Python programming basics

Robert Haase

December 2022

@TillKorten

Working with variables

- Variables can hold numeric values and you can do math with them

```
# initialize program
a = 5
b = 3

# run algorithm on given parameters
sum = a + b

# print out result
print (sum)
```

```
8
```

Mathematical operations

- Math commands supplement operators to be able to implement any form of calculations

- Power

```
In [ ]: pow(3, 2)
Out[ ]: 9
```

- Absolute

```
In [ ]: abs(-8)
Out[ ]: 8
```

- Rounding

```
In [ ]: round(4.6)
Out[ ]: 5
```

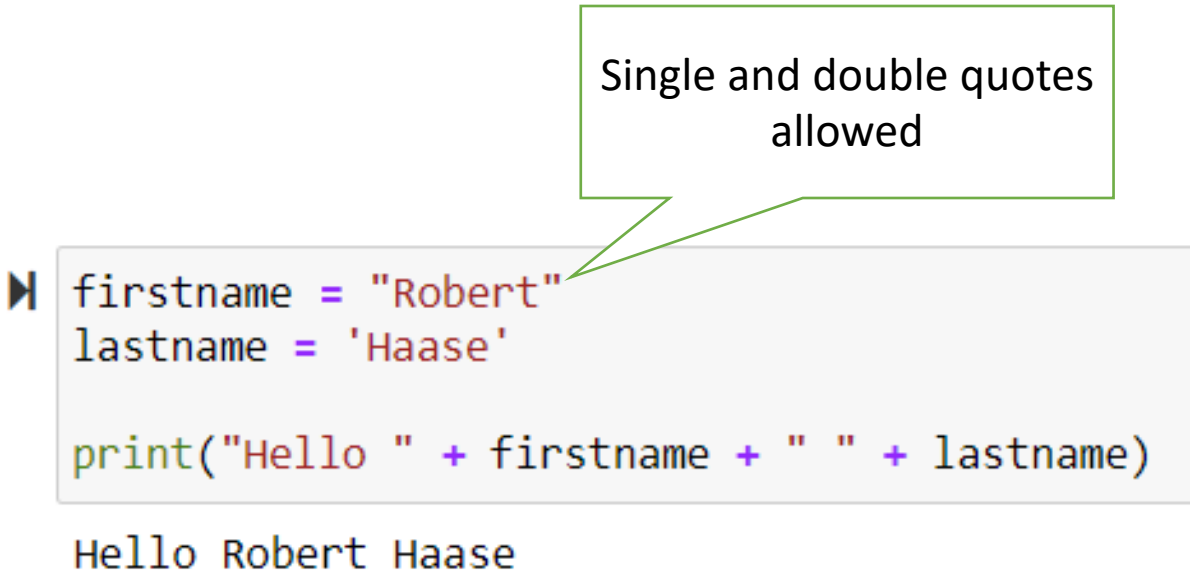
Be careful with some of them!

```
In [ ]: round(4.5)
Out[ ]: 4
```

https://en.wikipedia.org/wiki/Rounding#Round_half_to_even

Working with variables and string values

- Also strings as values for variables are supported



f-Strings

- String **f**ormatting is made easy using f-strings.

```
f"This is an f-string. a's value is {a}. Doubling the value of a gives {2*a}."
```

```
"This is an f-string. a's value is 5. Doubling the value of a gives 10."
```

- Using f-strings, you can also call code from within a string. Take care of code readability!

```
f"The first_name variable contains {first_name.lower().count('r')} r letters."
```

```
'The first_name variable contains 2 r letters.'
```

Comments

Comments should contain additional information such as

- User documentation
  - What does the program do?
  - How can this program be used?
- Your name / institute in case a reader has a question
- Comment why things are done.
- Do not comment what is written in the code already!

```
#
# This program sums up two numbers.
#
# Usage:
# * Run it in Python 3.8
#
# Author: Robert Haase, PoL TUD
#         Robert.haase@tu-dresden.de
# April 2021

# initialise program
a = 1
b = 2.5

# run complicated algorithm
final_result = a + b

# print the final result
print( final_result )
```

Live demo: Adapting and documenting a workflow notebook

Handling many items

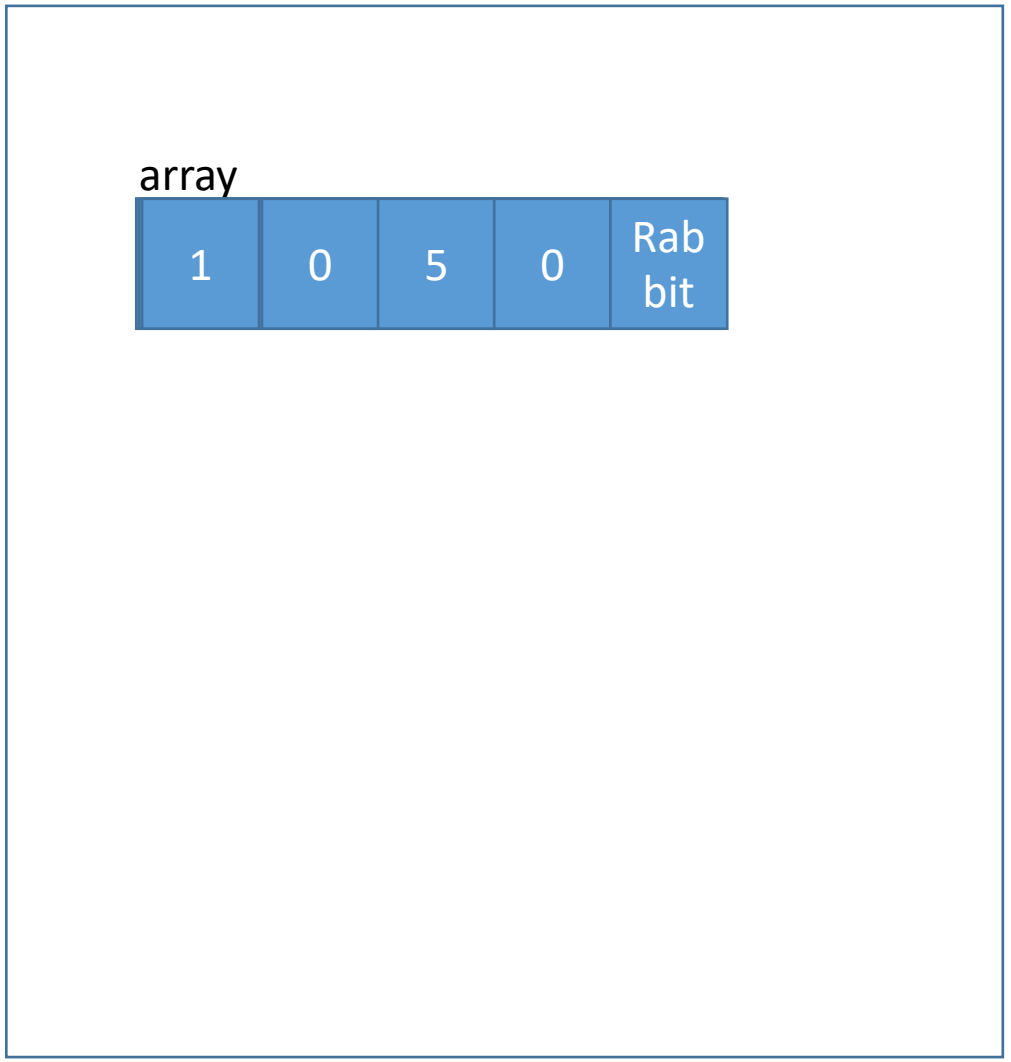
Lists

- Lists are variables, where you can store multiple values

Give me a “0”, five times!

```
array = [0] * 5
```

Computer memory



Arrays in Python

- Modifying lists entries

```
numbers = [0, 1, 2, 3, 4]

# write in one array element
numbers[1] = 5

print(numbers)
```

```
[0, 5, 2, 3, 4]
```

Note: The first element has index 0!

- Creating lists of defined size

What?

How many?

```
zeros = [0] * 10
print(zeros)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- Concatenating lists

```
ones = [1, 1, 1]
twos = [2, 2, 2, 2]

# concatenate arrays
numbers = ones + twos

print(numbers)
```

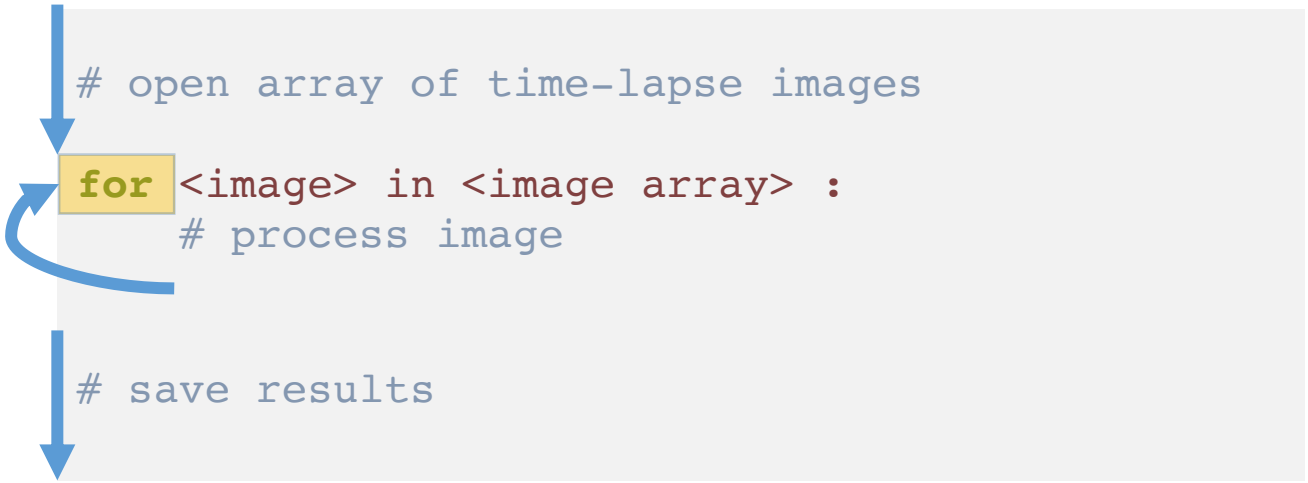
```
[1, 1, 1, 2, 2, 2, 2]
```

+ means appending

`for`: execute some lines of code *for* a number of times

PoL
Physics of Life
TU Dresden

- typically for all items in an array-like thing (lists, tuples, images)



`for-in`: Loop over items of a list

- Example list :

```
animal_set = ["Cat", "Dog", "Mouse"]

for animal in animal_set:
    print(animal)
```

```
Cat
Dog
Mouse
```

`range` creates numbers on the fly:

`range(start, stop, step)`

```
# for loops
for i in range(0, 5):
    print(i)
```

```
0
1
2
3
4
```

`for`-loop syntax pitfalls

- Indent the code within the `for` loop
  remember: indentation *means* combining operations to a block

```
# for loops
for i in range(0, 5):
print(i)
```

```
  File "<ipython-input-15-59c457ae0ac9>", line 3
    print(i)
        ^
IndentationError: expected an indented block
```

Don't forget to indent!

- Colon necessary

```
# for loops
for i in range(0, 5)
    print(i)
```

```
  File "<ipython-input-13-23157c0ed137>", line 2
    for i in range(0, 5)
                       ^
SyntaxError: invalid syntax
```

Don't forget the colon!