# Built-in funkce

# Co jsou built-in funkce?

- Funkce, které jsou součástí Pythonu a provádějí běžné nebo užitečné operace.
- Např. zjišťování délky stringu/seznamu, převádění mezi datovými typy, přidávání prvků do seznamu, řazení seznamu apod.
- Některé built-in funkce lze použít na různé datové typy a některé se vážou ke konkrétním typům.

# Built-in funkce

## Obecné

# Funkce `abs`

**Vrací absolutní hodnotu celého/desetinného čísla.**
Například

- `abs(-5) == 5`,
- `abs(2.718) == 2.718`.

Pro žádný jiný běžný datový typ nefunguje.

# Funkce enumerate

**Vytvoří ze seznamu/n-tice seznam dvojic (pořadí, hodnota).**

Obvykle se používá s for cyklem. Např.

```
lst = ["pes", "kočka"]
for i, element in enumerate(lst):
    print(i, element)
```

vytiskne

```
0 pes
1 kočka
```

# Funkce `filter`

**Dostává dva parametry – funkci a seznam/n-tici/slovník. Vrátí jako seznam prvky, pro které funkce vrací `True`.**

Například, pokud si definuju funkci

```
def divisible_by_five(cislo):
    return (cislo % 5 == 0)
```

pak můžu ze seznamu `lst = [4, 5, 10, 13]` dostat seznam všech čísel dělitelných pěti jako

```
lst2 = filter(divisible_by_five, lst)
```

# Funkce `input`

**Vrátí vstup od uživatele jako typ `string`. Jako parametrem dostává string, který se uživateli zobrazí.**

Například, napíšu-li

```
birth_year = input("Zadejte svůj rok narození: ")
```

uživateli se zobrazí

```
Zadejte svůj rok narození:
```

a odpověď se uloží **jako string** do proměnné `birth_year`.

# Funkce `isinstance`

**Dostane dva parametry – proměnnou a datový typ – a odpoví, zda je proměnná onoho typu.**

Například,

- `isinstance('kočka', str) == True`,
- `isinstance(3.14, float) == True`,
- `isinstance(4, list) == False` a
- `isinstance([1, 2], tuple) == False`.

# Funkce `len`

**Parametrem dostane cokoli, kde má „smysl“ počítat počet prvků. Ten vrátí jako celé číslo.**

Například

- `len("auto") == 4`,
- `len([4, 5, 6]) == 3` a
- `len({1: "jedna", 2: "dva"}) == 2`.

# Funkce map

**Dostane parametrem funkci a seznam/n-tici/slovník a vrátí seznam s funkcí aplikovanou na každý prvek.**

Například s funkcí

```
def last_letter(word):
    return word[-1]
```

můžu ze seznamu slov `words = ["pes", "kočka", "dikobraz"]` získat seznam jenom jejich posledních písmen jako

```
last_letters = map(last_letter, words)
```

# Funkce `max`/`min`

**Ze seznamu/n-tice/slovníku vrátí největší/nejmenší prvek.**

Například,

- `max([4, 7, 2]) == 7`,
- `min([4, 7, 2]) == 2`,
- `max(["pes", "kočka", "dikobraz"]) == "pes"`,
- `min(["pes", "kočka", "dikobraz"]) == "dikobraz"`.

# Funkce `print`

**Vytiskne to, co dostane parametrem, do konzole.**

Například

- `print(3)` vytiskne číslo 3 a
- `print("kočka")` vytiskne kočka.

# Funkce `range`

**Dostane parametrem celé číslo a vrátí seznam celých čísel menších než toto číslo počínaje `0`.**

Nejčastěji se používá s `for` cyklem. Tedy, např.

```
for i in range(5):
    print(i * 2)
```

vytiskne postupně čísla `0, 2, 4, 6, 8`, protože `range(5) == [0, 1, 2, 3, 4]`.

# Funkce `round`

**Zaokrouhlí dané desetinné číslo – od `.5` nahoru, jinak dolu.**

Například,

- `round(3.14) == 3` a
- `round(2.718) == 3`.

# Funkce `sum`

**Vrátí součet všech prvků v daném seznamu/n-tici. Funguje pouze pro čísla.**
Například,

- `sum([1, 2, 3, 4]) == 10`,
- `sum((69, 420, 1337)) == 1826`.

# Funkce `zip`

**Dostane parametrem libovolný počet seznamů/n-tic a vrátí seznam, kde každým prvkem je n-tice prvků na odpovídajících pozicích těchto seznamů/n-tic.**

Například,

- `zip([1, 2, 3], [4, 5, 6]) == [(1, 4), (2, 5), (3, 6)]`
- `zip(("kočka", "pes"), ("dělá", "dělá"), ("haf", "mňau")) == [("kočka", "dělá", "haf"), ("pes", "dělá", "mňau")]`.

# Funkce `int`, `float`, `str`, `list`, `tuple`, `dict`

**Převádějí jiné datové typy na stejnojmenné.**

Například,

- `int("5") == 5`,
- `str(1.23) == "1.23"`,
- `tuple([4, 5, 6]) == (4, 5, 6)`.