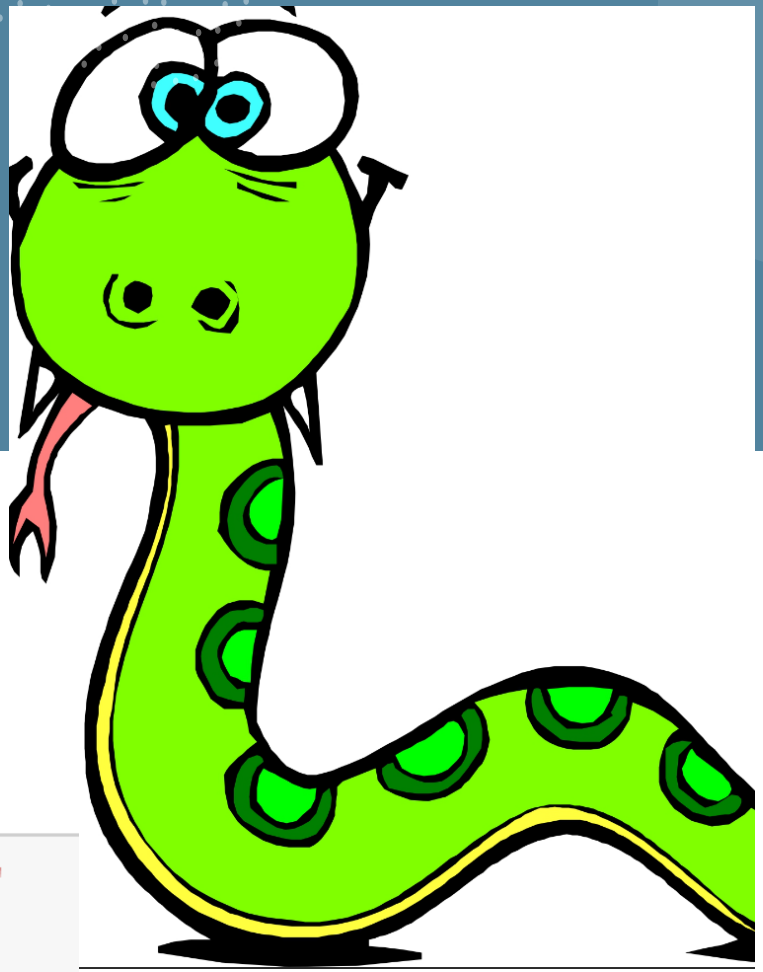
Welcome to my 5th Python Lecture



Lutz Plümer

```
WelcomeToMyLecture = "欢迎来到我的讲座"
print(WelcomeToMyLecture)
```

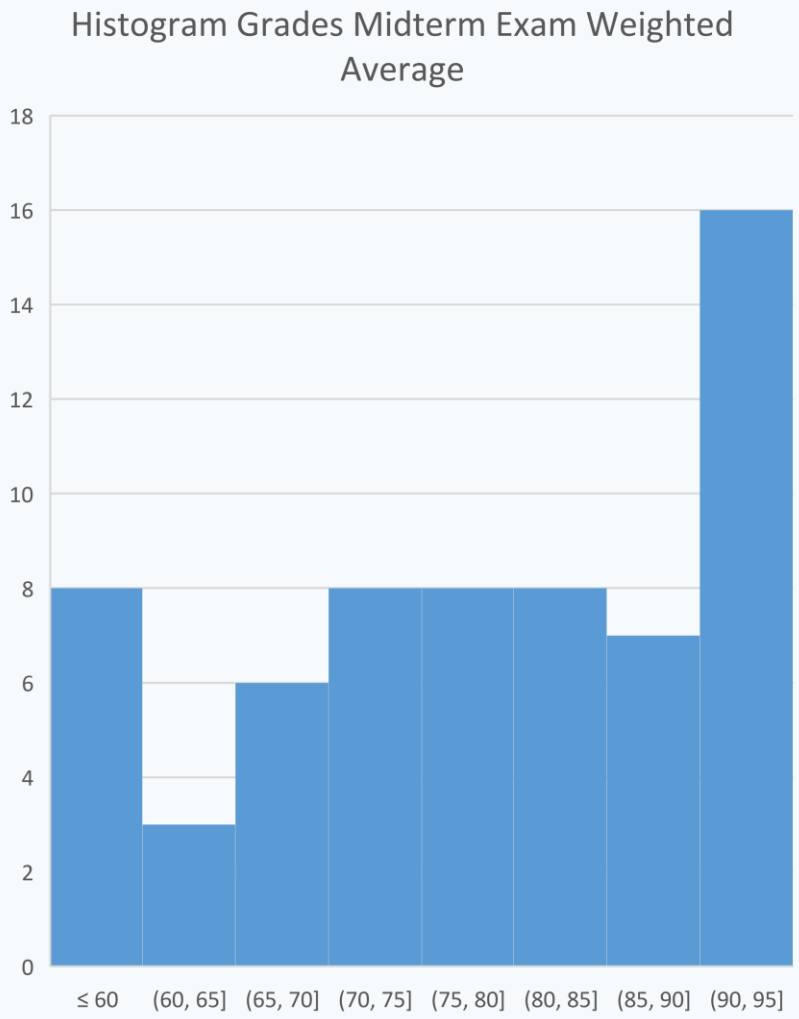
```
欢迎来到我的讲座
```

Some remarks on the Midterm Exam

Midterm Exam

- Assessment was based on a weighted average, where the first two exercises counted twice (which improved results considerably compared
  to normal average)
- 8 students < 60 (failed)
- 16 students > = 90 (A)
- 5 students >= 95 (A+)

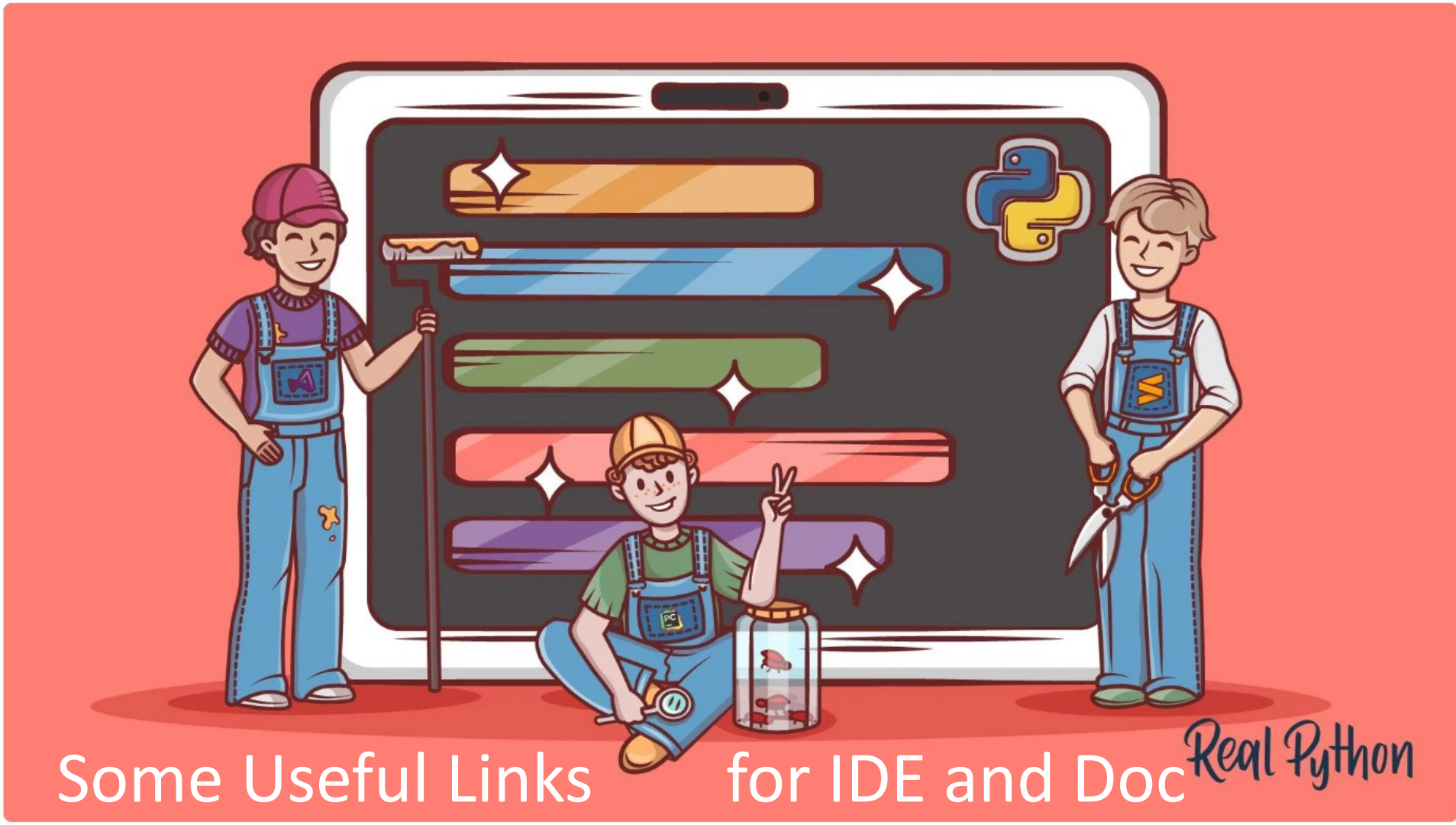
- I invite students who still have difficulties to a special constructive discussion Friday 11.00 a.m.
  how to improve
  (relaxed atmosphere, don't worry)



Good practice versus misconduct in exams

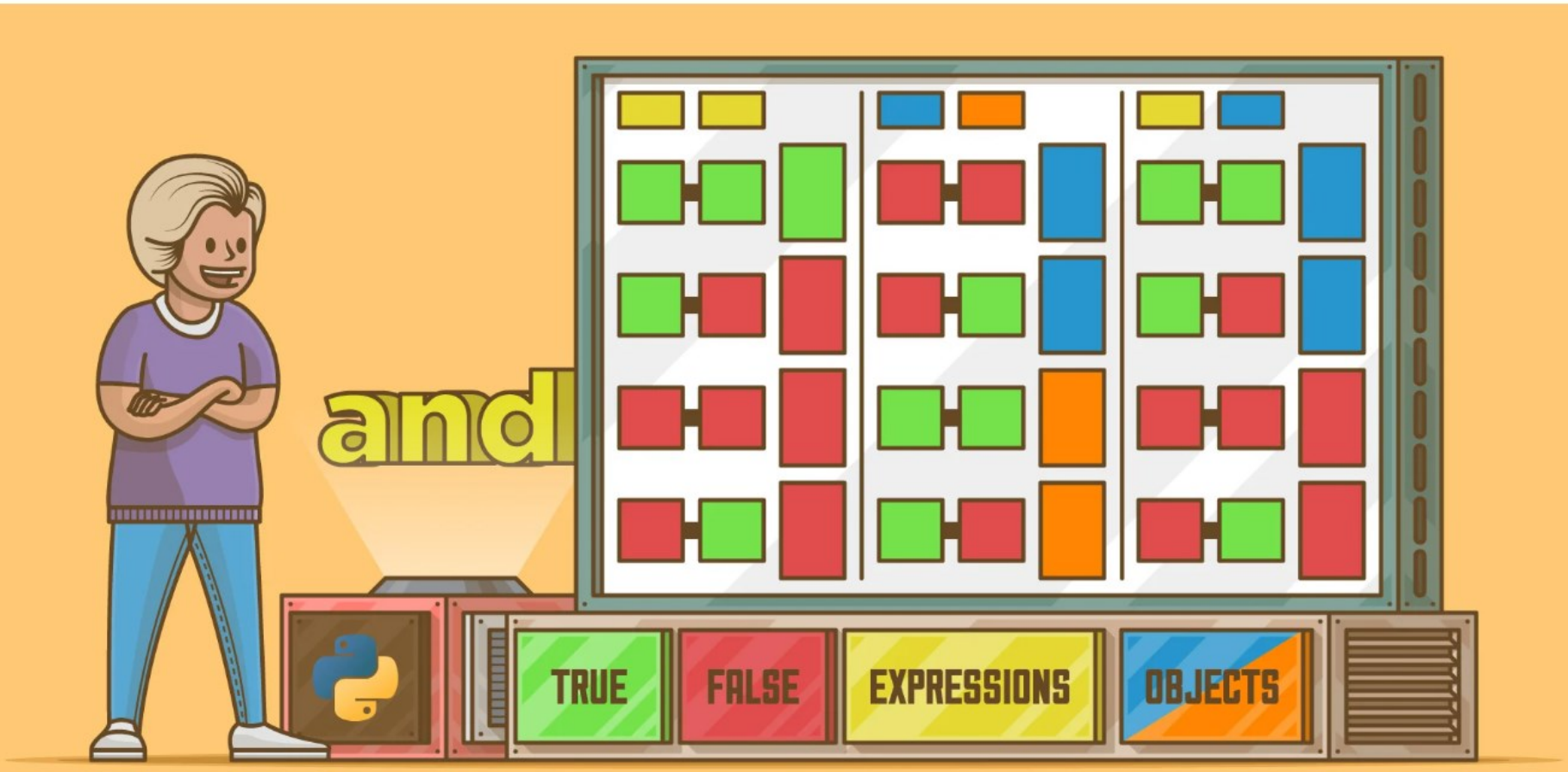
- my course is not aimed at dull learning aimed only at exams
- I am interested in teaching sustainable competence
- therefore we have strong interactive elements and computer use
- group work in class and homework
- computer use also in the exam, with individual performance
- this is susceptible to cheating I know
- there were some indications that I followed up on without finding solid evidence of misconduct
- However, we will learn from the Midterm Exam, make the rules more precise and improve the controls so that everyone can be sure that the exam is fair
- handwritten notes can be used in the Final Exam as well, but then 2 double pages (from your own hand, no copies) and nothing else:
  no slides, no old notebooks, no internet

- Those who actively participate in the lecture on a regular basis have a good chance of passing the exam
- On the other hand, those who prefer to play with their smartphones, play a passive role in group work and leave the exercises to others, have a low chance
- those who have particular difficulty should seek help from stronger members of their group
- and with the teaching assistants, we are always ready to help


Some Useful Links for IDE and Doc
Real Python

Some Useful Links

- There are more advanced Python IDEs (Integrated Development Environments), such as PyCharm:
  https://www.jetbrains.com/pycharm/
  http://y1.wxfeilian.cn/3/pycharm/
- Python Documentation:
  https://www.python.org/doc/
  https://docs.python.org/3/reference/index.html
- Tutorials:
  Real Python
  https://realpython.com/search?kind=article&kind=course&level=basics&order=newest
- Geeks for Geeks:
  https://www.geeksforgeeks.org/python-programming-language/
- This are suggestions for your Self Study



Some more useful operators in Python

Real Python

Operators

- You know already **+, -, *, /, //** (**integer** division), and **%** (**modulo**)
- You know, that you can use **+** to **concatenate** sequence such as lists or strings
- With **in** and **not in** you can test for membership of an element in a sequence

```
3 in [1, 2, 3, 4, 5]
```

```
True
```

```
7 in [1, 2, 3, 4, 5]
```

```
False
```

```
7 not in [1, 2, 3, 4, 5]
```

```
True
```

```
"ü" in "Plümer"
```

```
True
```

Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Examples with and, or, not and in

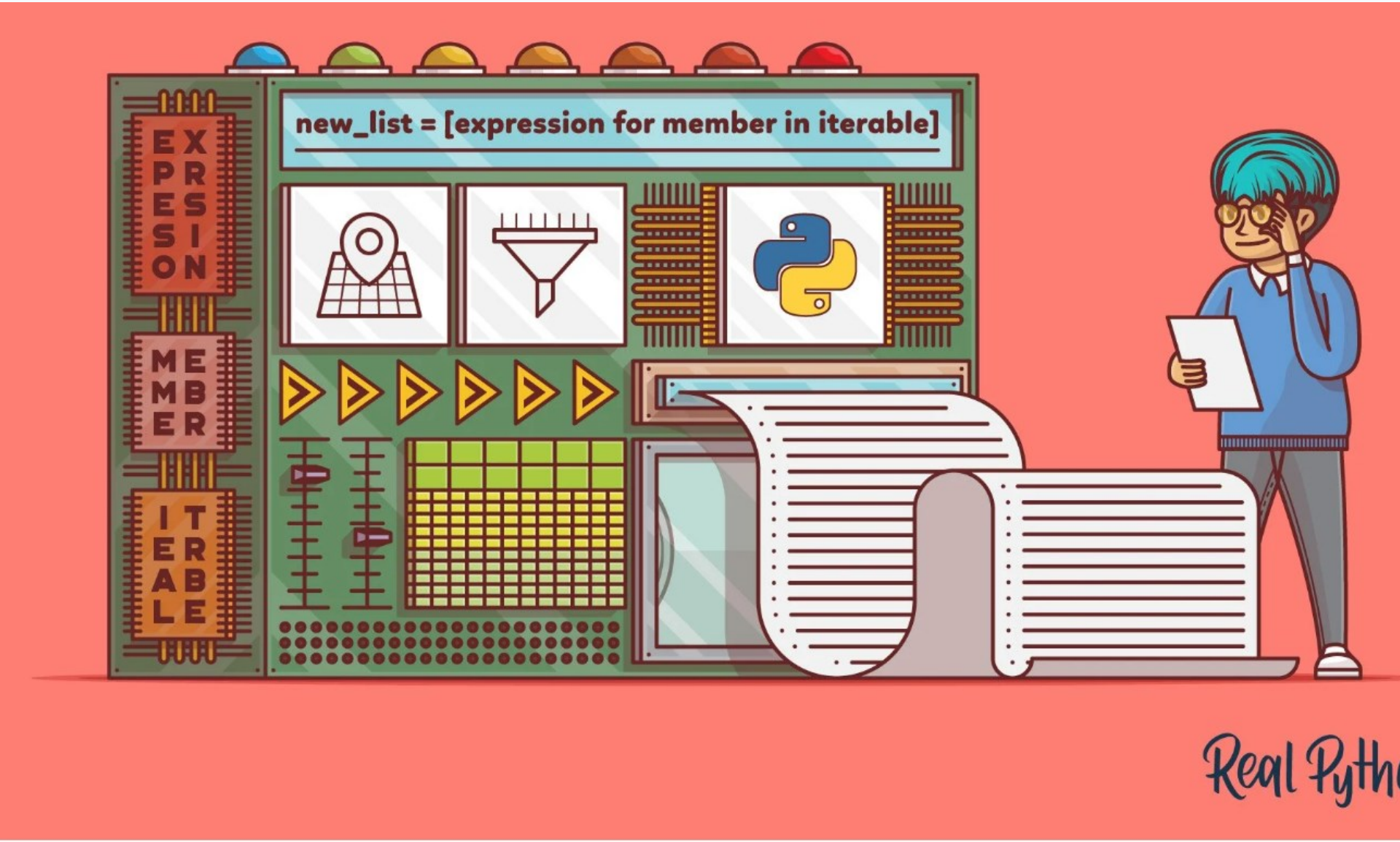
```
a  = 3
b = 5
a < b and a+1 < b
```

```
True
```

```
a < 4 or b < 4
```

```
True
```

```
"ü" in "Plümer" and "r" in "Plümer" and "c" not in "Plümer"
```


new_list = [expression for member in iterable]
EXPRESSION
MEMBER
ITERABLE
Real Pyth

List Comprehensions

- There is a very smart way to construct lists:
- Instead of

```
squares = []
for i in range(10):
    squares.append(i * i)
squares
```

```
]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- You can write

```
squares = [i * i for i in range(10)]
squares
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Syntax

- The syntax is:
  new_list = [expression **for** member **in** iterable]
- **Member** is the value, often i
- **Expression** is the member i or any value generated from i, such as i * i, i + i, …
- **Iterable** is any **sequence**, such as a list, a tuple or a string, or a **generator** such as **range**

List Comprehensions II

- You can even generate pairs:

```
squares2 = [(i, i * i) for i in range(10)]
squares2
```

```
[(0, 0),
 (1, 1),
 (2, 4),
 (3, 9),
 (4, 16),
 (5, 25),
 (6, 36),
 (7, 49),
 (8, 64),
 (9, 81)]
```

Add a condition

- new list = [expression **for** member **in** iterable **(if conditional)**]

```
sentence = 'the rocket came back from mars'
vowels = [i for i in sentence if i in 'aeiou']
vowels
```

```
['e', 'o', 'e', 'a', 'e', 'a', 'o', 'a']
```

```
"o" in sentence
```

```
True
```

```
"y" in sentence
```

```
False
```

- **in** is a test of membership in a sequence, such as a list or string

Square of even numbers

```
square_of_even = [i*i for i in range(10) if i%2==0]
square_of_even
```

```
[0, 4, 16, 36, 64]
```

Pairs of numbers with Condition

```
pairs = [(i,j) for i in range(5) for j in range(5)]
print(pairs)
```

```
[(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 0), (1, 1), (1, 2), (1, 3), (1,
0), (3, 1), (3, 2), (3, 3), (3, 4), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]
```

```
pairs = [(i,j) for i in range(5) for j in range(5) if (i + j) % 3 == 0]
print(pairs)
```

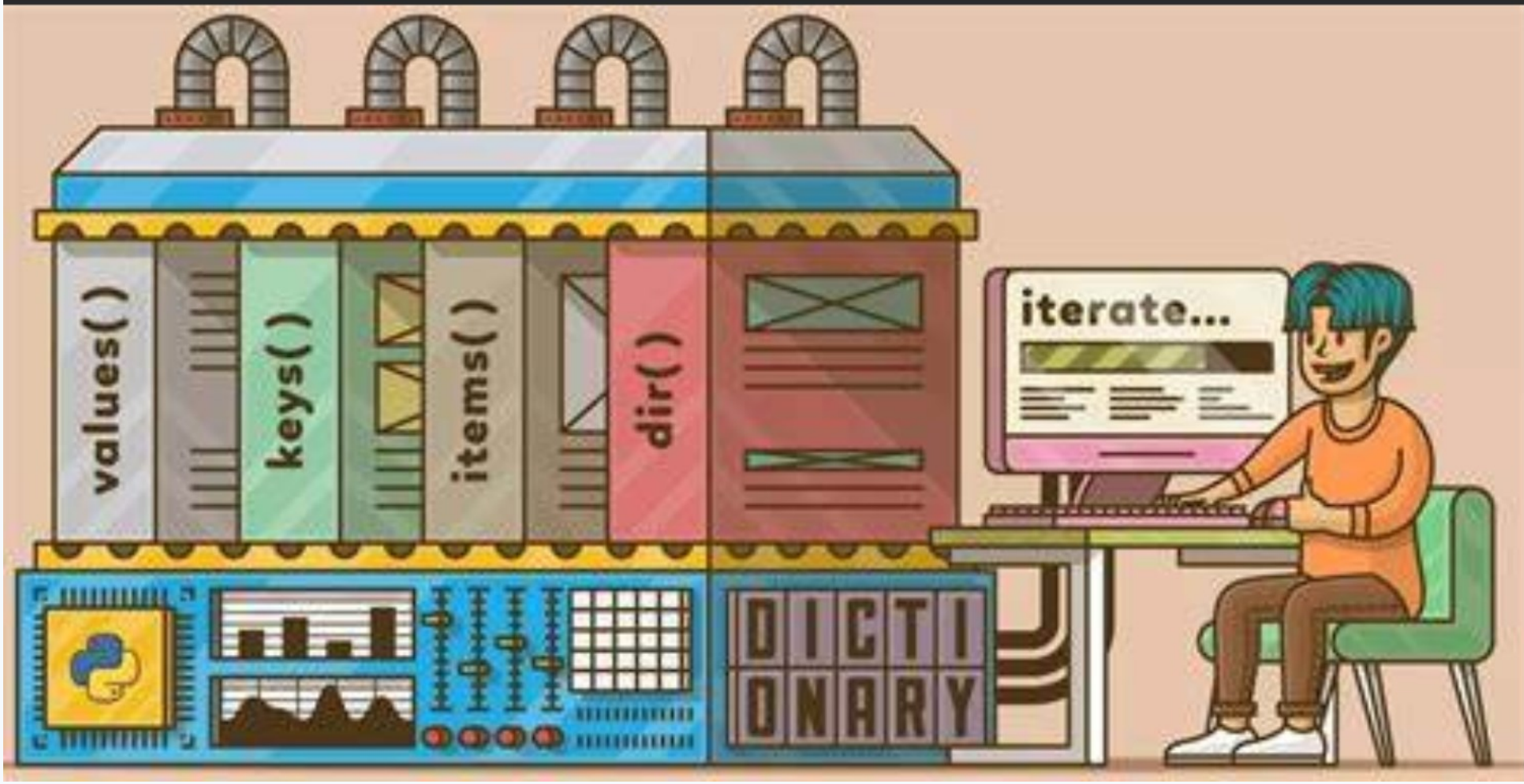
```
[(0, 0), (0, 3), (1, 2), (2, 1), (2, 4), (3, 0), (3, 3), (4, 2)]
```

```
pairs = [(i,j) for i in range(5) for j in range(5)]
print(pairs)
```

```
[(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 0), (1, 1), (1, 2), (1, 3), (1
0), (3, 1), (3, 2), (3, 3), (3, 4), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]
```

```
pairs = [(i,j) for i in range(5) for j in range(5) if (i + j) % 3 == 0]
print(pairs)
```

```
[(0, 0), (0, 3), (1, 2), (2, 1), (2, 4), (3, 0), (3, 3), (4, 2)]
```



Dictionaries

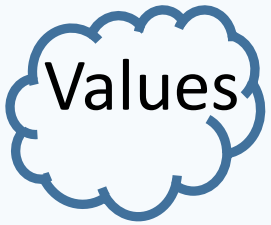
Topics for Today

- Today I will talk about Dictionaries, Classes and Objects
- start with Dictionaries
- Dictionaries are another kind of sequences, such as
- Text Strings: “this is a text string”
- Lists: [1, 2, 3]
- Tuples: (1, 2, 3)
- Dictionaries are formed by Key-Value-Pairs, such as

  dict = {'China': '中国', 'Singapore': '新加坡', 'Vietnam': '越南', 'Germany': '德国', 'Great Britain': '大不列颠', 'France': '法国', 'Spain': '西班牙', 'USA': '美国’}

- China, Singapore, ... are the Keys, and the Hanzi words 中国, … are the Values, in between there is a **:**

Dict as a Sequence of Key-Value Pairs




```
dict =
  {'China': '中国',
   'Singapore': '新加坡',
   'Vietnam': '越南',
   'Germany': '德国',
   'Great Britain': '大不列颠',
   'France': '法国',
   'Spain': '西班牙',
   'USA': '美国'}
```

Accessing Dictionary Values

- In a dictionary, you access a **Value** by its **Key** (never the other way round)
- Note: a **Key** can occur in a dictionary only **once**. A **Value** can occur **several** times
- A value is retrieved from a dictionary by specifying its corresponding key in **square brackets**

```
dict["China"]
```

```
'中国'
```

```
dict["Germany"]
```

```
'德国'
```

Adding a new Key Value Pair

- Adding an entry to an existing dictionary is simply a matter of assigning a new key and value:

```
dict["Ukraine"] = "乌克兰"
dict
```

```
{'China': '中国',
 'Singapore': '新加坡',
 'Vietnam': '越南',
 'Germany': '德国',
 'Great Britain': '大不列颠',
 'France': '法国',
 'Spain': '西班牙',
 'USA': '美国',
 'Ukraine': '乌克兰'}
```

Some more functions

- len()d gives the number of all key-value-pairs in a dictionary d
- k in d gives true if k is a key in d
- d.get(key, none) gives d[key] if key is in d, otherwise none (or any other value specified as second argument)
- d.keys is an iterator for all keys in d
- d.values is an iterator for all values in d
- d.items is an iterator for key-value pairs in d

```
d = dict
```

```
len(d)
```

```
9
```

```
d.get("France","none")
```

```
'法国'
```

```
d.get("Sweden","none")
```

```
'none'
```

Keys, Values and Items

```
d.keys()
```

```
dict_keys(['China', 'Singapore', 'Vietnam', 'Germany', 'Great Britain', 'France', 'Spain', 'USA', 'Ukraine'])
```

```
d.values()
```

```
dict_values(['中国', '新加坡', '越南', '德国', '大不列颠', '法国', '西班牙', '美国', '乌克兰'])
```

```
d.items()
```

```
dict_items([('China', '中国'), ('Singapore', '新加坡'), ('Vietnam', '越南'), ('Germany', '德国'), ('Great Britain', '大不列颠'), ('France', '法国'), ('Spain', '西班牙'), ('USA', '美国'), ('Ukraine', '乌克兰')])
```

```
d.keys()
```

```
dict_keys(['China', 'Singapore', 'Vietnam', 'Ge
```

```
d.values()
```

```
dict_values(['中国', '新加坡', '越南', '德国', '
```

```
d.items()
```

```
dict_items([('China', '中国'), ('Singapore', '新
颠'), ('France', '法国'), ('Spain', '西班牙'), (
```

For loops for keys(), values() and items()

```
for k in d.keys():
    print(k, end = ' ')
```

```
China Singapore Vietnam Germany Great Britain France Spain USA Ukraine
```

```
for v in d.values():
    print(v, end = ' ')
```

```
中国 新加坡 越南 德国 大不列颠 法国 西班牙 美国 乌克兰
```

```
for item in d.items():
    print(item, end = ' ')
```

```
('China', '中国') ('Singapore', '新加坡') ('Vietnam', '越南') ('Germany'
国') ('Spain', '西班牙') ('USA', '美国') ('Ukraine', '乌克兰')
```

Dictionary Comprehensions

```
values = [v for v in d.values()]
values
```

```
['中国', '新加坡', '越南', '德国', '大不列颠', '法国', '西班牙', '美国', '
```

```
keys = [k for k in d.keys()]
print(keys, end = ' ')
```

```
['China', 'Singapore', 'Vietnam', 'Germany', 'Great Britain', 'France',
```

```
items = [item for item in d.items()]
print(items, end = ' ')
```

```
[('China', '中国'), ('Singapore', '新加坡'), ('Vietnam', '越南'), ('Germ
e', '法国'), ('Spain', '西班牙'), ('USA', '美国'), ('Ukraine', '乌克兰')
```

Swapping a Dictionary

- In a Dictionary, you access values by keys, not the other way round
- In order to do it the other way, you need to swap the list, such as here by dictionary comprehension

```
swapped_dict = { value : key for (key, value ) in d.items() }
swapped_dict
```

```
{'中国': 'China',
 '新加坡': 'Singapore',
 '越南': 'Vietnam',
 '德国': 'Germany',
 '大不列颠': 'Great Britain',
 '法国': 'France',
 '西班牙': 'Spain',
 '美国': 'USA'}
```

```
swapped_dict['中国']
```

```
'China'
```

Classroom Exercises for Today

- No Homework until Thursday, next homework on Thursday
- Write the Programming Examples and modify
- Write a test if a Chinese character is in your Surname or Family Name
- Write a List Comprehension for Numbers Smaller than 160 which divide by 3 and 5
- Write a Dictionary for 5 Chinese Cities in English and Chinese Characters, taking the countries list as model
- Add one more city
- Write two sequences, one for the Chinese names and the English names, and construct the dictionary from these two sequences. Have a look at slide 26 and 14 to see how to do it