Python Data Types

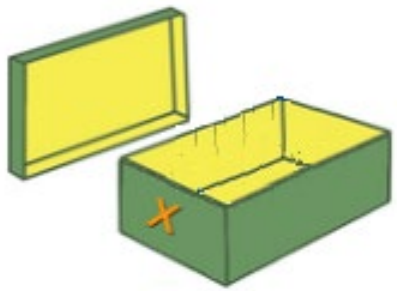
Data types in Python

- None → no data type
- bool → boolean type (True, False)
- int → integers (0, 1, -1, 2, …)
- float → floating-point numbers (0.1, -1.2, 3.41, …)
- str →strings (texts, like “gugus”, “hello”, ‘here we are’,…)

None

- None means that a variable does not yet contain a value.
- Makes sense if you want to define a variable but want to assign a value to it later.
- Example: The result of a search, which can also be unsuccessful (without result).

```
x = None
```



bool: boolean type

- True or False symbolizes in Python that a test is true (or false)
- Used for decisions (`if a < 5 :`)

```
x = 3957*1874
print(x > 100000)

z1 = x > 10000
z2 = x > 100000

print(f"z1 = {z1}")
print(f"z2 = {z2}")
print(type(z1))
```

Resultat

```
True
z1 = True
z2 = True
<class 'bool'>
```

int: Integer type

- The variables x1, x2, x3, x4, and z have the type int (integer).
- These variables can be used for integer arithmetic calculations (+, -, * , /, ...)

```
x1 = 2
x2 = 12
x3 = pow(4,x1)
x4 = 4
z = x1 * 5 + x2 % 5 + x3 - x4
print(f"z = {z}")
print(type(z))
```

Result

```
z = 24
<class 'int'>
```

- pow(a,b) calculates a^b, a%b returns the remainder of the division of a and b

int: Integer type

- In the following example, x1, x4, and z have the type int.
- Values that are read (via input) by the user are always of type string (=text), even if the user has entered a number.
- The texts must first be converted into integers using the int function (type conversion).

```
x1 = 2
x2 = input("Please enter the first number \n")
x3 = input("Please enter the second number \n")
x4 = 4
z = x1 * int(x2) + int(x3) - x4
print(z)
```

Results

```
Please enter the first number
5
Please enter the second number
12
18
```

float: Floating point number type

The variables f1, f2, f3 and w have the type float (floating point number).
The delimiter sign is actually a period (.).

```
f1 = 24.8 / 4
f2 = -35.5
f3 = f1 + f2
print(f3)
print(type(f3))
```

Results

```
-29.3
<class 'float'>
```

```
import math
w = math.sin(math.pi/2)
print(w)
```

Results

```
1.0
```

Rounding floating-point numbers

The round method can be used for rounding (and the cleaner display).
The second argument defines how many decimal places are displayed.

```
f1 = 24.8 / 3
f2 = -35.5
f3 = f1 + f2
print(f1)
print(f3)
print(round(f1,2))
print(round(f3,4))
```

Resultat

```
8.266666666666667
-27.233333333333334
8.27
-27.2333
```

str: Texts

Texts must always be placed in double quotes(") or single quotes(').
s1, s2, s3 have type str (string/text)
len returns the length of the text, texts can be joined by the + sign.

```
s1 = "Hans"
s2 = "Muster"
s3 = s1 + " " + s2

print(s3)
print(len(s1))
print(f"{s1}\t{s2}")
print(f"{s2}\t{s1}")
print(f"{s1}\n{s2}")
```

Results

```
Hans Muster
4
Hans    Muster
Muster  Hans
Hans
Muster
```

\t inserts a tab distance, \n inserts a new line

str: Index-Operator [ ]

With [idx] the letter at the point idx is accessed.

```
s1 = "Hans"
s2 = "Muster"
s3 = s1 + " " + s2
print(s1[0])
print(s2[3])
print(s2[-1])
print(s1[-2])
```

Resultat

```
H
t
r
n
```

First character of s1
Fourth character of s2
Last character of s2
Second last character of s1

str: Slices [* : *]

The index operator can also be used to get specific substrings.

```
myText = "Incomprehensibility"
print(myText[2:8])
print(myText[:8])
print(myText[-3:])
print(myText[0:10:2])
```

Result

```
compre
Incompre
ity
Icmrh
```

- The characters from 2 to 8
- The first 8 characters
- The last 3 characters
- Every second character, from the point 0