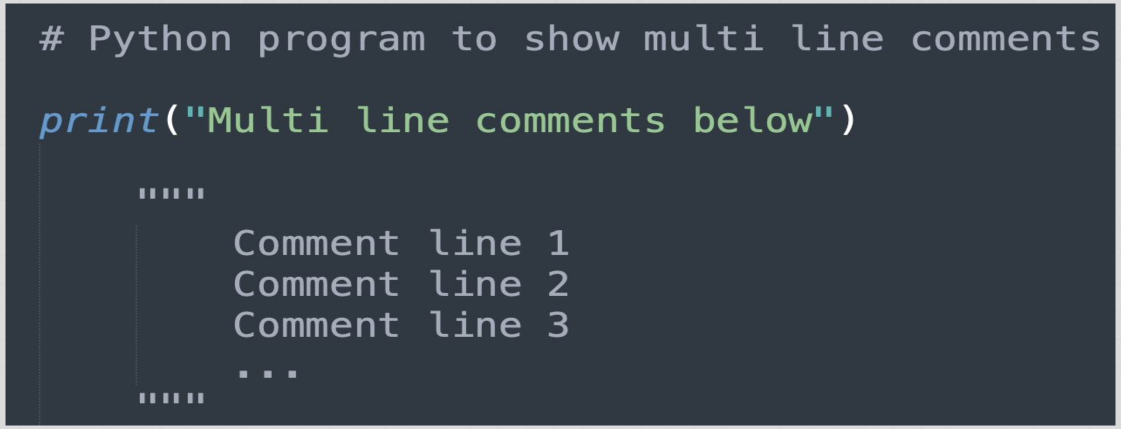
Single-line comments are often used for short descriptions of what the code is doing.



```python
# Print statement Practice:

print("Hello World")
    # Prints "Hello World" to the console

print("Wooden Spoon")
    # Prints "Wooden Spoon" to the console
```

Anything that follows the octothorpe character # on that line will not be processed

Multi-line comments are often used for descriptions of how the script works, or to prevent a section of the script from running when testing it.



```python
# Python program to show multi line comments
print("Multi line comments below")
    """
        Comment line 1
        Comment line 2
        Comment line 3
        ...
    """
```

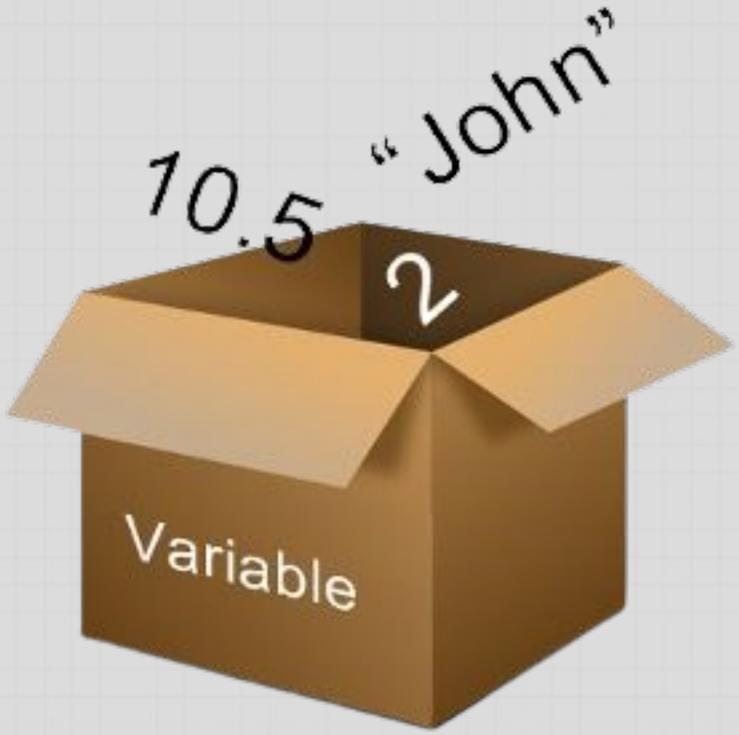
Starting with the triple quote “”” characters and ending with the triple quote “”” characters

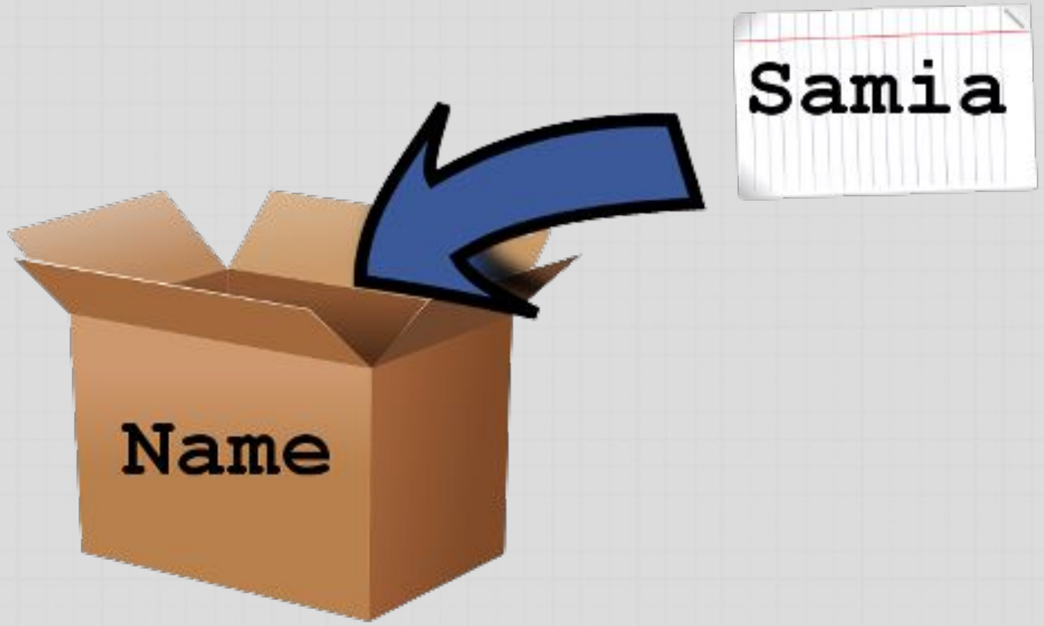
Common Escape Sequences

Escape Sequence	Name	Description
\n	Newline	Advances the cursor to the next line for subsequent printing
\t	Horizontal Tab	Causes the cursor to skip over to the next tab stop
\\	Backslash	Causes a backslash to be printed
\"	Double quote	Causes a double quotation mark to be printed
\’	Single quote	Causes a single quotation mark to be printed

What Is A Variable?

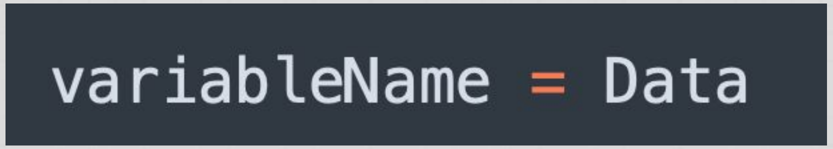
- A variable is a container for storing a data value



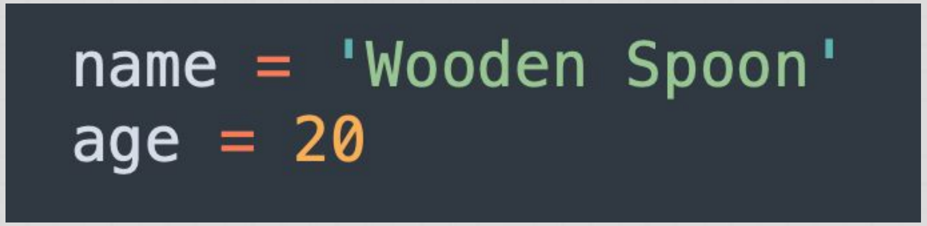


Variable

- Improves the reusability of the data
- Variables must be declared before use
- The Value stored in a variable can be changed during the program execution

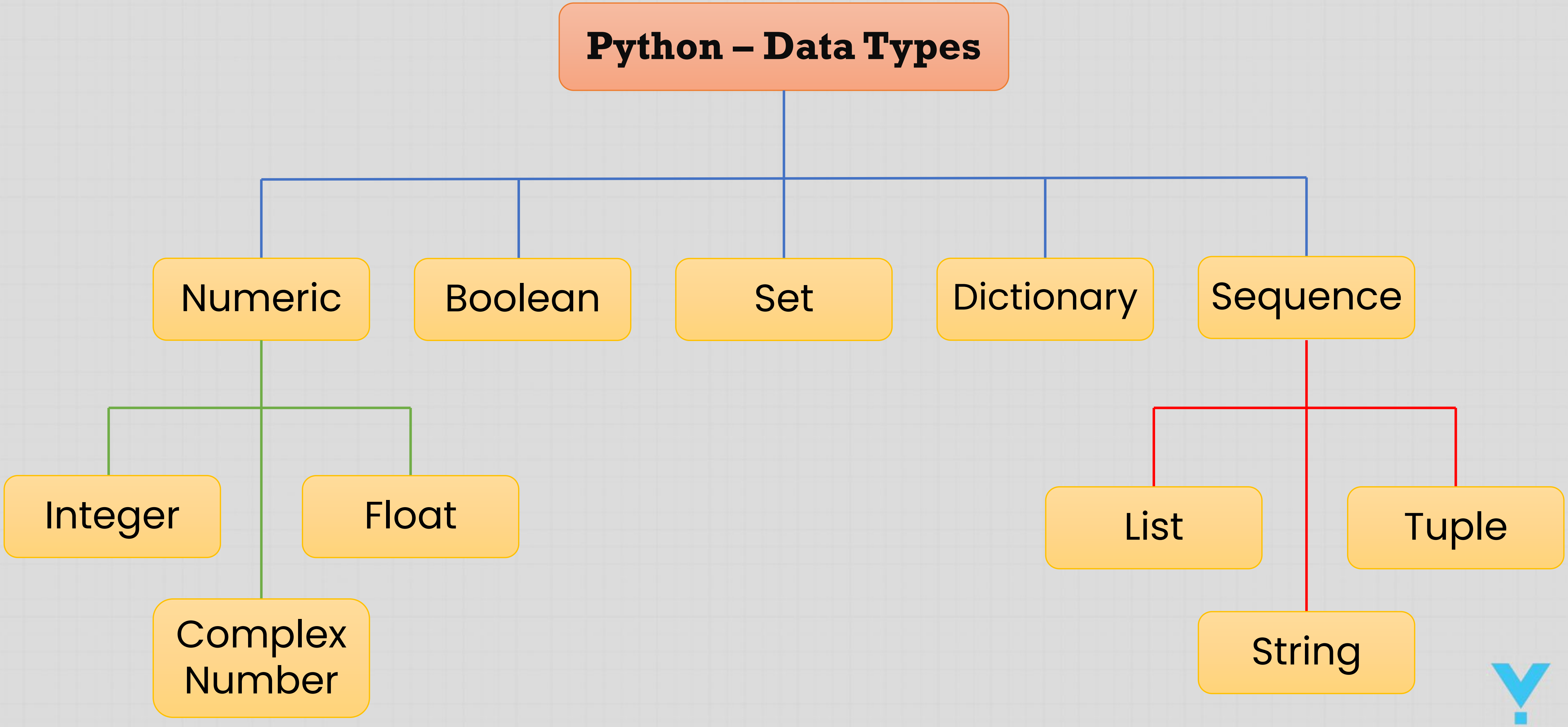


```
variableName = Data
```

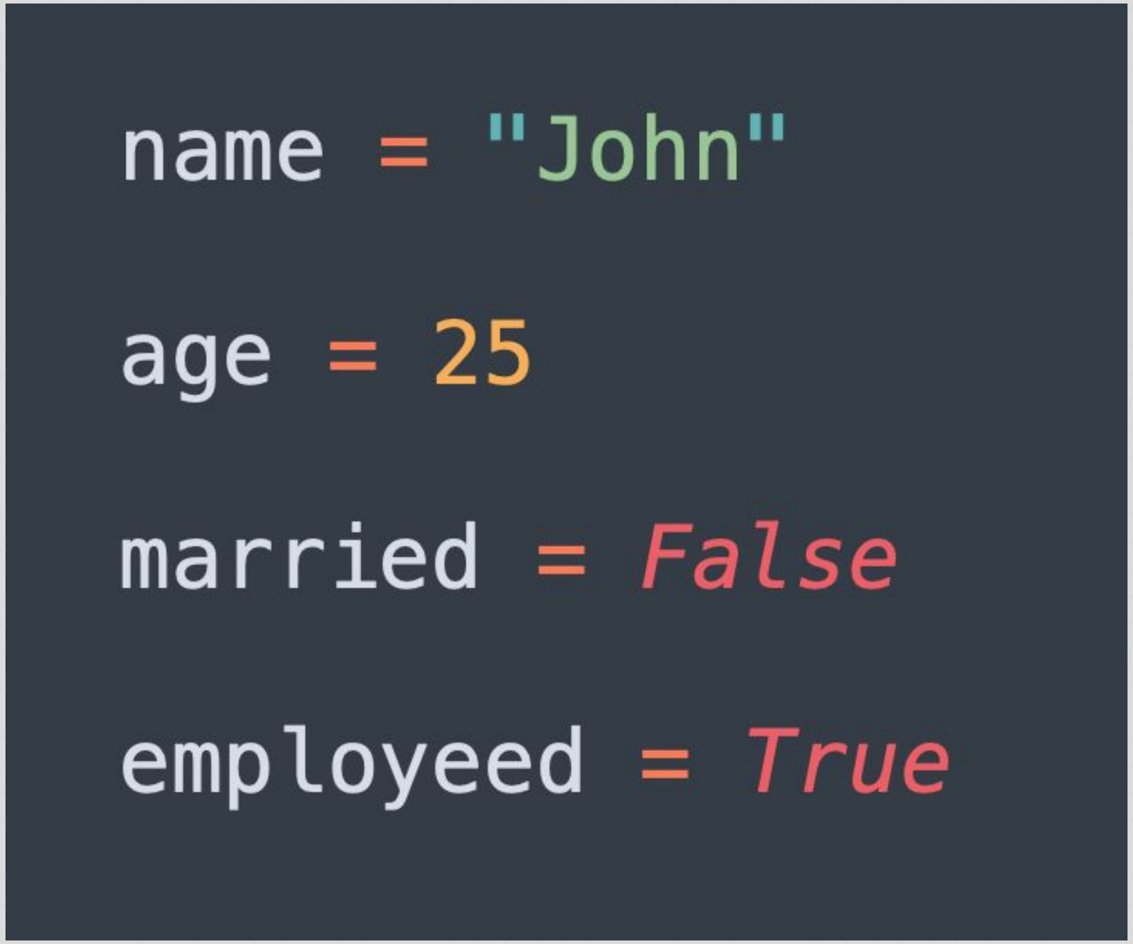


```
name = 'Wooden Spoon'
age = 20
```

Data Types In Python



Variable Declaration Example

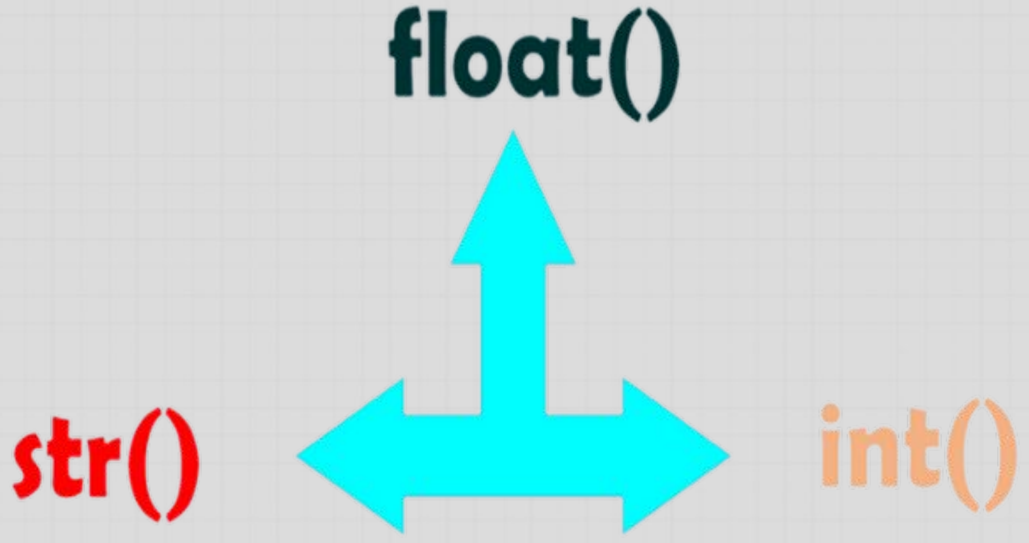


```
name = "John"

age = 25

married = False

employeed = True
```

Type Casting

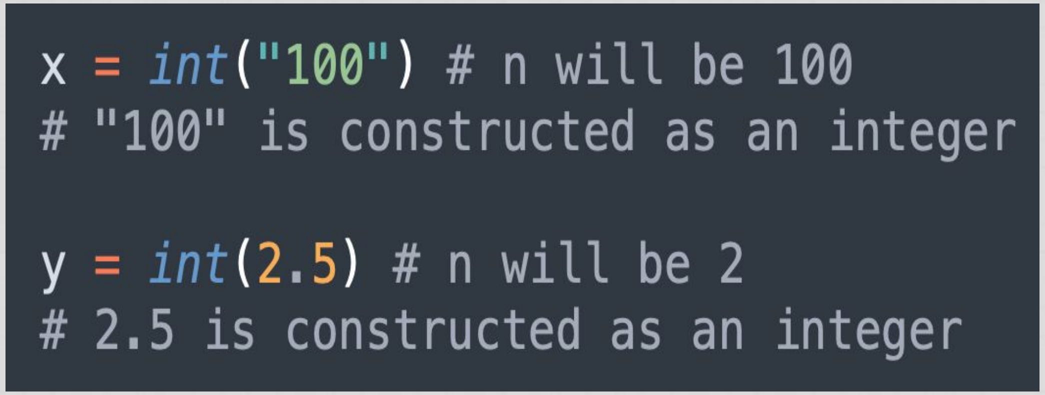
- Allows us to convert one type of value to another type

- Casting is done by using constructor functions:
  - int()
  - float()
  - str()

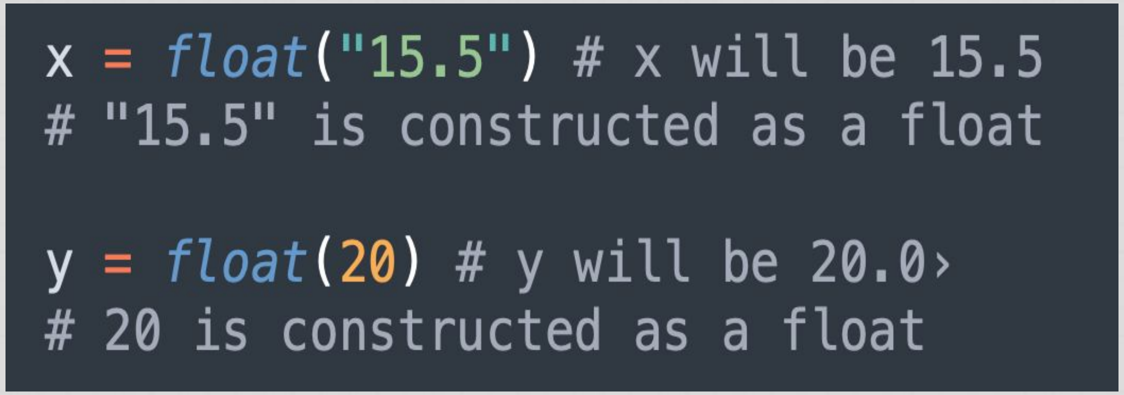


Constructor Functions

- int(): constructs an integer number from a literal (int, float, or string literals)
- float(): constructs a float number from a literal (int, or float, or string literals)
- str(): constructs a string from a literal (int, float, or string literals)



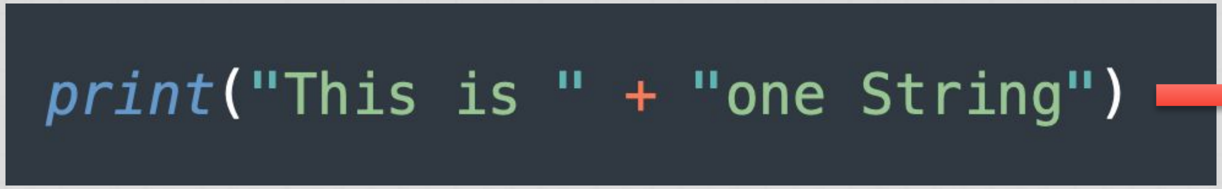
```
x = int("100") # n will be 100
# "100" is constructed as an integer

y = int(2.5) # n will be 2
# 2.5 is constructed as an integer
```



```
x = float("15.5") # x will be 15.5
# "15.5" is constructed as a float

y = float(20) # y will be 20.0
# 20 is constructed as a float
```

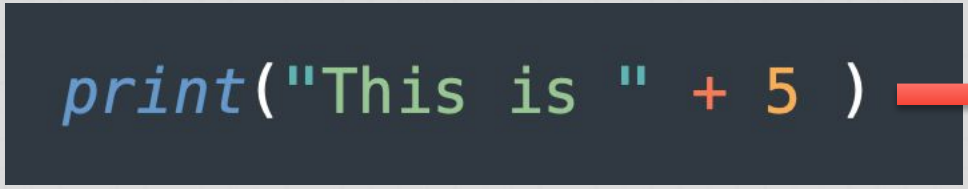
Concatenation with + operator

- The action of linking two strings together
- The two values on both right and left side of the + operator must be strings



```
print("This is " + "one String")
```

This is one string



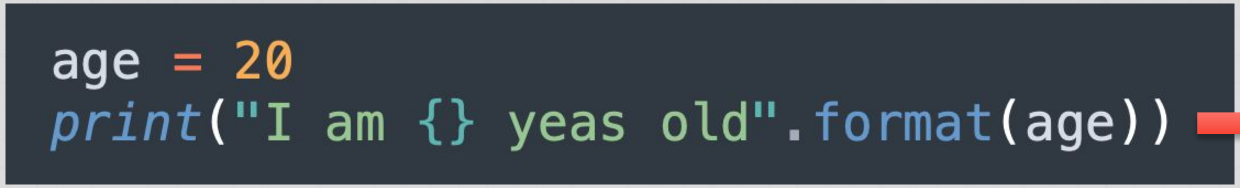
```
print("This is " + 5 )
```

Error

Concatenation with {} operator

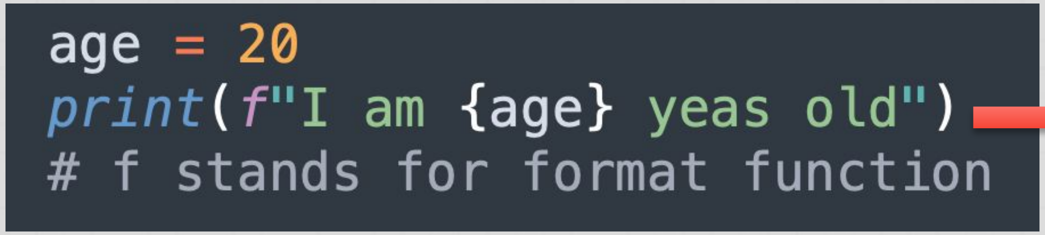
- The action of linking string and other types together

- The format() need to be called in order to pass different types into a string, it can easily be done by adding the character f before the opening double quote of the string



```
age = 20
print("I am {} yeas old".format(age))
```

I am 20 yeas old



```
age = 20
print(f"I am {age} yeas old")
# f stands for format function
```

I am 20 yeas old

Arithmetic Operators

NAME	OPERATOR	PURPOSE & NOTES	EXAMPLE	RESULT
ADDITION	+	Adds one value to another	10+5	15
SUBTRACTION	-	Subtracts one value from another	10-5	5
DIVISION	/	Divides two values	10/5	2
MULTIPLICATION	*	Multiplies two values	10*5	50
MODULUS	%	Divides two values and returns the remainder	10%3	1

Shorthand Operators

NAME	SHORTHAND OPERATOR	MEANING
Assignment	x = y	x = y
Addition Assignment	x += y	x = x + y
Subtraction Assignment	x -= y	x = x – y
Multiplication Assignment	x *= y	x = x * y
Division Assignment	x /= y	x = x / y
Remainder Assignment	x %= y	x = x % y

Relational Operators

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

Logical Operators

OPERATOR	DESCRIPTION
and	Logical AND
or	Logical OR
not	Logical NOT

Membership Operators

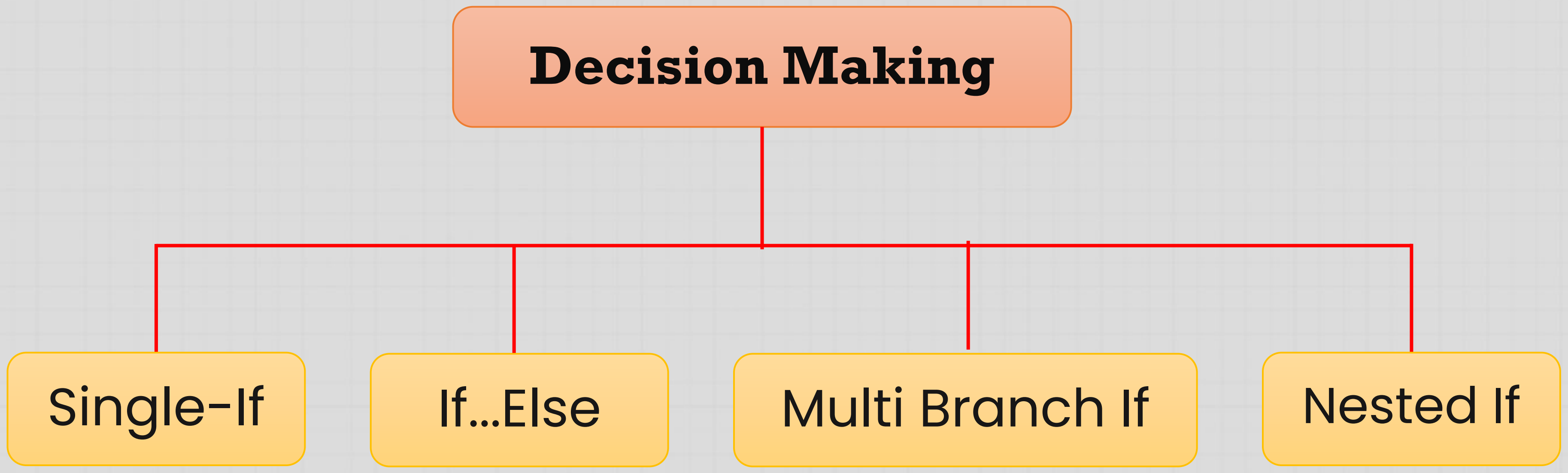
OPERATOR	DESCRIPTION
in	Returns true if the specified value is presented in the object
not in	Returns true if the specified value is not presented in the object

Identity Operators

OPERATOR	DESCRIPTION
is	Returns true if both operands are the same object
is not	Returns true if both operands are not the same object

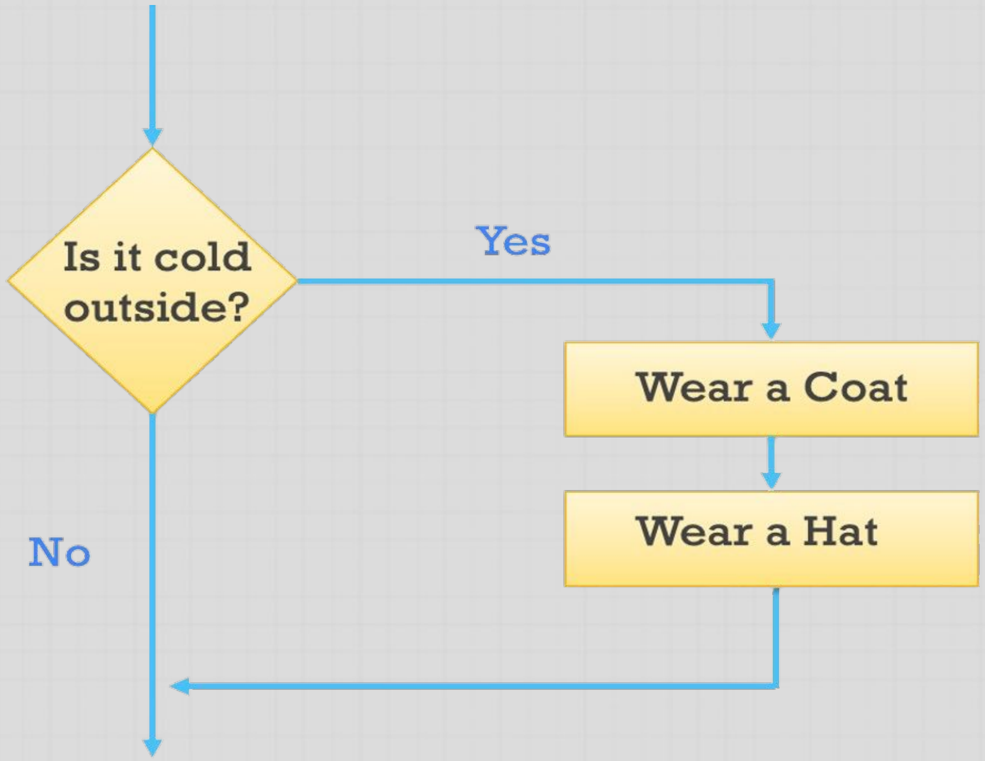
If Statements

- Used for making decisions based on specified criteria

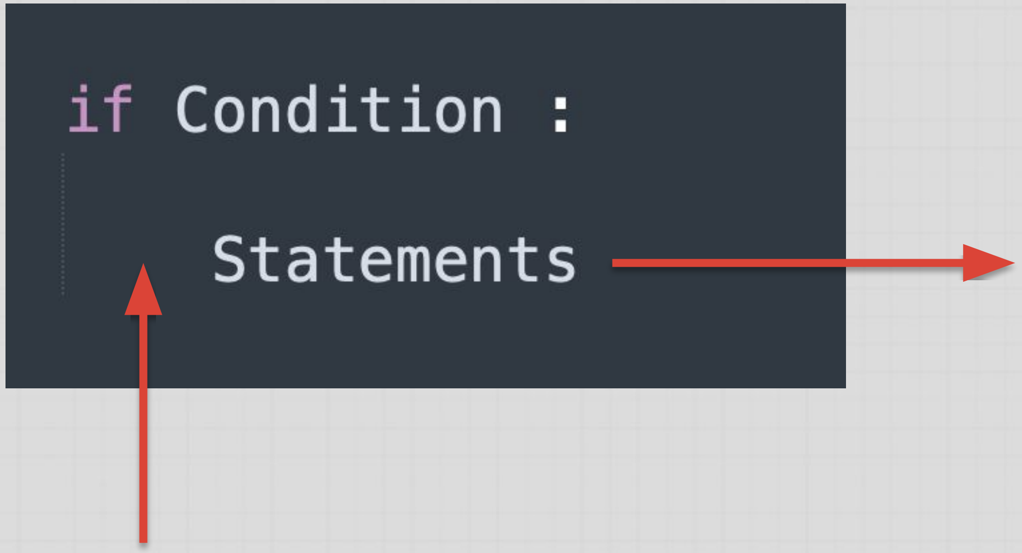


Single If

- The if statement evaluates a condition
- If the condition evaluates to true, any statements in the subsequent code block are executed



Single If – Syntax



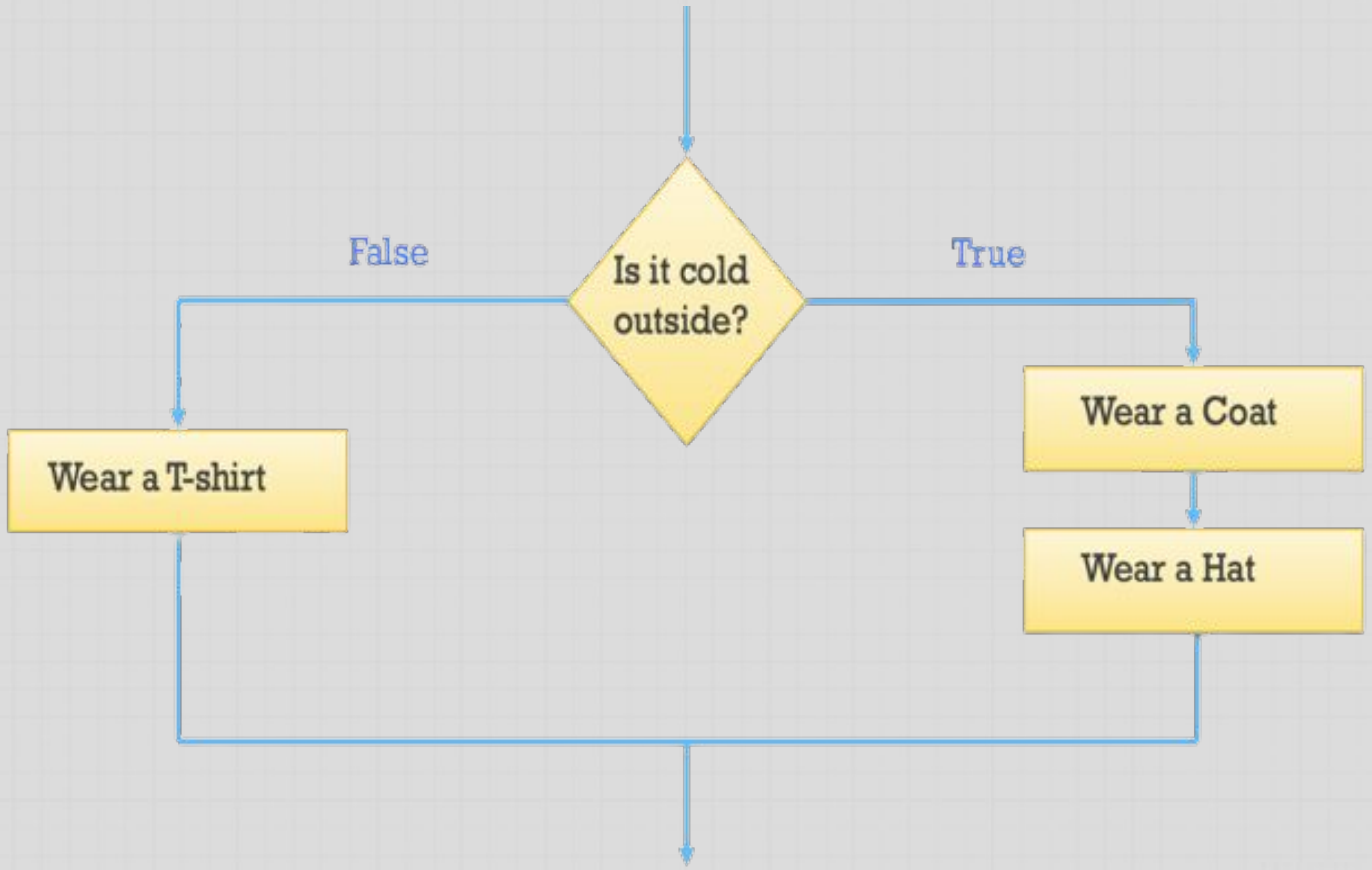
```
if Condition :
    Statements
```

This code fragment is within the scope of the if statement

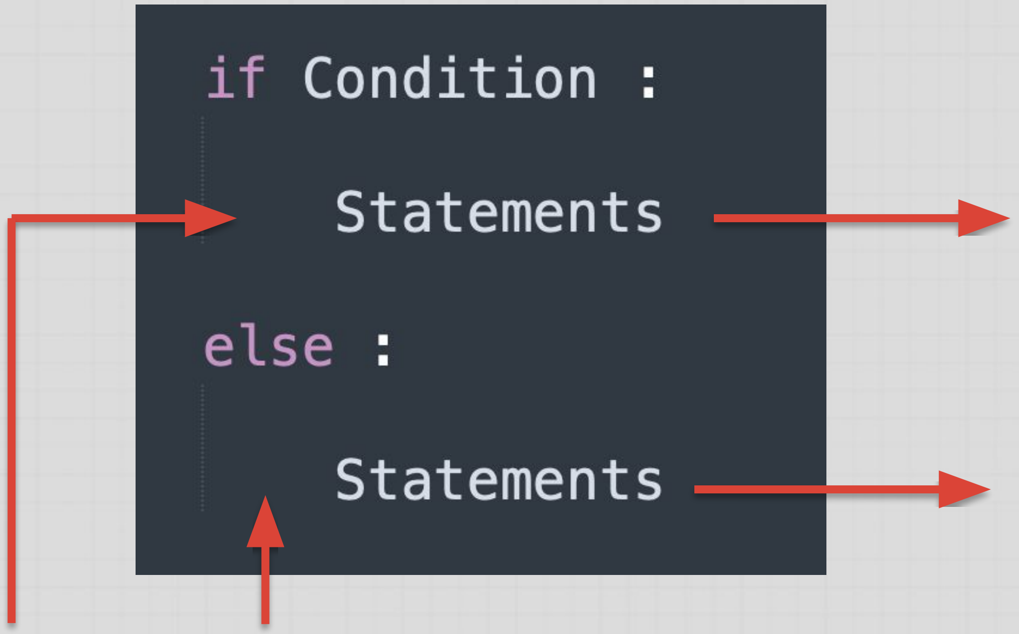
To define scope of the if statement, indentation (whitespaces at the beginning of line) is needed

If...Else

- The if...else statement checks a condition
- If it resolves to true, the first code block is executed
- If the condition resolves to false, the second code block is run instead



If…Else – Syntax



```
if Condition :
    Statements
else :
    Statements
```

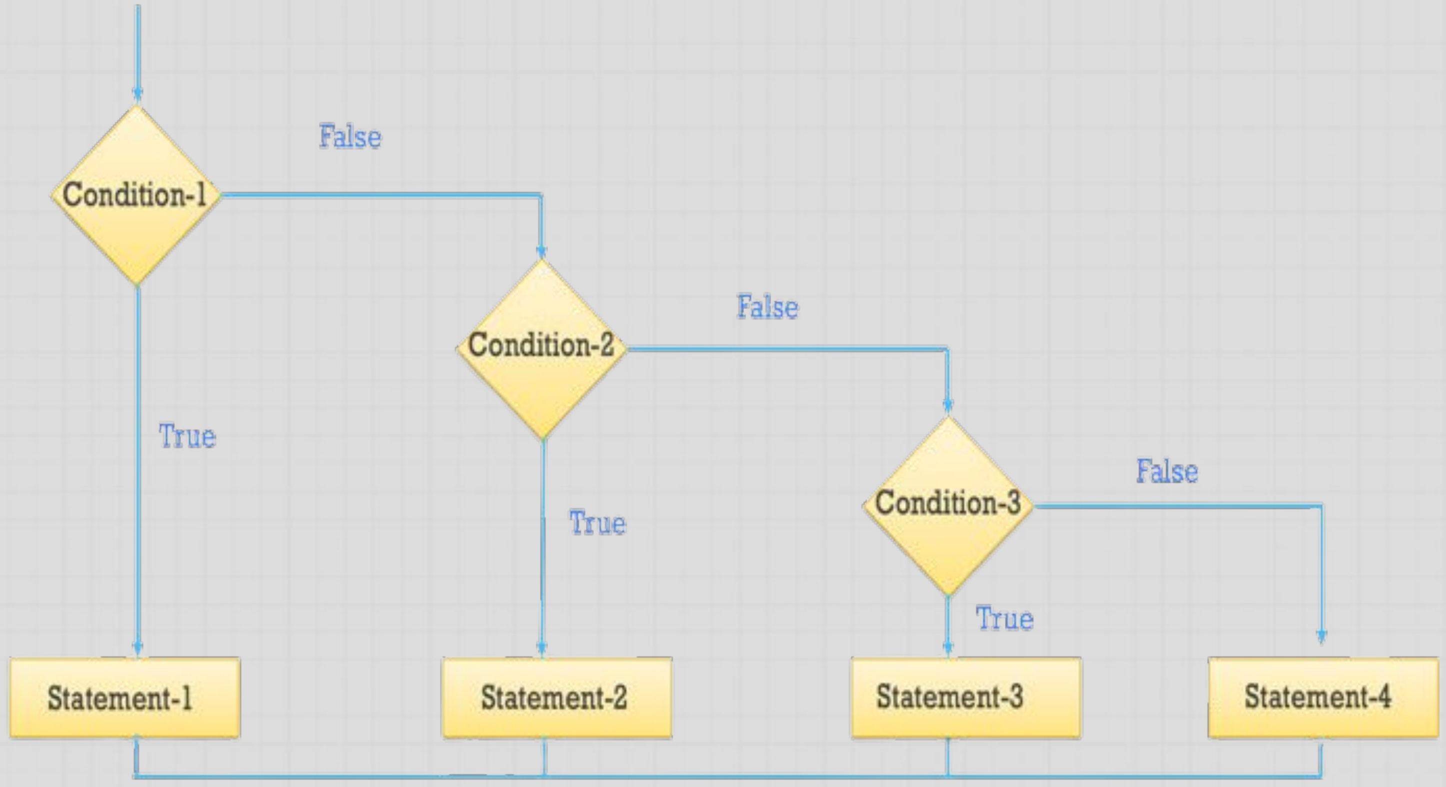
This code fragment is within the scope of the if statement

This code fragment is within the scope of the else statement

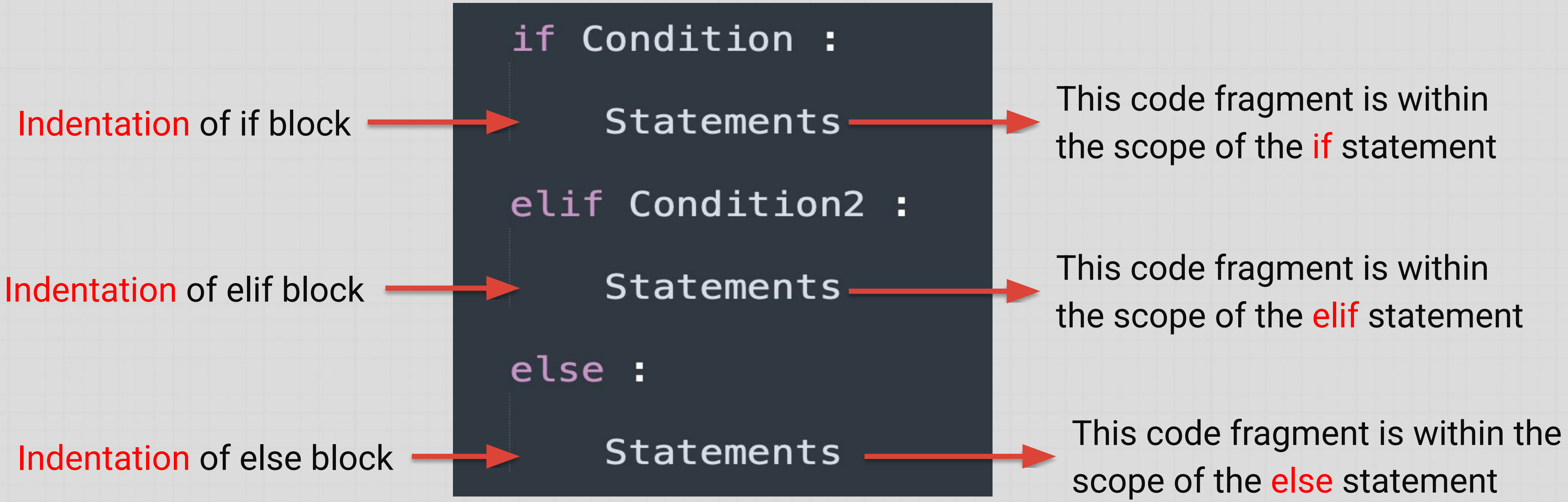
To define scope of the if statement, indentation (whitespaces at the beginning of line) is needed

Multi-branch If

- Multi-branch if statement can be used to create an else if clause
- It is used to make decision among several alternatives



Multi-branch If- Syntax



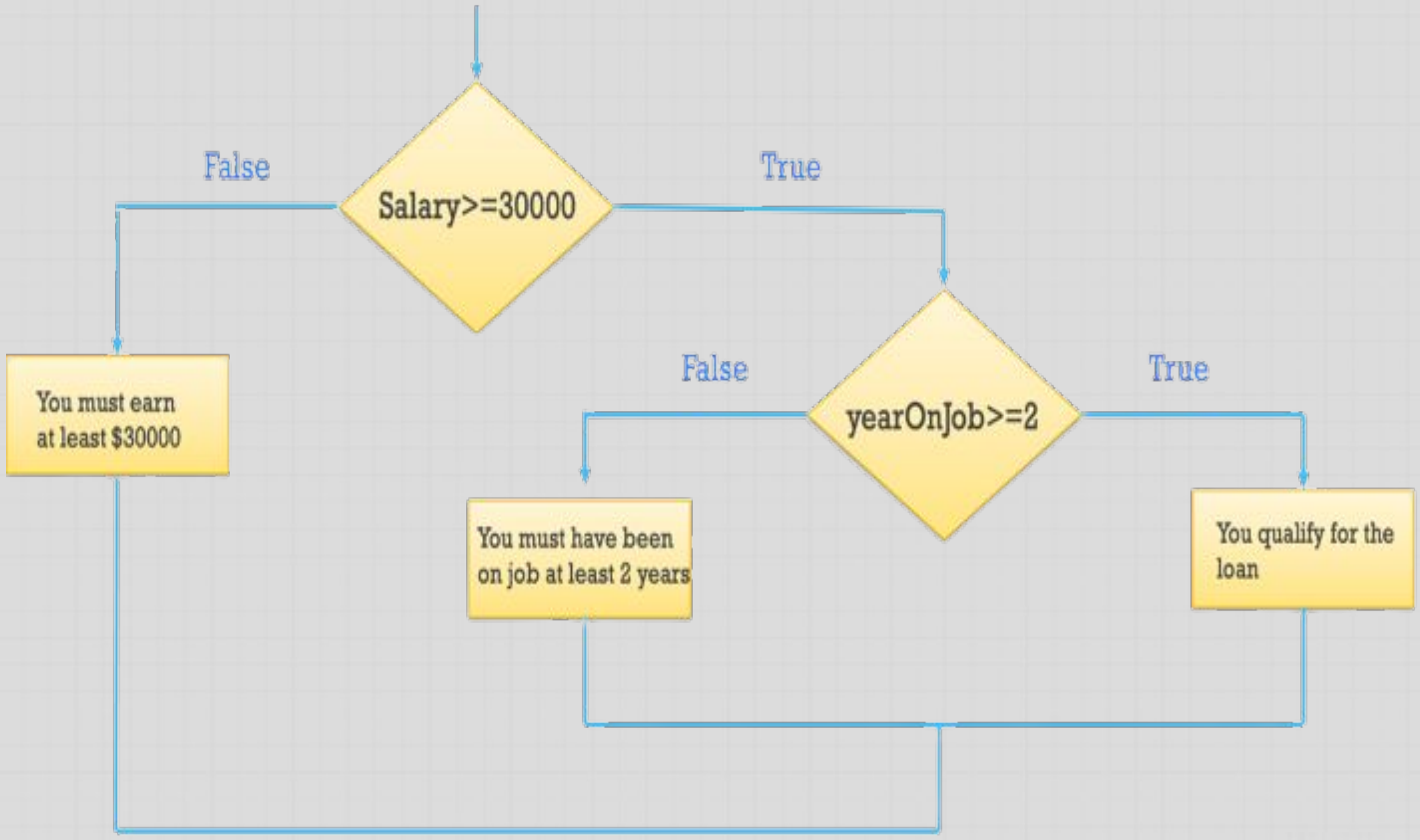
```
if Condition :
    Statements
elif Condition2 :
    Statements
else :
    Statements
```

Indentation of if block
Indentation of elif block
Indentation of else block
This code fragment is within the scope of the if statement
This code fragment is within the scope of the elif statement
This code fragment is within the scope of the else statement

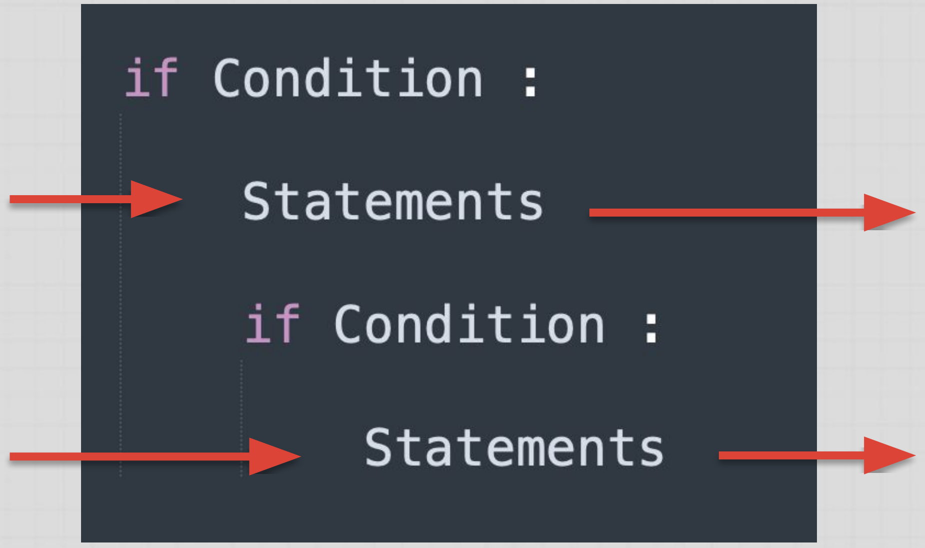
⚠ Multiple elif blocks can be given if needed

Nested If

- Nested if statements can be used for creating a pre-condition
- It's used if one condition can be evaluated to several alternatives


Salary>=30000
False
True
You must earn at least $30000
yearOnJob>=2
False
True
You must have been on job at least 2 years
You qualify for the loan

Nested If- Syntax



```
if Condition :
    Statements
    if Condition :
        Statements
```

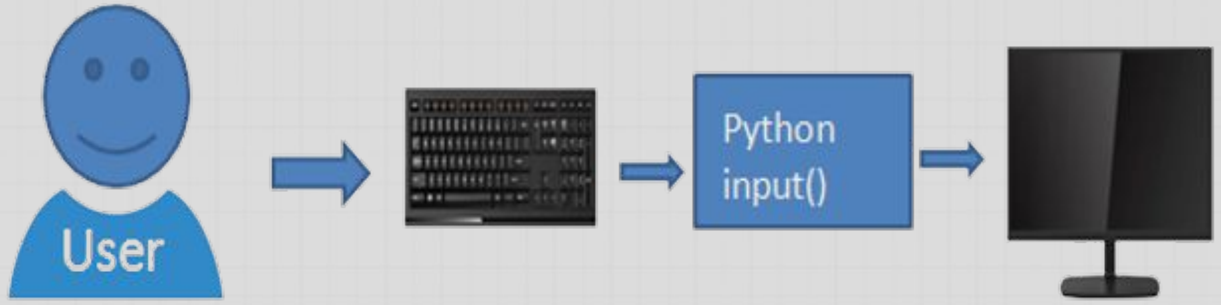
This code fragment is within the scope of the outer if statement

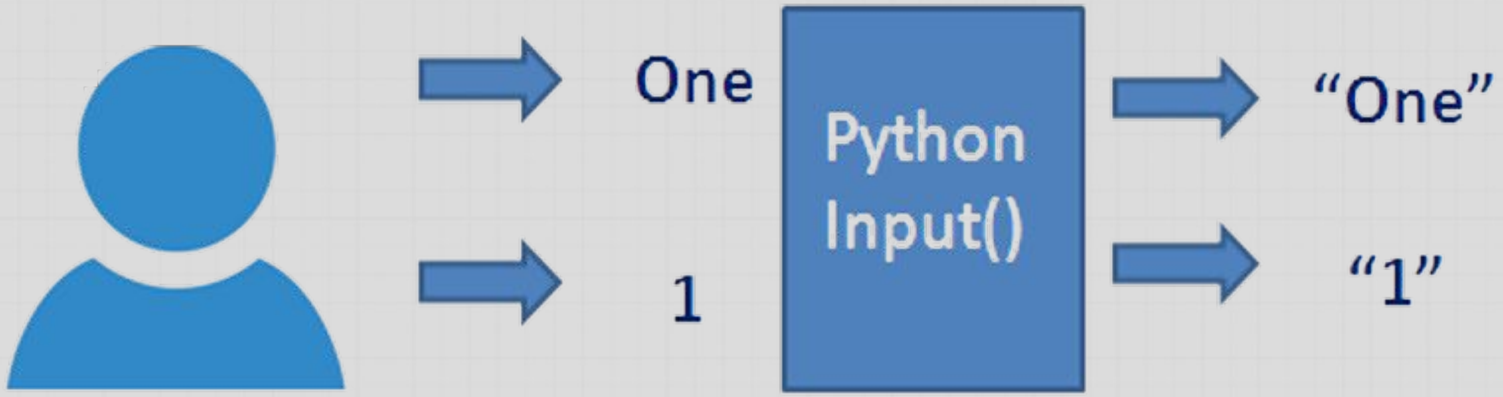
This code fragment is within the scope of the inner if statement

Outer and Inner If statements can be any type of if statement (Single if, If...else and Multi-branch If)

User Input

- We can ask the user for input by using the input() method
- It waits until the user provides input and returns it as a string





Input Method

```
name = input()

print("Hello "+name)
```

Waits for the user to provide an input

```
name = input("Enter your name:")

print("Hello "+name)
```

Displays the message “Enter your name:” first, then waits for the user to provide an input