BootCamp - zajęcia 2

Zdania logiczne w Pythonie

In [1]:
```
a = 1
b = 2

print("Sprawdzenie równości:", a == b)
print("Sprawdzenie nierówności:", a != b)
print("Sprawdzenie większości:", a > b)
print("Sprawdzenie większości lub równości:", a >= b)
print("Sprawdzenie mniejszości:", a < b)
print("Sprawdzenie mniejszości lub równości:", a <= b)
```

```
Sprawdzenie równości: False
Sprawdzenie nierówności: True
Sprawdzenie większości: False
Sprawdzenie większości lub równości: False
Sprawdzenie mniejszości: True
Sprawdzenie mniejszości lub równości: True
```

Operatory logiczne w Pythonie: and, or, not, in, is

In [2]:
```
c = 3
d = 4
e = 5
f = 6
liczby = [1, 2, 3, 4]

print("Koniunkcja:", c == d and f > e)
print("Alternatywa:", c == d or f > e)
print("'Jest w':", c in liczby)
print("Zaprzeczenie:", f not in liczby)
print("Zaprzeczenie:", c not in liczby)
print("'Czy jest tą samą zmienną?':", c is f)
```

```
Koniunkcja: False
Alternatywa: True
'Jest w': True
Zaprzeczenie: True
Zaprzeczenie: False
'Czy jest tą samą zmienną?': False
```

Składnia instrukcji warunkowej if

In [3]:
```
g = 7
h = 8

if g < h:
    print("{} jest mniejsze od {}".format(g, h))
```

```
7 jest mniejsze od 8
```

Słowa kluczowe elif oraz else

In [4]:
```
wiek = 16

if wiek < 10:
    print("dziecko")
elif 10 <= wiek < 18:
    print("nastolatek")
elif 18 <= wiek < 65:
    print("dorosły")
else:
    print("senior")
```

```
nastolatek
```

Short hand if

In [7]:
```
imie = "Jan"
print("Witaj Janie") if imie == "Jan" else print("Witaj nieznajomy")
```

```
Witaj Janie
```

Instrukcja switch case jako słownik

In [1]:
```
switcher = {
    "czekolada":3.8,
    "baton": 2.45,
    "cola": 7.88,
    "paluszki solone": 5.2,
    "tiger": 4.35
}
```

In [2]:
```
produkt = input('Podaj nazwę produktu: ')
if produkt in switcher.keys():
    print("Cena za produkt to:", switcher[produkt])
else: print("Nie ma takeigo produktu")
```

```
Cena za produkt to: 7.88
```

Wyrażenie lambda

In [6]:
```
x2 = lambda x: x**2

print(x2(6))
print(x2(-5))
```

```
36
25
```

In [8]:
```
czyJPG = lambda file: file.endswith(".jpg")

print(czyJPG("zdjecie.jpg"))
print(czyJPG("skrypt.py"))
print(czyJPG("obrazek.png"))
```

```
True
False
False
```

In [12]:
```
czyListaJestDluga = lambda lista: len(lista) > 10

print(czyListaJestDluga([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]))
print(czyListaJestDluga([1, 2, 3, 10, 11, 12, 13]))
```

```
True
False
```

Wyrażenie lambda to funkcja zapisana za pomocą prostrzej, jednolinijkowej składni. Bardziej złożone funkcje tworzymy za pomocą słowa kluczowego *def*, nauczymy się tego na jednych z kolejnych zajęć

Funkcja *map()* oraz *filter()*

In [13]:
```
numery = [1, 2, 3, 4, 5, 6, 7]

numeryPodwojone = map(lambda x: 2*x, numery)
print(numeryPodwojone)
```

In [15]:
```
numeryPodwojoneLista = list(numeryPodwojone)
print(numeryPodwojoneLista)
```

```
[2, 4, 6, 8, 10, 12, 14]
```

In [16]:
```
numeryDoKwadratu = list(map(lambda x: x**2, numery))
print(numeryDoKwadratu)
```

```
[1, 4, 9, 16, 25, 36, 49]
```

Przykład 1. Wobraźmy sobie, że pobraliśmy dane z API w postaci słownika:

```
data_from_api = [
{'city': 'Kraków', 'province_id': 8, 'current_temp': 3.5},
{'city': 'Warszawa', 'province_id': 1, 'current_temp': 2.8},
{'city': 'Suwałki', 'province_id': 9, 'current_temp': -0.5},
{'city': 'Gdańsk', 'province_id': 3, 'current_temp': -0.1},
{'city': 'Rzeszów', 'province_id': 7, 'current_temp': 3.9},
{'city': 'Wrocław', 'province_id': 2, 'current_temp': 5.0},
]
```

Teraz chcemy przeprowadzić reklasyfikację temperatur i zapisać ją w liście

In [19]:
```
data = [
    {'city': 'Kraków', 'province_id': 8, 'current_temp': 3.5},
    {'city': 'Warszawa', 'province_id': 1, 'current_temp': 2.8},
    {'city': 'Suwałki', 'province_id': 9, 'current_temp': -0.5},
    {'city': 'Gdańsk', 'province_id': 3, 'current_temp': -0.1},
    {'city': 'Rzeszów', 'province_id': 7, 'current_temp': 3.9},
    {'city': 'Wrocław', 'province_id': 2, 'current_temp': 5.0},
]
czyCieplo = list(map(lambda obj: "cieplo" if obj['current_temp'] > 1 else "zimno", data))
print(czyCieplo)
```

```
['cieplo', 'cieplo', 'zimno', 'zimno', 'cieplo', 'cieplo']
```

Teraz chcemy listę, w której pojawią się nazwy tylko danne dla tych miast, gdzie jest dodatnia temperatura

In [29]:
```
ciepleMiasta = list(filter(lambda city: city['current_temp'] >= 0, data))
print(ciepleMiasta)
```

```
[{'city': 'Kraków', 'province_id': 8, 'current_temp': 3.5}, {'city': 'Warszawa', 'province_id': 1, 'current_temp': 2.8}, {'city': 'Rzeszów', 'province_id': 7, 'current_temp': 3.9}, {'city': 'Wrocław', 'province_id': 2, 'current_temp': 5.0}]
```

Zadanie 11 - przykład rozwiązania

Otrzymałeś dane w postaci pliku JSON (plik JSON odpowiada strukturze słownika w Python).

Zapoznaj się ze strukturą danych i zaproponuj ich filtrowanie a następnie reklasyfikację za pomocą funkcji filter oraz map(). Dane pochodzą z API um Warszawy dostarczającego aktualne informacje o komunikacji miejskiej Dane do zadania to odjazdy autobusu linii 523 z przystanku Metro Politechnika

In [56]:
```
data = [
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"1","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"04:39:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"4","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"04:55:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"021","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"05:11:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"023","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"05:21:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"024","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"05:29:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"8","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"05:36:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"026","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"05:43:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"025","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"05:48:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"027","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"05:53:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"10","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"05:58:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"028","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"06:02:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"11","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-OBEM","key":"trasa"},{"value":"06:06:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"029","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"06:10:00","key":"czas"}]},
        {"values":[{"value":"null","key":"symbol_2"},{"value":"null","key":"symbol_1"},{"value":"12","key":"brygada"},{"value":"Stare Bemowo","key":"kierunek"},{"value":"TP-BEM","key":"trasa"},{"value":"06:14:00","key":"czas"}]}
]
```

In [57]:
```
# Wybranie odjazdów które następują po godz. 6:00 rano - przykład filtrowania
odjazdyPo6Rano = list(filter(lambda obj: obj['values'][5]['value'].startswith('06'), data))
print(odjazdyPo6Rano)
print(30*'<-->')
# Wybranie tylko godzin odjazdu
godzinyOdjazdu = list(map(lambda obj: obj['values'][5]['value'], odjazdyPo6Rano))
print(godzinyOdjazdu)
print(10*'<-->')
# Reklasyfikacja
klasyfikacja = list(map(lambda h: "pierwsza połowa godziny" if int(h[3])<3 or int(h[3:5])==30 else "druga połowa godziny", godzinyOdjazdu))
print(klasyfikacja)
```

```
[{'values': [{'value': 'null', 'key': 'symbol_2'}, {'value': 'null', 'key': 'symbol_1'}, {'value': '028', 'key': 'brygada'}, {'value': 'Stare Bemowo', 'key': 'kierunek'}, {'value': 'TP-BEM', 'key': 'trasa'}, {'value': '06:02:00', 'key': 'czas'}]}, {'values': [{'value': 'null', 'key': 'symbol_2'}, {'value': 'null', 'key': 'symbol_1'}, {'value': '11', 'key': 'brygada'}, {'value': 'Stare Bemowo', 'key': 'kierunek'}, {'value': 'TP-OBEM', 'key': 'trasa'}, {'value': '06:06:00', 'key': 'czas'}]}, {'values': [{'value': 'null', 'key': 'symbol_2'}, {'value': 'null', 'key': 'symbol_1'}, {'value': '029', 'key': 'brygada'}, {'value': 'Stare Bemowo', 'key': 'kierunek'}, {'value': 'TP-BEM', 'key': 'trasa'}, {'value': '06:10:00', 'key': 'czas'}]}, {'values': [{'value': 'null', 'key': 'symbol_2'}, {'value': 'null', 'key': 'symbol_1'}, {'value': '12', 'key': 'brygada'}, {'value': 'Stare Bemowo', 'key': 'kierunek'}, {'value': 'TP-BEM', 'key': 'trasa'}, {'value': '06:14:00', 'key': 'czas'}]}]
<-->
['06:02:00', '06:06:00', '06:10:00', '06:14:00']
<-->
['pierwsza połowa godziny', 'pierwsza połowa godziny', 'pierwsza połowa godziny', 'pierwsza połowa godziny']
```

In [ ]:
```

```