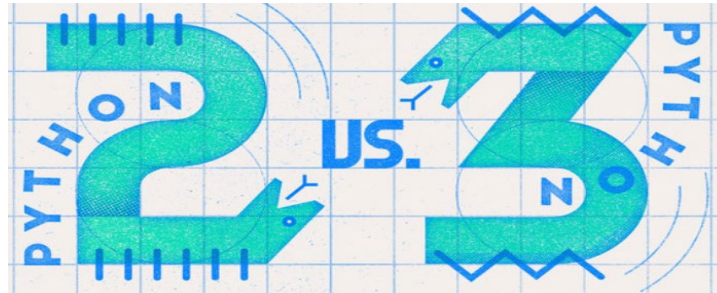




# Python Syntax



# Program Sequence

---

Programs run line by line, from top to bottom.

```
x = 42
y = 3.12
z = "Hans"
print(f"The value of x = {x}")
print(f"The value of y = {y}")
print(f"The value of z = {z}")
print(f"The value of x + y = {x + y}")
```



Program flow  
direction

Output



```
The value of x = 42
The value of y = 3.12
The value of z = Hans
The value of x + y = 45.12
```




# Python as calculator

---

Python knows the *normal* basic operators `+`, `-`, `*` and `/`

```
print( 341 + 56/4 - 4*12 )  
print( 1344 - 16*4 + 42/12 )
```



```
307.0  
1283.5
```

... as well as special operators like `**` (exponentiation), `//` (floor division), and `%` (modulo)

```
print( 2**4 )  
print(17 // 4)  
print(20 % 3)
```



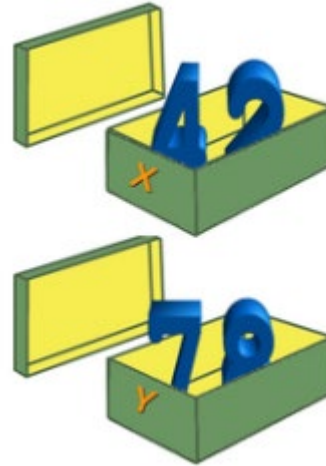
```
16  
4  
2
```

# Variables

---

Variables can be imagined as "containers" in which a value is stored

```
x = 42  
y = 78
```





# Assignment

---

The equal sign "=" assigns a value to a variable

|                         |   |                                   |
|-------------------------|---|-----------------------------------|
| <code>x1 = 42</code>    | ← | x1 contains the number value 42   |
| <code>x2 = 3.12</code>  | ← | x2 contains the number value 3.12 |
| <code>p = "Hans"</code> | ← | p contains the text value Hans    |

# Variables / Values

---

The value of a variable can then be used later.

```
x1 = 42
x2 = 3.12
x3 = -1.12

print(f"The value of x1 + x2 * x3 = {x1+x2*x3}")
print(f"The value of x1 * x2 - x3 = {x1*x2-x3}")
```

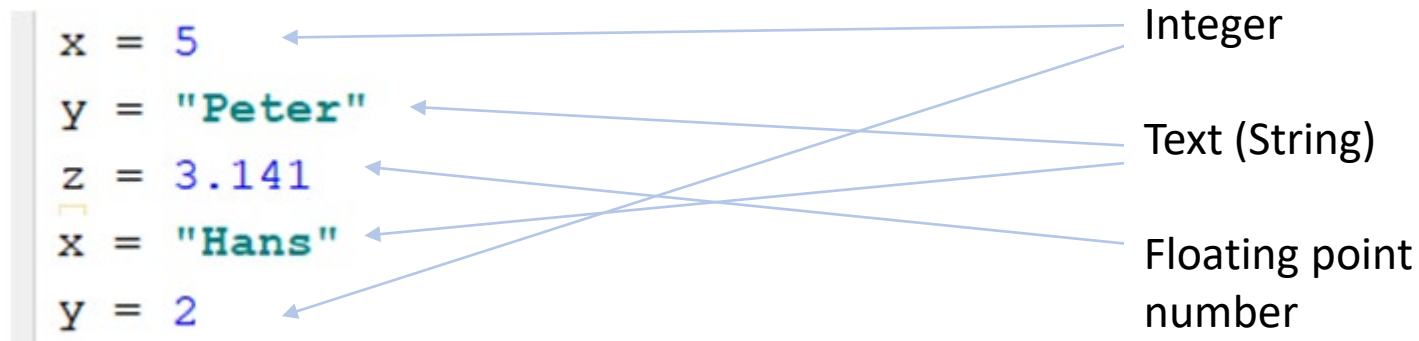
Output



```
The value of x1 + x2 * x3 = 38.5056
The value of x1 * x2 - x3 = 132.16
```

# Creating variables

- Variables are created by value assignment.
- The type of variable is determined at run time and may change.



In other programming languages all variables must be declared before they can be used, and they cannot change their type.

# Variables without value

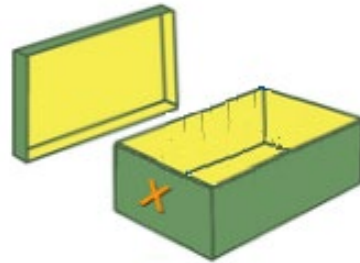
---

Variables can also have no value at the beginning.

```
x = None
```

Variable without value

None is a python  
keyword







# Variable Names (identifiers)

---

Names / identifiers may contain the following characters

- Capital letters A to Z
- Lowercase letters a to z
- Underscore \_
- The numbers 0 to 9 (not at the beginning: 5x)
- Examples

```
height = 23  
x5 = 1.2  
last_name = "Meier"  
minValue = -12
```



# Structure by indentation

---

In Python, the code is structured by indentation (here if-branches)

```
v1 = 3*4+16-9
v2 = 3*4+8+2-18/3
if 5 < v1:
    print(f"5 is smaller than {v1}")
if 30 > v2:
    print(f"30 is greater than {v2}")
    print(f"{v1} is smaller than {v2}")
```

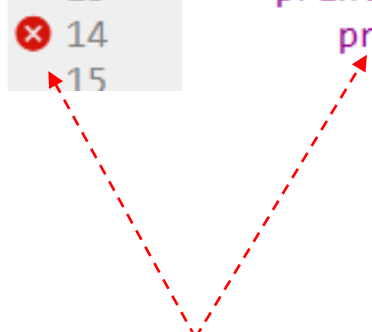
Most other programming languages use parentheses to structure the code.

# Structure by indentation

---

If the indentation is missing or unnecessary this is displayed in the editor as an error (red)

```
8  v1 = 3*4+16-9
9  v2 = 3*4+8+2-18/3
10 if 5 < v1:
11     print(f"5 is smaller than {v1}")
12 if 30 > v2:
13     print(f"30 is greater than {v2}")
14     print(f"{v1} is smaller than {v2}")
15
```



Error, wrong indentation!

# Comments / Remarks

---

Lines after # -signs are comments or remarks

```
v1 = 3*4+16-9
v2 = 3*4+8+2-18/3
# compare v1 with 5
if 5 < v1:
    # print only if 5 < v1
    print(f"5 is smaller than {v1}")
if 30 > v2:
    # print only if 30 < v2
    print(f"30 is greater than {v2}")
    print(f"{v1} is smaller than {v2}")
```

Single comment lines

# Comments / Remarks

---

- Multiline comments can be written within `"""` characters.

```
"""  
The following lines show the python syntax  
"""  
  
v1 = 3*4+16-9  
v2 = 3*4+8+2-18/3  
# compare v1 with 5  
if 5 < v1:  
    # print only if 5 < v1  
    print(f"5 is smaller than {v1}")  
if 30 > v2:  
    # print only if 30 < v2  
    print(f"30 is greater than {v2}")  
    print(f"{v1} is smaller than {v2}")
```

Multi-line comments