

1. Indentation

Indentation refers to space at the beginning of a code line. In other programming languages indentation is used for easier reading, but in Python it is **very important** because it **indicates a block of code**.

Wrong

```
1 if 5 > 2:
2 #This belongs to IF statement block
3 #of code
4 print ("Five is greater than two")
5 #This doesn't belong to the IF
6 #statement block of code
```

Correct

```
1 if 5 > 2:
2     #This belongs to IF statement block
3     #of code
4     print ("Five is greater than two")
5 #This doesn't belong to the IF
6 #statement block of code
```

2. Comments

Comments can be used for explaining the code. They make code more readable. **A comment starts with # symbol.**

```
#This is a comment
print("Hello, World!")
```

If a comment is placed at the end of the line, Python will **ignore everything after** the comment if it is in the same line

```
print("Hello, World!") #This is a comment
```

We can use a multiple-line comment by using 3 quotation symbols (""") and the beginning and 3 quotation symbols (""") at the end of the comment.

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

3. Variables

Variables are containers for **storing data values**. Python doesn't have any command for declaring variables. A variable is created the moment we assign the value to it.

```
x = 5
y = "John"
```

A variable can have a **short name** (1 letter) or more **descriptive name** (age, carname, total_volume)

Rules for Python variables:

- A variable must start with a letter or underscore
- A variable can not start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, _)
- Variable names are case-sensitive (age, Age, AGE, agE, aGe are all different)

```
#Correct names      #Incorrect names
myVar = "John"      2myvar = "John"
my_var = "John"     my-var = "John"
_my_var = "John"    my var = "John"
myVar = "John"
```

4. Data types

Data type is an important concept in programming because **it defines the data** that is **stored** in the **variable**. Variables can store data of different types, and **different types can do different things**.

Example of data types:

```
Text Type: str
Numeric Types: int, float
Sequence Type: list
Boolean Type: bool
```

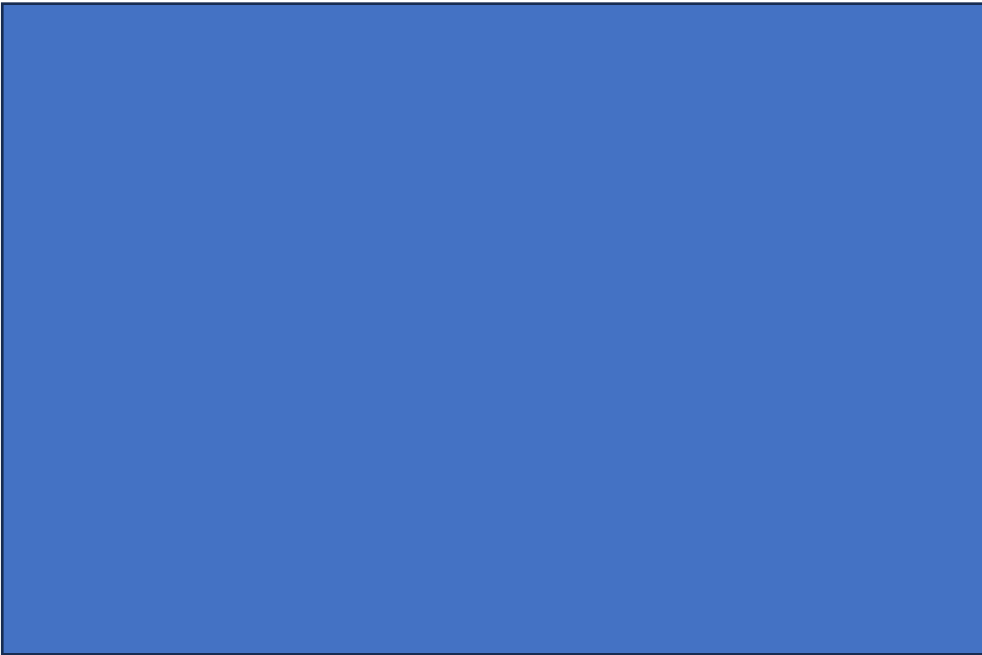
5. Output

To output information on the screen we use `print()` function

```
print("Hello World")
```

To output a variable, we have to use `+` symbol, we can **combine text** and a variable that is **string type only**.

```
x = "awesome"  
print("Python is " + x)
```



The most easiest way to show a number with text is to use a `,` (comma) operator. This will add the number next to your text.

```
1 SArea = 50  
2 TArea = 25  
3 print("Square area is",SArea,"and triangle area is",TArea)
```

6. Input

Input is used when we want to get some information from the user, can be text or number

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

Everything that the user writes will be text, if we need a number, we will need to convert it to the correct data type

```
age = int(input("Write your age: "))
```

```
weight = float(input("Write your weight"))
```

7. Strings

A **string** is a textual type of data. It can contain 1 character, 1 word or more words. Strings in Python are surrounded by either single quotation marks, or double quotation marks.

```
print("Hello")  
print('Hello')
```

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

```
a = "Hello"
```

We can assign a **multiline string** to a variable by using **three quotes or three single quotes**:

```
a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""  
print(a)
```

```
a = '''Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.'''  
print(a)
```

We can **measure the length** of a string by using the `len()` function

```
a = "Hello, World!"  
print(len(a))
```

We can **combine** two or more strings into one string by using `+` operator

```
a = "Hello"  
b = "World"  
c = a + " " + b  
print(c)
```

8. Numbers

There are three numeric types in Python:

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

Int, or **integer**, is a whole number, positive or negative, without decimals, of unlimited length.

```
x = 1
y = 35656222554887711
z = -3255522
```

Float, or "**floating point number**" is a number, positive or negative, containing one or more decimals

```
x = 1.10
y = 1.0
z = -35.59
```

We can convert from one type to another with the `int()` and `float()` methods:

```
x = 1    # int
y = 2.8  # float

#convert from int to float:
a = float(x)

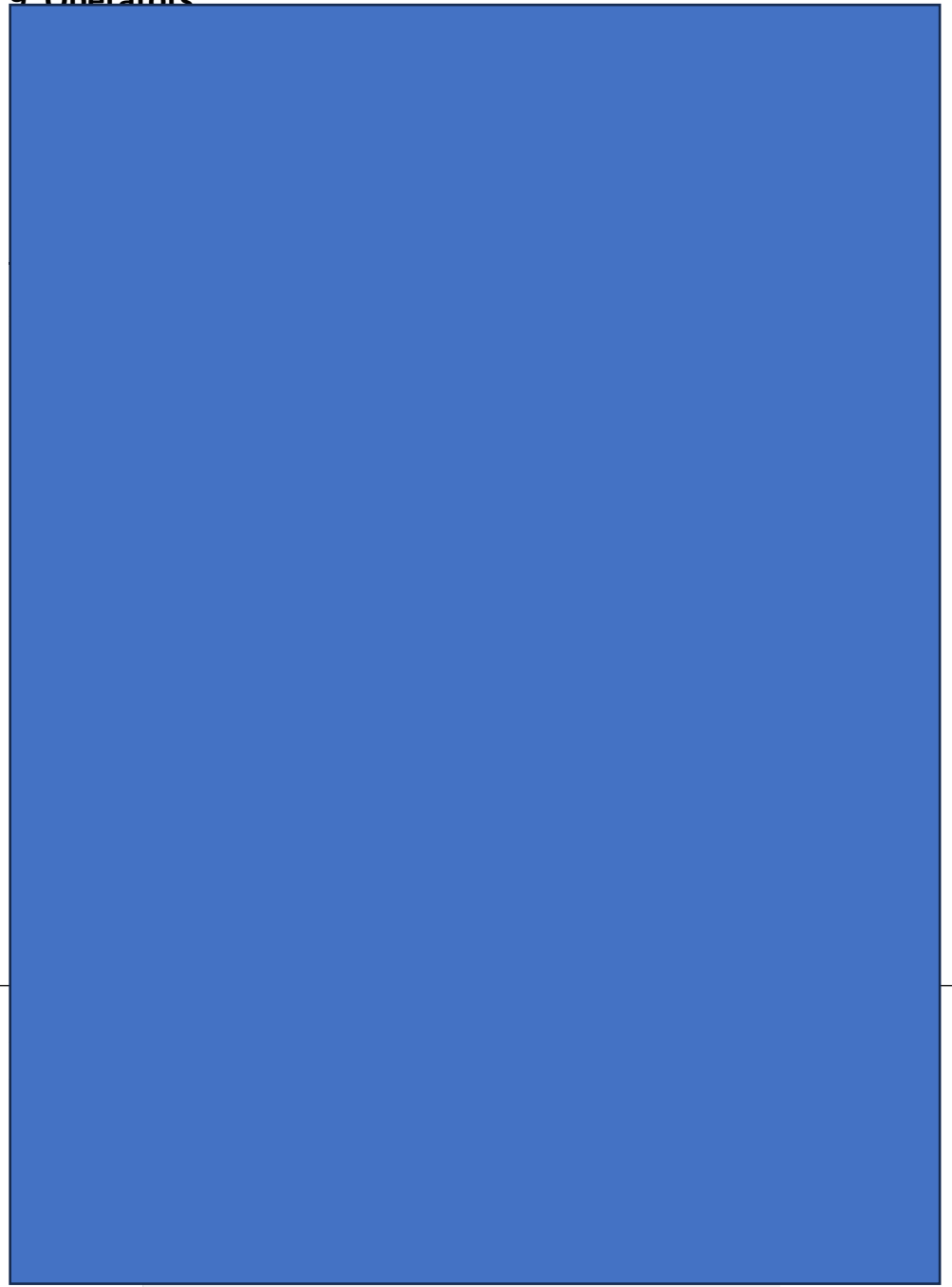
#convert from float to int:
b = int(y)
```

Python has a built-in module called `random` that can be used to make random numbers:

```
import random

print(random.randrange(1, 34))
```

9. Operators



/=

x /= 3

x = x / 3

Comparison Operators

Comparison operators are used to **compare two values**:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

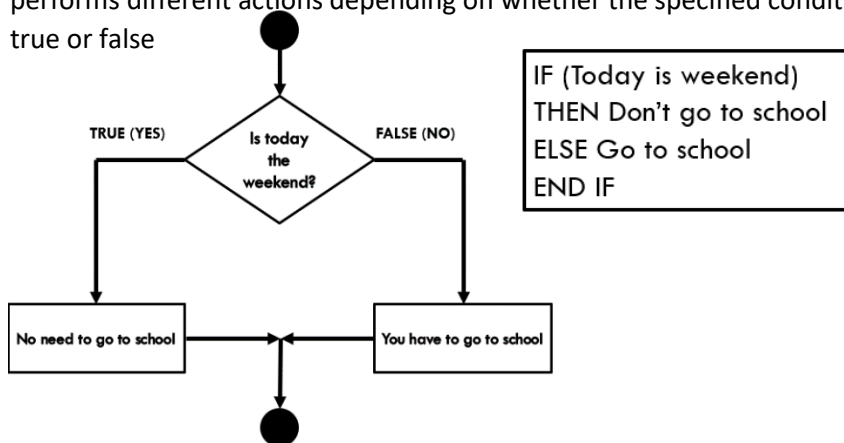
Logical Operators

Logical operators are used to **combine conditional statements**:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

10. Conditional Statements

A **conditional statement** is a feature of a programming language, which performs different actions depending on whether the specified condition is true or false



IF statements

IF statements are used to check IF a condition is true or not, if it is true then we will do an action

```
1 a = 33
2 b = 200
3 if b > a:
4     print("b is greater than a")
```

ELSE

ELSE is used when our conditions are not correct and we still want to do something. **Has** to go with IF

```
1 a = 33
2 b = 200
3 if b < a:
4     print("b is less than a")
5 else:
6     print("b is greater than a")
```

ELIF (ELSE-IF)

If we have **more than one condition**, we have to use **ELIF**. The **ELIF** keyword is Python's way of saying "if the **previous** conditions were **not true**, then **try this condition**"

```
1 a = 33
2 b = 33
3 if b < a:
4     print("b is less than a")
5 elif b == a:
6     print("b is same as a")
7 else:
8     print("b is greater than a")
```