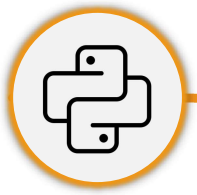
Numeric Data Type



Numeric Values

In this section we'll cover **numeric data types**, including how to convert between them, perform arithmetic operations, and apply numeric functions.

TOPIC WE'LL COVER	GOALS FOR THIS SECTION
Numeric Data Types	• Review the different numeric data types
Numeric Type Conversion	• Learn to convert between data types
Arithmetic Operators	• Perform moderately complex arithmetic operations
Numeric Functions	

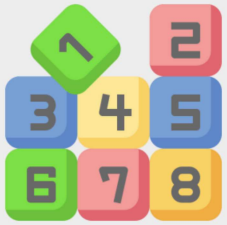
Numeric Data Types

There are three types of **numeric data types** in Python

Integers → `int`

Whole numbers <u>without</u> **decimal** points

Example: `26`, `11`, `2023`, `-7`



Floating-Point Numbers → `float`

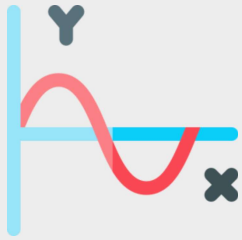
Real numbers <u>with</u> **decimal** points

Example: `3.14`, `0.0`, `1.6`, `-7.0`

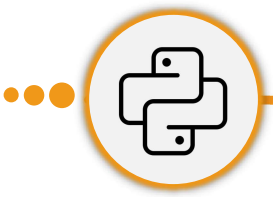
Complex → `complex`

Numbers <u>with</u> **real** & **imaginary** part

Example: `3 + 2j`, `-5j`



Numeric Type Conversion



Use `<data-type>(object)` to **convert any number** into a numeric data type

You can convert floats into integers

```
number = 1.5
type(number)
```

```
float
```

```
int_number = int(number)
int_number, type(int_number)
```

```
(1, int)
```

You can convert integers into floats

```
number = 1
float_number = float(number)
float_number, type(float_number)
```

```
(1.0, float)
```

You can convert strings into floats (or integers)

```
txt_num = "3.14"
type(number)
```

```
str
```

```
my_number = float(txt_num)
my_number, type(my_number)
```

```
(3.14, float)
```

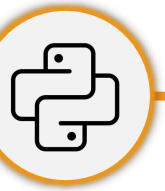
If the string contains **non-numeric** characters?

```
number = "314ab"
number = int(number)
```

```
ValueError: invalid literal for int() with
base 10: '314ab'
```

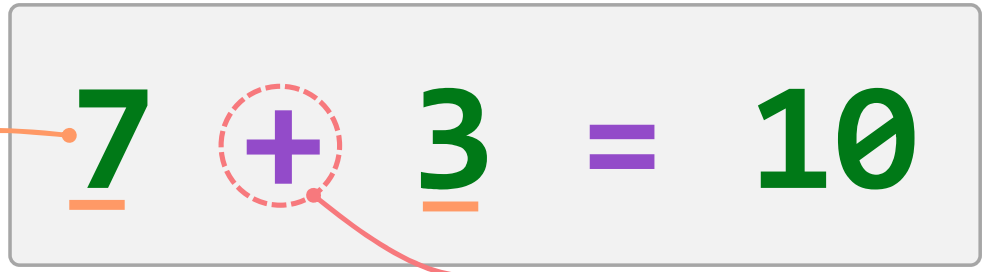
`ValueError` happens when a function receives an input/argument with an **inappropriate type**

Arithmetic Operators



1	**Addition (+)**	Adds Two Values	**10 + 7 = 17**
2	**Subtraction (-)**	Subtracts One Value From Another	**10 - 7 = 3**
3	**Multiplication (*)**	Multiplies Two Values	**10 * 7 = 70**
4	**Division (/)**	Divides One Value by Another	**10 / 7 = 1.428**
5	**Floor Division (//)**	Divides One Value by Another, then **Rounds down** to the nearest **integer**	**10 // 7 = 1**
6	**Modulo (%)**	Returns the Remainder of a Division	**10 % 7 = 3**
7	**Exponentiation (**)**	Raises One Value to the Power of Another	**5 ** 3 = 125**

Operators & Operands



Operands : The values that an operator acts on are called operands
››› 7 & 3 are operands

Operator : are special symbols that designate that some sort of computation should be performed

```
7 + "3"
```

```
TypeError: unsupported operand
type(s) for int and 'str'
```

Order of Operators

Python uses the standard **PEMDAS** order of operations to perform calculations

- **P**arentheses
- **E**xponentiation
- **M**ultiplication & **D**ivision (including floor division, modulo), from left to right
- **A**ddition & **S**ubtraction, from left to right

Without Parentheses

```
2*6 + 3 - 2**4 / 2
```
1. Exponentiation

```
2*6 + 3 - 16/2
```
2. Multiplication & division

```
12 + 3 - 8.0
```
3. Addition & subtraction (left to right)

```
7.0
```

With Parentheses (to control the order)

```
2*(6 + (3 - 2)**(4 / 2))
```
1. Subtraction & division in inner parentheses

```
2*(6 + 1**2.0)
```
2. Exponentiation in outer parentheses

```
2*(6 + 1.0)
```
3. Then addition in outer parentheses

```
2 * 7.0
```
4. Finally multiplication

```
14.0
```

Let’s Test What We’ve learned!

- Numeric Type Conversion
- Arithmetic Operations
- Order of Operations

Numeric_Data_Types_&_Arithmetic_01.ipynb

Assignment Operators

The sign = is used to **assign a value** to a variable in Python

```
my_var = 7
my_var = my_var + 1
my_var += 1
print(my_var)
```
8

```
my_var = 7
my_var = my_var - 1
my_var -= 1
print(my_var)
```
6

```
my_var = 7
my_var = my_var * 2
my_var *= 2
print(my_var)
```
14

```
my_var = 7
my_var = my_var / 2
my_var /= 2
print(my_var)
```
3.5

Shorthand

All Operators in Python

Python operators are listed in the **following table**

All Operators in Python	
Arithmetic Operators	+ - * / % // **
Assignment Operators	= += -= /= *= %= //= **=
Comparison Operators	> < >= <= == !=
Logical Operators	and or not
Identity Operators	is is not
Membership Operators	in not in
Bitwise Operators	& | ^ ~ << >>

Assignment: Arithmetic Operators

From: **Mohsen Abbasi (Financial Planner)**

Subject: **Financial Calculations**

Hi there!

I need your Python skill to calculate the following numbers: **(the details are in the attached jupyter file)**

- Gross profit from selling a **black shoes**
- Gross margin from selling a **black shoes**
- Price needed for gross margin of **60%**
- Sales tax on a **black shoes** sale
- Amount of money if the gross profit from selling **100 black shoes** is invested for one year

Financial_Calculations.ipynb

Reply Forward

----- Result Preview -----

```
print(gross_profit)
print(gross_margin)
print(price_needed_for60)
print(sales_tax)
print(money_after_1year)
```

```
6.68
0.2672
45.8
2.5
768.2
```

Solution: Arithmetic Operators

From: **Mohsen Abbasi (Financial Planner)**

Subject: **Financial Calculations**

Hi there!

I need your Python skill to calculate the following numbers: **(the details are in the attached jupyter file)**

- Gross profit from selling a **black shoes**
- Gross margin from selling a **black shoes**
- Price needed for gross margin of **60%**
- Sales tax on a **black shoes** sale
- Amount of money if the gross profit from selling **100 black shoes** is invested for one year

Financial_Calculations.ipynb

Reply Forward

----- Code Solution -----

```
gross_profit = blackshoes_price - blackshoes_cost
```

```
6.68
```

```
gross_margin = gross_profit / blackshoes_price
```

```
0.2672
```

```
desired_margin = 0.6
price_needed_for60 = blackshoes_cost / (1 - desired_margin)
```

```
45.8
```

```
tax_rate = 0.1
sales_tax = blackshoes_price * tax_rate
```

```
2.5
```

```
interest_rate = 0.15
invested_money = 100 * gross_profit
money_after_1year = invested_money + (invested_money *
interest_rate)
```

```
768.2
```

Numeric Functions

| Single-Number Functions |

round	Rounds a number to a specified number of digits	`round(number, number of digits to round)`
abs	Returns the absolute value of a number	`abs(number)`

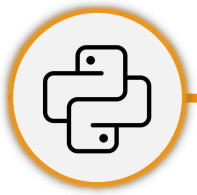
More than one argument, separated by **comma**

| Multiple-Number Functions |

sum	Sums all numbers in an iterable	`sum(iterable)`
min	Returns the smallest value in an iterable	`min(iterable)`
max	Returns the largest value in an iterable	`max(iterable)`

1. List
2. Tuple
3. Set
...

Numeric Functions → Round & Abs



1. The `round( )` function rounds a number to a specified number of digits
2. The `abs( )` function returns the absolute value of a number

Some Examples

```
round(3.141592, 2)
```

This rounds the number to **2 decimals places**

```
3.14
```

```
abs(-3)
```

Always returns a **positive** number

```
3
```

```
round(3.141592)
```

If number of digits **isn't** provided, it will round to the **nearest integers**

```
3
```

```
round(9.51)
```

It rounds down if **< 0.5**
It rounds up if **>= 0.5**

```
10
```

Numeric Functions → Sum, Min, Max

1. The `sum( )` function performs **sum** operation on an iterable
2. The `min( )` function performs **minimum** operation on an iterable
3. The `max( )` function performs **maximum** operation on an iterable

Some Examples

```
sum([10, 20, 30])
```

This **sums** the number in the list

```
60
```

```
min((10, 20, 30))
```

This finds the **minimum** value in the tuple

```
10
```

```
max({10, 20, 30})
```

This finds the **maximum** value in the set

```
30
```

- **Remember**

Do not use `sum`, `min`, `max` as a variable name in your code

```
TypeError : 'int' object is not callable
```

Let's Test What We've learned!

- Numeric Functions in Python
- `round()`, `abs()`
- `sum()`, `min()`, `max()`

Numeric_Functions_01.ipynb

Assignment: Numeric Functions

From: **Salar H Shamchi (izshop manager)**

Subject: **FAQ (Frequently Asked Questions)**

Hi there!

Can you quickly calculate the following numbers for me?

- Which price is the **highest** one?
- Which price is the **lowest** one?
- Can you calculate the **average** for prices in the list?
  (**you need to round it to the nearest euro**)

Thank You

Salar_Questions.ipynb

Reply | Forward

----- Result Preview -----

```
prices_list = [
39, 8.5, 8, 24.99, 10, 12, 35, 9, 33.5, 44, 53.99, 11,
21, 11.9, 21, 28.9, 19.9, 22, 7, 14, 20, 12, 11, 9, 10
]

lowest_price = min(prices_list)
highest_price = max(prices_list)

print(highest_price, lowest_price)
```

```
53.99  7
```

```
prices_num = 25
round(sum(price_list)/prices_num)
```

```
20
```

Solution: Numeric Functions

From: **Salar H Shamchi (izshop manager)**

Subject: **FAQ (Frequently Asked Questions)**

Hi there!

Can you quickly calculate the following numbers for me?

- Which price is the **highest** one?
- Which price is the **lowest** one?
- Can you calculate the **average** for prices in the list?
  (**you need to round it to the nearest euro**)

Thank You

Salar_Questions.ipynb

Reply | Forward

----- Code Solution -----

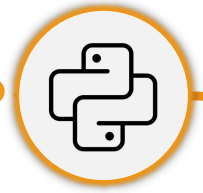
```
prices_list = [
39, 8.5, 8, 24.99, 10, 12, 35, 9, 33.5, 44, 53.99, 11,
21, 11.9, 21, 28.9, 19.9, 22, 7, 14, 20, 12, 11, 9, 10
]
lowest_price = min(prices_list)
highest_price = max(prices_list)

print(highest_price, lowest_price)
```

```
53.99  7
```

```
prices_num = 25
round(sum(price_list)/prices_num))
```

```
20
```



Section Wrap-Up

- ✔ Data Analyst typically work with **integer** & **float** numeric data type
- ✔ Arithmetic operations follow **PEMDAS** order of operations
  - Use **parentheses** to control the order in mathematic operations
- ✔ Python has **built-in functions** that work with numbers
  - These functions are `round()`, `abs()`, `sum()`, `min()`, `max()` helping you in simple but frequent operations