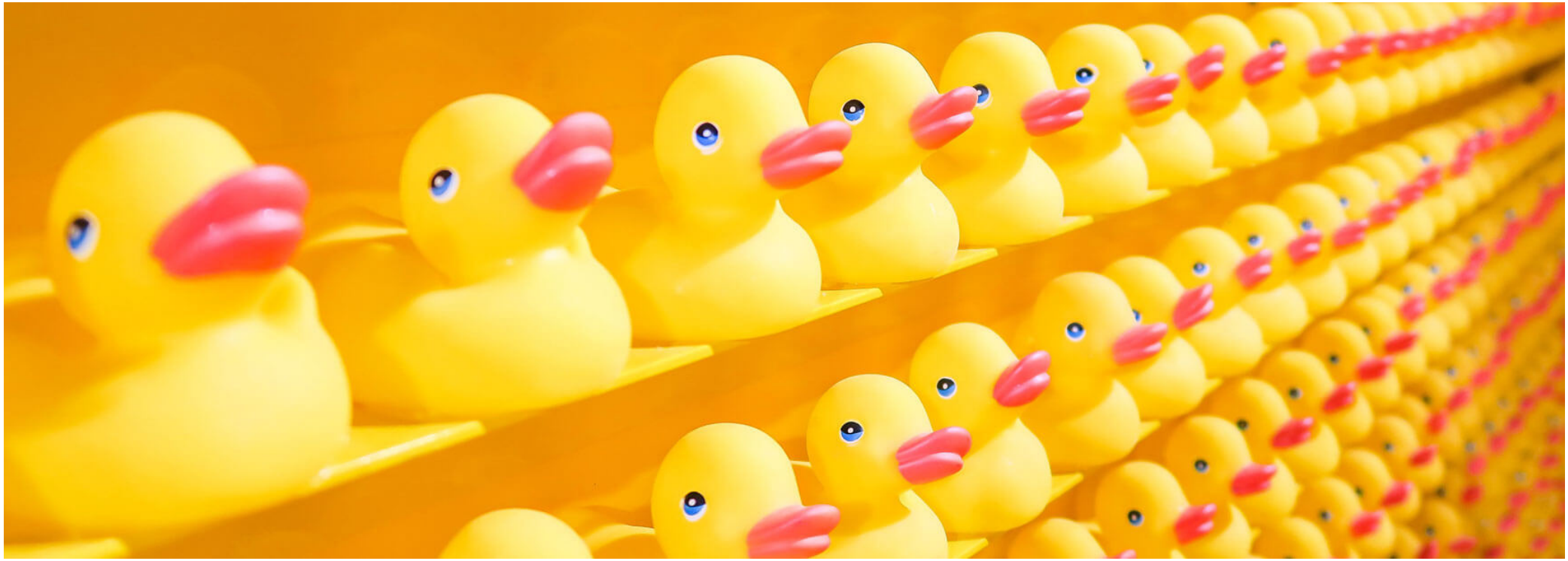
Iteration and Loops

Why iterate?



While-loops

```
while CONDITION:
    code to loop over
```

```
>>> i = 0
>>> while i < 5:
...     print(i)
...     i += 1
...
0
1
2
3
4
>>>
```

```
>>> i = 0
>>> while i < 5:
...     print(i)
...     i += 1
...     if i == 1:
...             print("STOP")
...             break
...
0
STOP
>>>
```

Exercise 4: Number Guessing

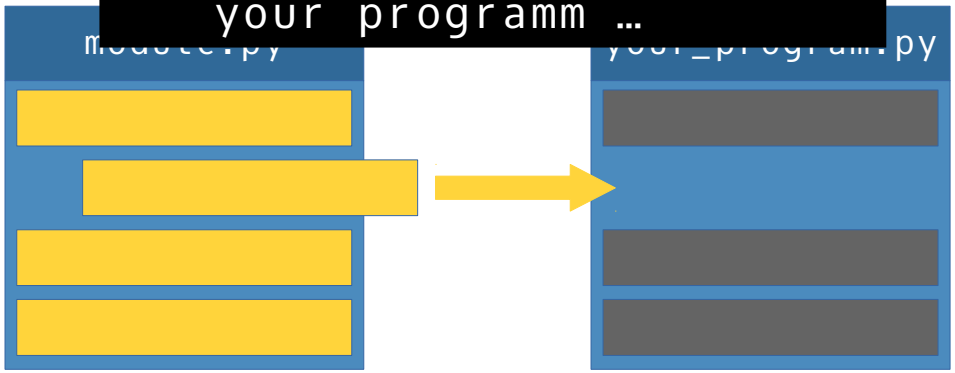
Write a program that makes the user guess an integer number.

The user should have multiple attemps on guessing the number.

Importing Modules

```
>>>
>>> import random
>>> random.randint(0, 5)
1
>>> from random import randint
>>> randint(0, 5)
1
>>> from random import *
>>> randint(0, 5)
2
>>> gauss(1, 3)
4.965213656566166
>>> import random as r
```

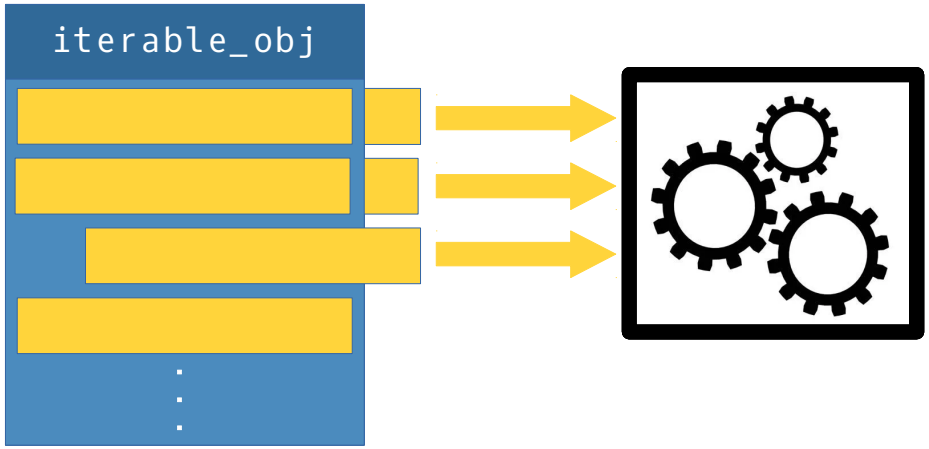
```
if __name__ == "__main__":
    your programm …
```




```
(conda-env) jonas@jonas-nb:~/Dokumente/pythonBootcamp2021/code$ conda install package-name
(conda-env) jonas@jonas-nb:~/Dokumente/pythonBootcamp2021/code$ conda install package-name=2.3.1
```

Iteration

for-loops
List & Generator Comprehension
Map & Filter
Aggregation



Iterables

Arrays, Dictionaries, Sets, Tupels
File Objects
Generators
…

for-loops

```
>>> array = [0, 1, 2, 3, 4, 5, 6]
>>> for number in array:
...     print(number)
...
0
1
2
3
4
5
6
>>>
```

```
>>> array = [0, 1, 2, 3, 4, 5, 6]
>>> for number in array:
...     print(number)
...     if number == 4:
...             break
...
0
1
2
3
4
>>>
```

```
>>> array1 = [1, 2, 3, 4]
>>> array2 = [5, 6, 7, 8]
>>> for i, j in zip(array1, array2):
...     print(f"i = {i}, j = {j}, i+j={i+j}")
...
i = 1, j = 5, i+j=6
i = 2, j = 6, i+j=8
i = 3, j = 7, i+j=10
i = 4, j = 8, i+j=12
>>> for index, value in enumerate(array1):
...     print(f"index = {index}, value = {value}")
...
index = 0, value = 1
index = 1, value = 2
index = 2, value = 3
index = 3, value = 4
>>>
```

```
>>> dict = {"A": 1, "B": 2, "C": 3}
>>> for key in dict:
...     print(f"{key}: {dict[key]}")
...
A: 1
B: 2
C: 3
>>> for key, value in dict.items():
...     print(f"{key}: {dict[key]}")
...
A: 1
B: 2
C: 3
>>> for value in dict.values():
...     print(value)
...
1
2
3
>>>
```

tqdm

```
>>> from tqdm import tqdm
>>> array = range(100)
>>> for i in tqdm(array):
...     pass
...
100%|████████████████████████████████████████████████████████████████████████| 100/100 [00:00<00:00, 927943.36it/s]
>>>
```

https://tqdm.github.io/

List Comprehension

```
new_list = []
for i in old_list:
    if filter(i):
        new_list.append(expressions(i))
```

```
[ expression for item in list if conditional ]
```

=

```
for item in list:
    if conditional:
        expression
```

```
>>> array = [0, 1, 2, 3, 4]
>>> [ x**2 for x in array ]
[0, 1, 4, 9, 16]
>>> [ x**2 for x in array if x % 2 == 0 ]
[0, 4, 16]
>>>
```

```
>>> matrix = [ [1, 2], [3, 4] ]
>>> [ number for array in matrix for number in array ]
[1, 2, 3, 4]
>>>
```

Dictionary Comprehension

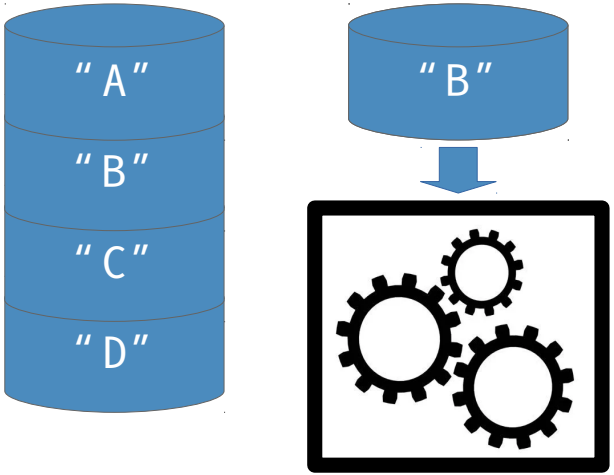
```
{ key:value for key, value in dictionary.items() }
{ key:value for key, value in zip(key_array, value_array) }
{ element:element for element in array }
```

=

```
new_dict1 = {}
for key, value in dictionary.items():
    new_dict1[key] = value
new_dict2 = {}
for key, value in zip(key_array, value_array):
    new_dict2[key] = value
new_dict2 = {}
for element in array:
    new_dict3[element] = element
```

Generator Comprehension

```
( expression for item in list if conditional )
```




```
>>> array = [0, 1, 2, 3, 4, 5]
>>> generator = ( x**2 for x in array )
>>> generator
<generator object <genexpr> at 0x7f8f748f82e0>
>>> generator[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'generator' object is not subscriptable
>>> for x in generator:
...     print(x)
...
0
1
4
9
16
25
```

Map & Filter

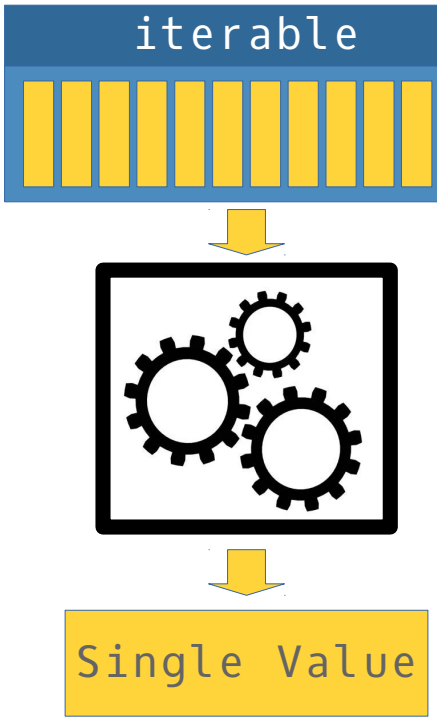
```
map(function, iterables)
```

```
>>> array = [0, 1, 2, 3, 4]
>>> def square(x):
...     return x**2
...
>>> mapped_array = map(square, array)
>>> mapped_array
<map object at 0x7f8f75cb9b80>
>>> list(mapped_array)
[0, 1, 4, 9, 16]
>>>
```

```
filter(function, iterables)
```

```
>>> array = [0, 4, 6, 7, 5, 3]
>>> def five_or_more(x):
...     return x >= 5
...
>>> filtered_array = filter(five_or_more, array)
>>> filtered_array
<filter object at 0x7f8f748e4f40>
>>> list(filtered_array)
[6, 7, 5]
>>>
```

Aggregation



Function	Return Value
len(iterable)	length of iterable
sum(iterable)	sum of all elements
min(iterable)	smallest value
max(iterable)	biggest value
all(iterable)	True if all elements True
any(iterable)	True if at least one element is True

Exercise 5: FizzBuzz

Write a programm that prints all integers between 0 and 100.
If the integer is divisible by 3 print “Fizz” instead.
If the integer is divisible by 5 print “Buzz” instead.
If the integer is both print “FizzBuzz” instead.
Feel free to add more phrases and divisors.