Lambda funkcija

Lambda funkcije so anonimne funkcije, kar pomeni, da nimajo imena (niso vezane na spremenljivko).

```
lambda x,y : x +y
```

Sestavljene so iz:

- lambda - keyword
- parametri so napisani med lambda in :
- "single expression" (1 vrstica kode). Rezultat / vrednost tega "single expression" se vrne kot vrednost funkcije

In [1]:
```
(lambda x, y: x+y)(2, 3)
```

Out[1]:
```
5
```

In [2]:
```
add = lambda x, y: x + y
print(add)
print(type(add))
```

```
<function <lambda> at 0x000001D590FDDE50>
<class 'function'>
```

In [5]:
```
add(5,3)
```

Out[5]:
```
8
```

Primer, če bi zgornjo lambda funkcijo napisalo kot navadno funkcijo.

In [6]:
```
def add(x, y):
    return x + y
```

In [ ]:
```

```

Lambda funkcije pridejo najbolj do izraza, kjer je treba kot argument posredovati funkcijo. Namesto dejanske funkcije lahko posredujemo lambda funkcijo.

Za primer vzemimo funkcijo `sorted()` .

https://docs.python.org/3/library/functions.html#sorted

Naša naloga je sortirati sledeče vrednosti glede na **market_cap** vrednost, od največje do najmanjše.

In [13]:
```
data = [
  {
    "id": "binancecoin",
    "symbol": "bnb",
    "name": "Binance Coin",
    "image": "https://assets.coingecko.com/coins/images/825/large/binance-coin-logo.
    "current_price": 212.03,
    "market_cap": 33015186690,
    "total_volume": 2490184836,
    "high_24h": 230.59,
    "low_24h": 210.87,
  },
  {
    "id": "bitcoin",
    "symbol": "btc",
    "name": "Bitcoin",
    "image": "https://assets.coingecko.com/coins/images/1/large/bitcoin.png?15470335
    "current_price": 47553,
    "market_cap": 901453728232,
    "total_volume": 47427138554,
    "high_24h": 51131,
    "low_24h": 48056,
  },
  {
    "id": "cardano",
    "symbol": "ada",
    "name": "Cardano",
    "image": "https://assets.coingecko.com/coins/images/975/large/cardano.png?154703
    "current_price": 0.84514,
    "market_cap": 27210647217,
    "total_volume": 3204270671,
    "high_24h": 0.919055,
    "low_24h": 0.843236,
  },
  {
    "id": "ethereum",
    "symbol": "eth",
    "name": "Ethereum",
    "image": "https://assets.coingecko.com/coins/images/279/large/ethereum.png?15953
    "current_price": 1479.97,
    "market_cap": 172447578072,
    "total_volume": 24709055087,
    "high_24h": 1597.13,
    "low_24h": 1493,
  },
  {
    "id": "litecoin",
    "symbol": "ltc",
    "name": "Litecoin",
    "image": "https://assets.coingecko.com/coins/images/2/large/litecoin.png?1547033
    "current_price": 171.49,
    "market_cap": 11561005268,
    "total_volume": 4950077782,
    "high_24h": 187.34,
    "low_24h": 172.45,
  },
  {
    "id": "polkadot",
    "symbol": "dot",
    "name": "Polkadot",
    "image": "https://assets.coingecko.com/coins/images/12171/large/aJGBjJFU_400x400
    "current_price": 29.28,
    "market_cap": 28856989783,
    "total_volume": 1266769267,
    "high_24h": 32.2,
    "low_24h": 29.54,
  },
  {
    "id": "ripple",
    "symbol": "xrp",
    "name": "XRP",
    "image": "https://assets.coingecko.com/coins/images/44/large/xrp-symbol-white-12
    "current_price": 0.360658,
    "market_cap": 16580549437,
    "total_volume": 2357746464,
    "high_24h": 0.381072,
    "low_24h": 0.358941,
  },
  {
    "id": "tether",
    "symbol": "usdt",
    "name": "Tether",
    "image": "https://assets.coingecko.com/coins/images/325/large/Tether-logo.png?15
    "current_price": 0.83869,
    "market_cap": 32307660438,
    "total_volume": 82854947322,
    "high_24h": 0.843104,
    "low_24h": 0.832594,
  },
  {
    "id": "uniswap",
    "symbol": "uni",
    "name": "Uniswap",
    "image": "https://assets.coingecko.com/coins/images/12504/large/uniswap-uni.png?
    "current_price": 24.94,
    "market_cap": 13099199643,
    "total_volume": 939432128,
    "high_24h": 27.92,
    "low_24h": 24.78,
  }
]
```

https://docs.python.org/3/library/functions.html#sorted

```
sorted(iterable, *, key=None, reverse=False)
```

V dokumentaciji vidimo, da lahko kontroliramo katere vrednosti primerjamo z uporabo **key** parametra.

Kot **key** lahko podamo našo funkcijo, ki sprejme 1 argument in vrne vrednost po kateri primerjamo.

In [22]:

```
def sort_funkcija(x):
    print(f'{x["id"]}    \t {x["market_cap"]}')
    return x["market_cap"]

sorted(data, key=sort_funkcija, reverse=True)
```

```
binancecoin      33015186690
bitcoin          901453728232
cardano          27210647217
ethereum         172447578072
litecoin         11561005268
polkadot         28856989783
ripple           16580549437
tether           32307660438
uniswap          13099199643
```

Out[22]:

```
[{'id': 'bitcoin',
  'symbol': 'btc',
  'name': 'Bitcoin',
  'image': 'https://assets.coingecko.com/coins/images/1/large/bitcoin.png?154703357
9',
  'current_price': 47553,
  'market_cap': 901453728232,
  'total_volume': 47427138554,
  'high_24h': 51131,
  'low_24h': 48056},
 {'id': 'ethereum',
  'symbol': 'eth',
  'name': 'Ethereum',
  'image': 'https://assets.coingecko.com/coins/images/279/large/ethereum.png?1595348
880',
  'current_price': 1479.97,
  'market_cap': 172447578072,
  'total_volume': 24709055087,
  'high_24h': 1597.13,
  'low_24h': 1493},
 {'id': 'binancecoin',
  'symbol': 'bnb',
  'name': 'Binance Coin',
  'image': 'https://assets.coingecko.com/coins/images/825/large/binance-coin-logo.pn
g?1547034615',
  'current_price': 212.03,
  'market_cap': 33015186690,
  'total_volume': 2490184836,
  'high_24h': 230.59,
  'low_24h': 210.87},
 {'id': 'tether',
  'symbol': 'usdt',
  'name': 'Tether',
  'image': 'https://assets.coingecko.com/coins/images/325/large/Tether-logo.png?1598
003707',
  'current_price': 0.83869,
  'market_cap': 32307660438,
  'total_volume': 82854947322,
  'high_24h': 0.843104,
  'low_24h': 0.832594},
 {'id': 'polkadot',
  'symbol': 'dot',
  'name': 'Polkadot',
  'image': 'https://assets.coingecko.com/coins/images/12171/large/aJGBjJFU_400x400.j
pg?1597804776',
  'current_price': 29.28,
  'market_cap': 28856989783,
  'total_volume': 1266769267,
  'high_24h': 32.2,
  'low_24h': 29.54},
 {'id': 'cardano',
  'symbol': 'ada',
  'name': 'Cardano',
  'image': 'https://assets.coingecko.com/coins/images/975/large/cardano.png?15470348
60',
  'current_price': 0.84514,
  'market_cap': 27210647217,
  'total_volume': 3204270671,
  'high_24h': 0.919055,
  'low_24h': 0.843236},
 {'id': 'ripple',
  'symbol': 'xrp',
  'name': 'XRP',
  'image': 'https://assets.coingecko.com/coins/images/44/large/xrp-symbol-white-128.
png?1605778731',
  'current_price': 0.360658,
  'market_cap': 16580549437,
  'total_volume': 2357746464,
  'high_24h': 0.381072,
  'low_24h': 0.358941},
 {'id': 'uniswap',
  'symbol': 'uni',
  'name': 'Uniswap',
  'image': 'https://assets.coingecko.com/coins/images/12504/large/uniswap-uni.png?16
00306604',
  'current_price': 24.94,
  'market_cap': 13099199643,
  'total_volume': 939432128,
  'high_24h': 27.92,
  'low_24h': 24.78},
 {'id': 'litecoin',
  'symbol': 'ltc',
  'name': 'Litecoin',
  'image': 'https://assets.coingecko.com/coins/images/2/large/litecoin.png?154703358
0',
  'current_price': 171.49,
  'market_cap': 11561005268,
  'total_volume': 4950077782,
  'high_24h': 187.34,
  'low_24h': 172.45}]
```

Isto sortiranje lahko dobimo z uporabo lambda funkcije.

In [23]:
```python
sorted(data, key=lambda x: x["market_cap"], reverse=True)
```

Out[23]:
```
[{'id': 'bitcoin',
  'symbol': 'btc',
  'name': 'Bitcoin',
  'image': 'https://assets.coingecko.com/coins/images/1/large/bitcoin.png?154703357
9',
  'current_price': 47553,
  'market_cap': 901453728232,
  'total_volume': 47427138554,
  'high_24h': 51131,
  'low_24h': 48056},
 {'id': 'ethereum',
  'symbol': 'eth',
  'name': 'Ethereum',
  'image': 'https://assets.coingecko.com/coins/images/279/large/ethereum.png?1595348
880',
  'current_price': 1479.97,
  'market_cap': 172447578072,
  'total_volume': 24709055087,
  'high_24h': 1597.13,
  'low_24h': 1493},
 {'id': 'binancecoin',
  'symbol': 'bnb',
  'name': 'Binance Coin',
  'image': 'https://assets.coingecko.com/coins/images/825/large/binance-coin-logo.pn
g?1547034615',
  'current_price': 212.03,
  'market_cap': 33015186690,
  'total_volume': 2490184836,
  'high_24h': 230.59,
  'low_24h': 210.87},
 {'id': 'tether',
  'symbol': 'usdt',
  'name': 'Tether',
  'image': 'https://assets.coingecko.com/coins/images/325/large/Tether-logo.png?1598
003707',
  'current_price': 0.83869,
  'market_cap': 32307660438,
  'total_volume': 82854947322,
  'high_24h': 0.843104,
  'low_24h': 0.832594},
 {'id': 'polkadot',
  'symbol': 'dot',
  'name': 'Polkadot',
  'image': 'https://assets.coingecko.com/coins/images/12171/large/aJGBjJFU_400x400.j
pg?1597804776',
  'current_price': 29.28,
  'market_cap': 28856989783,
  'total_volume': 1266769267,
  'high_24h': 32.2,
  'low_24h': 29.54},
 {'id': 'cardano',
  'symbol': 'ada',
  'name': 'Cardano',
  'image': 'https://assets.coingecko.com/coins/images/975/large/cardano.png?15470348
60',
  'current_price': 0.84514,
  'market_cap': 27210647217,
  'total_volume': 3204270671,
  'high_24h': 0.919055,
  'low_24h': 0.843236},
 {'id': 'ripple',
  'symbol': 'xrp',
  'name': 'XRP',
  'image': 'https://assets.coingecko.com/coins/images/44/large/xrp-symbol-white-128.
png?1605778731',
  'current_price': 0.360658,
  'market_cap': 16580549437,
  'total_volume': 2357746464,
  'high_24h': 0.381072,
  'low_24h': 0.358941},
 {'id': 'uniswap',
  'symbol': 'uni',
  'name': 'Uniswap',
  'image': 'https://assets.coingecko.com/coins/images/12504/large/uniswap-uni.png?16
00306604',
  'current_price': 24.94,
  'market_cap': 13099199643,
  'total_volume': 939432128,
  'high_24h': 27.92,
  'low_24h': 24.78},
 {'id': 'litecoin',
  'symbol': 'ltc',
  'name': 'Litecoin',
  'image': 'https://assets.coingecko.com/coins/images/2/large/litecoin.png?154703358
0',
  'current_price': 171.49,
  'market_cap': 11561005268,
  'total_volume': 4950077782,
  'high_24h': 187.34,
  'low_24h': 172.45}]
```

Naloga:

Imamo podatke o GDP Evropskih držav od leta 2010 do 2020. Uporabite funkcijo **sorted()** in določite takšno **lambda funkcijo**, da razvrstimo države po GDP leta 2020 od največje do najmanjše. Izpišite imena držav od največje do najmanjše.

Primeri:

```
Input:
data = [["Austria", 392.623, 431.515, 409.652, 430.203, 442.698, 381.998,
394.215, 417.721, 456.166, 447.718, 432.894],
["Belgium", 484.450, 527.492,498.161, 521.090, 531.651, 456.067, 469.931,
495.953, 532.268, 517.609, 503.416],
["Bosnia", 17.164, 18.629, 17.207, 18.155, 18.522, 16.210, 16.910, 18.081,
20.162, 20.106, 18.893],
["Bulgaria", 50.611, 57.420, 53.901, 55.557, 56.815, 50.201, 53.236,
58.342, 65.197, 66.250, 67.917],
["Croatia", 59.866, 62.399, 56.549, 58.158, 57.683, 49.519, 51.623, 55.201,
60.805, 60.702, 56.768],
["Cyprus", 25.608, 27.454, 25.055, 24.094, 23.401, 19.691, 20.461, 22.189,
24.493, 24.280, 23.246],
["Czech Republic", 207.478, 227.948, 207.376, 209.402, 207.818, 186.830,
195.090, 215.914, 245.226, 246.953, 241.975],
["Denmark",321.995,344.003,327.149,343.584,
352.994,302.673,311.988,329.866,352.058,347.176,339.626],
["Estonia",19.536,23.191,23.057,25.145, 26.658,
22.916,23.994,26.850,30.761,31.038,30.468],
["Finland",248.262,273.925,256.849,270.065,273.042,232.582,239.150,252.867,274.
```

```
["France",2647.537,2864.030,2685.311,2811.957,2856.697,2439.435,2466.152,2591.7

["Germany",3423.466,3761.142,3545.946,3753.687,3904.921,3383.091,3496.606,3664.

["Greece",299.919,288.062,245.807,239.937,237.406,196.690,195.303,203.493,218.2

["Hungary",130.923,140.782,127.857,135.221,140.083,123.074,126.008,
139.844,161.182,170.407,149.939],
["Iceland",13.684,15.159,14.724,16.034, 17.758,17.389,20.618,24.457,25.965,
23.918,20.805],
["Ireland",222.533,238.088,
225.140,238.708,259.200,290.858,301.968,335.211,382.754,384.940,399.064],
["Italy",2129.021,2278.376,2073.971,2131.159,2155.151,1833.195,1869.973,1950.70

["Latvia",23.809,28.496,28.141,30.260,31.385,26.986,27.707,30.528,34.882,35.045

["Liechtenstein",5.082,5.740,5.456,6.392,6.657,6.268,6.215],
["Lithuania",37.200,43.564,
42.887,46.423,48.632,41.538,42.991,47.645,53.302,53.641,55.064],
["Luxembourg",53.312,60.060,56.709,61.759,66.209,57.233,58.985,62.449,69.553,69

["Malta",8.757,9.511,9.215,10.154,11.302,10.701,11.446,12.764,14.560,14.859,14.

["Montenegro",4.147,4.543,4.090,4.466,4.595,4.055,4.376,4.855,5.457,5.424,4.943

["Netherlands",848.133,904.915,839.436,
877.198,892.397,765.650,783.852,833.575,914.519,902.355,886.339],
["Norway",429.131,498.832,510.229,523.502,499.338,386.663,371.345,398.394,434.1

["Poland",479.161,528.571,500.846,524.399,545.284,477.568,471.843,526.749,585.8

["Portugal",238.748,245.119,216.488,226.144,229.995,199.521,206.361,221.280,240

["Romania",166.225,
183.443,171.196,190.948,199.628,177.895,188.495,211.407,239.552,243.698,248.624

["Serbia",41.369,49.280,43.300, 48.394,47.062,39.629,40.630,44.120,
50.509,51.523,51.999],
["Slovakia",89.668,
98.271,93.466,98.509,101.109,87.814,89.885,95.821,106.573,106.552,101.892],
["Slovenia",48.103,51.338,46.378,48.131,49.969,43.124,44.660,48.545,54.059,54.1

["Spain",1434.286,1489.431,1336.759,1362.280,1379.098,1199.688,1238.010,
1317.104,1427.533,1397.870,1247.464],
["Sweden",488.909,563.797,544.482,579.361,574.413,498.118,512.205,540.545,556.0

["Switzerland",583.053,
699.670,667.890,688.747,709.496,679.721,670.247,680.029,705.546,715.360,707.868

["Turkey",772.290,832.497,873.696,950.328,934.075,859.449,863.390,852.648,771.2

["United
Kingdom",2455.309,2635.799,2677.082,2755.356,3036.310,2897.060,2669.107,2640.06
```

Output:

```
Germany
United Kingdom
France
Italy
Spain
Netherlands
Switzerland
Turkey
Poland
Sweden
Belgium
Austria
Ireland
Norway
Denmark
Finland
Romania
Czech Republic
Portugal
Greece
Hungary
Slovakia
Luxembourg
Bulgaria
Croatia
Lithuania
Serbia
Slovenia
Latvia
Estonia
Cyprus
Iceland
Bosnia
Malta
Liechtenstein
Montenegro
```

In [96]:

```
data = [["Austria", 392.623, 431.515, 409.652, 430.203, 442.698, 381.998, 394.215, 4
["Belgium", 484.450, 527.492,498.161, 521.090, 531.651, 456.067, 469.931, 495.953, 5
["Bosnia", 17.164, 18.629, 17.207, 18.155, 18.522, 16.210, 16.910, 18.081, 20.162, 2
["Bulgaria", 50.611, 57.420, 53.901, 55.557, 56.815, 50.201, 53.236, 58.342, 65.197,
["Croatia", 59.866, 62.399, 56.549, 58.158, 57.683, 49.519, 51.623, 55.201, 60.805,
["Cyprus", 25.608, 27.454, 25.055, 24.094, 23.401, 19.691, 20.461, 22.189, 24.493, 2
["Czech Republic", 207.478, 227.948, 207.376, 209.402, 207.818, 186.830, 195.090, 21
["Denmark",321.995,344.003,327.149,343.584, 352.994,302.673,311.988,329.866,352.058,
["Estonia",19.536,23.191,23.057,25.145, 26.658, 22.916,23.994,26.850,30.761,31.038,3
["Finland",248.262,273.925,256.849,270.065,273.042,232.582,239.150,252.867,274.210,2
["France",2647.537,2864.030,2685.311,2811.957,2856.697,2439.435,2466.152,2591.775,27
["Germany",3423.466,3761.142,3545.946,3753.687,3904.921,3383.091,3496.606,3664.511,3
["Greece",299.919,288.062,245.807,239.937,237.406,196.690,195.303,203.493,218.230,21
["Hungary",130.923,140.782,127.857,135.221,140.083,123.074,126.008, 139.844,161.182,
["Iceland",13.684,15.159,14.724,16.034, 17.758,17.389,20.618,24.457,25.965, 23.918,2
["Ireland",222.533,238.088, 225.140,238.708,259.200,290.858,301.968,335.211,382.754,
["Italy",2129.021,2278.376,2073.971,2131.159,2155.151,1833.195,1869.973,1950.703,207
["Latvia",23.809,28.496,28.141,30.260,31.385,26.986,27.707,30.528,34.882,35.045,33.0
["Lithuania",37.200,43.564, 42.887,46.423,48.632,41.538,42.991,47.645,53.302,53.641,
["Luxembourg",53.312,60.060,56.709,61.759,66.209,57.233,58.985,62.449,69.553,69.453,
["Malta",8.757,9.511,9.215,10.154,11.302,10.701,11.446,12.764,14.560,14.859,14.290],
["Montenegro",4.147,4.543,4.090,4.466,4.595,4.055,4.376,4.855,5.457,5.424,4.943],
["Netherlands",848.133,904.915,839.436, 877.198,892.397,765.650,783.852,833.575,914.
["Norway",429.131,498.832,510.229,523.502,499.338,386.663,371.345,398.394,434.167,41
["Poland",479.161,528.571,500.846,524.399,545.284,477.568,471.843,526.749,585.816,56
["Portugal",238.748,245.119,216.488,226.144,229.995,199.521,206.361,221.280,240.901,
["Romania",166.225, 183.443,171.196,190.948,199.628,177.895,188.495,211.407,239.552,
["Serbia",41.369,49.280,43.300, 48.394,47.062,39.629,40.630,44.120, 50.509,51.523,51
["Slovakia",89.668, 98.271,93.466,98.509,101.109,87.814,89.885,95.821,106.573,106.55
["Slovenia",48.103,51.338,46.378,48.131,49.969,43.124,44.660,48.545,54.059,54.154,51
["Spain",1434.286,1489.431,1336.759,1362.280,1379.098,1199.688,1238.010, 1317.104,14
["Sweden",488.909,563.797,544.482,579.361,574.413,498.118,512.205,540.545,556.073,52
["Switzerland",583.053, 699.670,667.890,688.747,709.496,679.721,670.247,680.029,705.
["Turkey",772.290,832.497,873.696,950.328,934.075,859.449,863.390,852.648,771.274,74
["United Kingdom",2455.309,2635.799,2677.082,2755.356,3036.310,2897.060,2669.107,264
```

In [40]:
```python
data_sorted = sorted(data, key=lambda x: x[-1], reverse=True)
for i in data_sorted:
    print(i[0])
```

```
Germany
United Kingdom
France
Italy
Spain
Netherlands
Switzerland
Turkey
Poland
Sweden
Belgium
Austria
Ireland
Norway
Denmark
Finland
Romania
Czech Republic
Portugal
Greece
Hungary
Slovakia
Luxembourg
Bulgaria
Croatia
Lithuania
Serbia
Slovenia
Latvia
Estonia
Cyprus
Iceland
Bosnia
Malta
Montenegro
```

In [ ]:

Generators

Generatorji so funkcije namenjene generiranju iteratorjev (objekti, ki so lahko iterirani - list, itd..).

Razlika je, da generatorji generiranjo vrednosti eno po eno, ne vse naenkrat, kar jih nrdi veliko bolj memory-efficient.

Ustvarimo jih enako kot navadno funkcijo, le da namesto `return` uporbimo `yield` .

`yield` pavzira funkcijo in shrani njeno stanje, tako da lahko kasneje nadaljujemo kjer smo končali.

In [1]:
```
def moj_range(n):
    print("Start creating moj range")
    while n<10:
        yield n
        n += 1
    print("Stop generator")

val = moj_range(5)
print(val)
print(type(val))
```

```
<generator object moj_range at 0x0000026B4F8F8A50>
<class 'generator'>
```

Ko prvič pokličemo `next()` se program začne izvajati na začetku funkcije in nadaljuje do `yield` kjer vrne vrednost.

Naslednji klici `next()` nadaljujejo izvajanje programa od `yield` naprej do naslednjega `yield` .

Če ne naleti na `yield` dvigne StopIteration exception.

In [2]:
```
print(next(val))
```

```
Start creating moj range
5
```

In [3]:
```
next(val)
```

Out[3]:
```
6
```

In [4]:
```
next(val)
```

Out[4]:
```
7
```

In [5]:
```
#val = moj_range(5) # Če vmes ponovno kličemo generator bo šlo od začetka.
```

In [6]:
```
next(val)
```

Out[6]:
```
8
```

In [7]:
```
next(val)
```

Out[7]:
```
9
```

In [8]:
```
next(val)
```

```
Stop generator

---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-8-a2a2bf9708c5> in <module>
----> 1 next(val)

StopIteration:
```

Z generatorjem lahko ustvarimo svojo `range()` funkcionalnost.

In [9]:
```
def moj_range(n, m, step=1):
    while n<m:
        yield n
        n+=step

print("Primer: moj_range")
for i in moj_range(1, 20, 2):
    print(i)


print("Primer: range()")
for i in range(1, 20, 2):
    print(i)
```

```
Primer: moj_range
1
3
5
7
9
11
13
15
17
19
Primer: range()
1
3
5
7
9
11
13
15
17
19
```

In [ ]:

```
```

Comprehensions

Poleg generatorjev, lahko za kreiranje listov uporabimo tudi **list comprehensions**.

- List comprehensions so bolj berljivi od built-in funkcij, ki potrebujejo lambda expressions
- List comprehensions nam dovolijo filtriranje elementov

In [69]:

```
# Primer: želimo narediti list kvadratov iz lista a
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
squares = [x**2 for x in a]
print(a)
print(squares)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [70]:

```
# Primer: Filtriranje elementov
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

even_squares = [x**2 for x in a if x%2 == 0 and x%3==0]
print(a)
print(even_squares)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
[36, 144]
```

set comprehensions

In [85]:
```
a = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_squares = {x**2 for x in a if x % 2 == 0}
print(even_squares)
print(type(even_squares))
```

```
{64, 100, 4, 36, 16}
<class 'set'>
```

Dictionary Comprehensions

```
ict_variable = {key:value for (key,value) in dictonary.items()}
```

In [86]:
```
dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
# Double each value in the dictionary
double_dict1 = {k:v*2 for (k,v) in dict1.items()}
print(double_dict1)
```

```
{'a': 2, 'b': 4, 'c': 6, 'd': 8, 'e': 10}
```

Generator Expressions

Podobno kot list comprehensions lahko zapišemo tudi generatorje. Razlika je, da oni vrnejo generator objekt in ne list-e.

In [71]:
```
import sys

my_list = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

list_comprehension = [x for x in my_list]
set_comprehension = {x for x in my_list}
gen_expression = (x for x in my_list) # the language name for these is generator exp

print(list_comprehension)
print(type(list_comprehension))
print(sys.getsizeof(list_comprehension))
print()

print(set_comprehension)
print(type(set_comprehension))
print(sys.getsizeof(set_comprehension))
print()

print(gen_expression)
print(type(gen_expression))
print(sys.getsizeof(gen_expression))
for val in gen_expression:
    print(val)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
<class 'list'>
184

{'d', 'j', 'f', 'g', 'i', 'a', 'b', 'c', 'e', 'h'}
<class 'set'>
728

<generator object <genexpr> at 0x000001D5910FE900>
<class 'generator'>
112
a
b
c
d
e
f
g
```

```
h
i
j
```

Glavna razlika med `generator expressions` in `list comprehension` je, da so generatorji počasnejši ampak prišparajo na spominu.

Variable scope

Spremenljivke se razlikujejo tudi po tem koliko dolgo obstajajo (variable lifetime) in od kje lahko dostopamo do njih (variable scope).

Spremenljivka definirana znotraj funkcije (kot parameter ali navadno) obstaja samo znotraj funkcije.

Ko se izvajanje funkcije konča, spremenljivka neha obstajati.

In [72]:
```python
def funkcija(spr1):
    spr2 = 10
    print(f"Spr1: {spr1}")
    print(f"Spr2: {spr2}")


funkcija(5)
print(f"Spr1: {spr1}")
print(f"Spr2: {spr2}")
```

```
Spr1: 5
Spr2: 10

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-72-d9649ca9516e> in <module>
      6
      7 funkcija(5)
----> 8 print(f"Spr1: {spr1}")
      9 print(f"Spr2: {spr2}")

NameError: name 'spr1' is not defined
```

Spremenljivka definirana znotraj naše glavne kode (zunaj naših funkcij) je **globalna spremenljivka** in je dostopna skozi našo celotno kodo.

In [73]:
```python
spr1 = 5
print(f"Spr1: {spr1}")

if spr1 == 5:
    spr2 = 10
print(f"Spremenljivka2: {spr2}")
print()

def funkcija():
    spr3 = 200
    print(f"Spr1: {spr1}")
    print(f"Spr2: {spr2}")
    print(f"Spr3: {spr3}")

funkcija()
print()

print(f"Spr1: {spr1}")
print(f"Spr2: {spr2}")
```

```
Spr1: 5
```

```
Spremenljivka2: 10

Spr1: 5
Spr2: 10
Spr3: 200

Spr1: 5
Spr2: 10
```

Problem se lahko pojavi, če znotraj funkcije definiramo spremenljivko z enakim imenom, ki že obstaja kot globalna spremenljivka.

V tem primeru bo python spremenljivki označil kot dve različni spremenljivki. Ena dostopna znotraj funkcije, druga dostopna zunaj funkcije.

In [164…

```
spr1 = 5
print(f"Spr1: {spr1}")

def funkcija():
    spr1 = 100
    print(f"Spr1: {spr1}")

funkcija()
print(f"Spr1: {spr1}")
```

```
Spr1: 5
Spr1: 100
Spr1: 5
```

Parameter se obnaša kot lokalna spremenljivka.

In [175…

```
spr1 = 5
print(f"Spr1: {spr1}")

def funkcija(spr1):
    print(f"Spr1: {spr1}")

funkcija(100)
print(f"Spr1: {spr1}")
```

```
Spr1: 5
Spr1: 100
Spr1: 5
```

Paziti je potrebno, ko posredujemo list ali dictionary kot argument.

In [74]:

```
def funkcija(l):
    print(l)
    l[0] = 100

seznam = [3, 7, 13]
funkcija(seznam)
print(seznam)
```

```
[3, 7, 13]
[100, 7, 13]
```

In [75]:

```
def funkcija(d):
    print(d)
    d["a"] = 100

dict_ = {"a": 5, "b": 6, "c": 7}
funkcija(dict_)
print(dict_)
```

```
{'a': 5, 'b': 6, 'c': 7}
{'a': 100, 'b': 6, 'c': 7}
```

In [ ]:

Če želimo spreminjati globalno spremenljivko znotraj funkcije (znotraj local scope) moramo uporabiti besedo **global**.

In [76]:
```
spr1 = 5
print(f"Spr1: {spr1}")

def funkcija():
    global spr1
    spr1 = 100
    print(f"Spr1: {spr1}")

funkcija()
print(f"Spr1: {spr1}")
```

```
Spr1: 5
Spr1: 100
Spr1: 100
```

S to besedo lahko tudi ustvarimo novo globalno spremenljivko, znotraj localnega scopa.

In [77]:
```
def funkcija():
    global spr1
    spr1 = 5
    print(f"Spr1: {spr1}")

funkcija()
print(f"Spr1: {spr1}")
```

```
Spr1: 5
Spr1: 5
```

In [ ]:

Naloga:

Napišite funkcijo, kjer lahko igramo **vislice**. Funkcija **vislice()** naj ima 2 parametra. Prvi je besedo katero se ugiba in drugi število možnih ugibov. Če števila ugibov ne podamo naj bo default vrednost 10. Uporabnika konstantno sprašujte naj vnese črko. Nato izpišite iskano besedo. Črke katere je uporabnik uganil izpišite normalno, črke katere še ni uganil pa nadomestite z _. Dodatno zraven prikazujte katere vse črke je uporabnik že preizkusil. Če uporabnik besedo uspešno ugani v danih poizkusih naj funkcija vrne vrednost True. V nasprotnem primeru naj vrne vrednost False.

Primeri:

```
Input:
vislice("jabolko")

Output:
Guesses so far [].
What is your guess? a
_ a_ _ _ _ _

Guesses so far ['a'].
What is your guess? e
```

```
_ a_ _ _ _ _

Guesses so far ['a', 'e'].
What is your guess? o
_ a_ o_ _ o

Guesses so far ['a', 'e', 'o'].
What is your guess? p
_ a_ o_ _ o

Guesses so far ['a', 'e', 'o', 'p'].
What is your guess? r
_ a_ o_ _ o

Guesses so far ['a', 'e', 'o', 'p', 'r'].
What is your guess? l
_ a_ ol_ o

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l'].
What is your guess? k
_ a_ olko

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l', 'k'].
What is your guess? j
ja_ olko

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l', 'k', 'j'].
What is your guess? b
jabolko
KONEC
True
```

In [207…

```
# Rešitev
def vislice(beseda, n=10):
    correct_guesses = []
    all_guesses = []

    try_ = 0
    while try_ < n:
        print()
        guess = input(f"Guesses so far {all_guesses}. \nWhat is your guess? ")
        all_guesses.append(guess)
        if guess in beseda:
            correct_guesses.append(guess)

        beseda_print = ""
        for ch in beseda:
            if ch in correct_guesses:
                beseda_print += ch
            else:
                beseda_print += "_ "
        print(beseda_print)
        if len(set(correct_guesses)) == len(set(beseda)):
            print("KONEC")
            return True

        try_ += 1

    return False
```

```
print(vislice("jabolko"))
```

```
Guesses so far [].
What is your guess? a
_ a_ _ _ _ _ _

Guesses so far ['a'].
What is your guess? e
_ a_ _ _ _ _ _

Guesses so far ['a', 'e'].
What is your guess? o
_ a_ o_ _ _ o

Guesses so far ['a', 'e', 'o'].
What is your guess? p
_ a_ o_ _ _ o

Guesses so far ['a', 'e', 'o', 'p'].
What is your guess? r
_ a_ o_ _ _ o

Guesses so far ['a', 'e', 'o', 'p', 'r'].
What is your guess? l
_ a_ ol_ o

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l'].
What is your guess? k
_ a_ olko

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l', 'k'].
What is your guess? j
ja_ olko

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l', 'k', 'j'].
What is your guess? b
jabolko
KONEC
True
```

Naloga:

Ustvarite program **Križci in Krožci** Igralno polje lahko predstavite kot liste znotraj lista, kjer *E* predstavlja prazno polje. board = [["X", "E", "E"], ["O", "E", "E"], ["E", "E", "E"]] Od igralcev nato izmenično zahtevajte polje v katerega želijo postaviti svoj znak. Privzememo lahko, da bodo igralci igrali pravično in vpisovali samo prazna polja.

Primeri:

```
Output:
['E', 'E', 'E']
['E', 'E', 'E']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '00

['X', 'E', 'E']
['E', 'E', 'E']
['E', 'E', 'E']
It's O's turn. Make a move (exp: 12): '12
```

```
['X', 'E', 'E']
['E', 'E', 'O']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '10

['X', 'E', 'E']
['X', 'E', 'O']
['E', 'E', 'E']
It's O's turn. Make a move (exp: 12): '12

['X', 'E', 'E']
['X', 'E', 'O']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '20
X je ZMAGOVALEC!
```

In [92]:

```python
def display_board(board):
    for row in board:
        print(row)

def make_move(on_turn, board):
    move = input(f"It's {on_turn}'s turn. Make a move (exp: 12): '")
    row = int(move[0])
    col = int(move[1])
    board[row][col] = on_turn


def is_game_over(board):
    for row in board:
        if row[0] != "E":
            if row[0] == row[1] and row[0] == row[2]:
                return True

    for i in range(3):
        if board[0][i] != "E":
            if board[0][i] == board[1][i] and board[0][i] == board[2][i]:
                return True

    if board[0][0] != "E":
        if board[0][0] == board[1][1] and board[0][0] == board[2][2]:
            return True

    if board[0][2] != "E":
        if board[0][2] == board[1][1] and board[0][2] == board[2][0]:
            return True

    return False

def play():
    board = [["E", "E", "E"],
             ["E", "E", "E"],
             ["E", "E", "E"]]
    on_turn = "X"
    while True:
        display_board(board)
        make_move(on_turn, board)

        game_over = is_game_over(board)
        if game_over:
            print(f"{on_turn} je ZMAGOVALEC!")
            break
        else:
            if on_turn == "X":
                on_turn = "O"
            elif on_turn == "O":
                on_turn = "X"
        print()


play()
```

```
['E', 'E', 'E']
['E', 'E', 'E']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '00
Changing players

['X', 'E', 'E']
['E', 'E', 'E']
['E', 'E', 'E']
It's O's turn. Make a move (exp: 12): '12
Changing players

['X', 'E', 'E']
['E', 'E', 'O']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '10
Changing players

['X', 'E', 'E']
['X', 'E', 'O']
['E', 'E', 'E']
It's O's turn. Make a move (exp: 12): '12
Changing players

['X', 'E', 'E']
['X', 'E', 'O']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '20
X je ZMAGOVALEC!
```

In [ ]:

In [ ]:

In [ ]: