UNIVERSITÄT BIELEFELD
Faculty of Technology

Programming

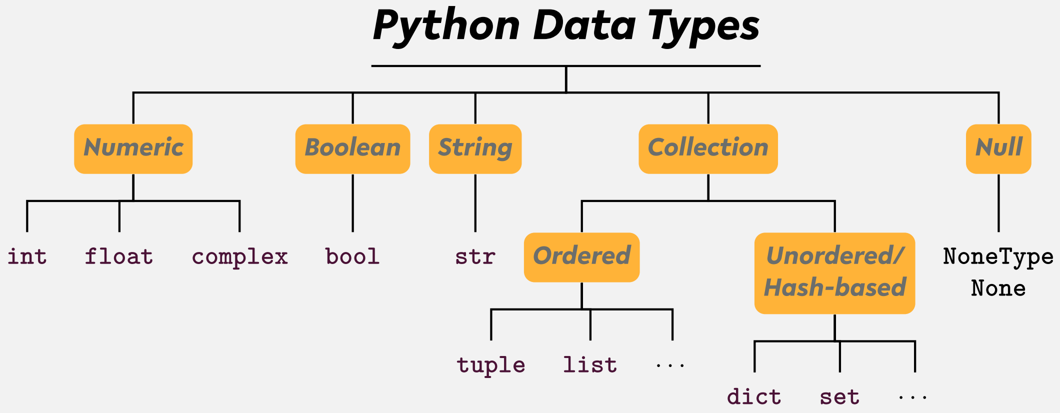
Loops

Luna Pianesi

Faculty of Technology, Bielefeld University

UNIVERSITÄT
BIELEFELD

Recap



... and user-defined types

UNIVERSITÄT BIELEFELD

Conditional statements: `if/else` *clause*

```
if «Boolean expression»:
␣␣␣␣«statement»
```

⚠ Mind the indentation!

OR

```
if «Boolean expression»:
␣␣␣␣«statement»
else:
␣␣␣␣«alternative statement»
```

Boolean operators, Comparisons

- Elementary logic: `and`, `or`, `not`
- Comparators:

`==`	“is equal/equivalent to”
`!=`	“is not equal/equivalent to”
`>`	“is larger than”
`<`	“is is smaller than”
`>=`	“is larger or equal to”
`<=`	“is smaller or equal to”
`is`	“is identical instance of”
`is not`	“is not identical instance of”
`in`	“is contained in collection”
`not in`	“is not contained in collection”

UNIVERSITÄT BIELEFELD

Loops

What are loops?

- Loops are the ability of programming languages to execute something again and again
- They are a ***control flow statement***
- They allow us to execute a group of instructions as long as the initial condition remains satisfied
- Two *keywords*: ***for*** and ***while***

For loops

While loops

UNIVERSITÄT BIELEFELD

`for`-*Loop*

```
for «control variable name» in «iterable»:
␣␣␣␣«statement»                          ⚠ Mind the indentation!
```

for-*Loop: Iteration over ordered collections*

Loop over elements

```
# tuple filled with arbitrary elements
my_tuple = (1, 2.0, 'text', list(), dict())

# for-loop over my_tuple with control
    variable 'el'
for el in my_tuple:
    msg = 'element:␣{}'.format(el)
    print(msg)
```

`for`-Loop: Iteration over ordered collections

Loop over indices with `range`

```python
# tuple filled with arbitrary elements
my_tuple = (1, 2.0, 'text', list(), dict())

# for-loop over my_tuple with control
    variable 'i'
for i in range(len(my_tuple)):
    el = my_tuple[i]
    msg = 'element␣{}:␣{}'.format(i, el)
    print(msg)
```

`for`-*Loop: Iteration over ordered collections*

Update `list` in for-loop

```python
# list filled with arbitrary elements
my_list = [1, 2.0, 'text', list(), dict()]

# for-loop over my_list with control
   variable 'i'
for i in range(len(my_list)):
    # update element with index i
    my_list[i] = 'element␣{}:␣{}'.format(i,
        my_list[i])
    print(my_list[i])
```

for-*Loop: Iteration over ordered collections*

Loop over indices and elements with enumerate

```python
# list filled with arbitrary elements
my_list = [1, 2.0, 'text', list(), dict()]

# for-loop over my_list with control
   variables 'i' and 'el'
for i, el in enumerate(my_list):
    # update element with index i
    my_list[i] = 'element␣{}:␣{}'.format(i,
       el)
    print('old:␣{},␣new:␣{}'.format(el,
       my_list[i]))
```

UNIVERSITÄT BIELEFELD

`for`-*Loop: Iteration over unordered collections*

Loop over elements of a `set`

```python
# set filled with arbitrary elements
my_set = {1, 1, 1, 2.0, 'text'}

# for-loop over my_set with control variable
    'el'
for el in my_set:
    msg = 'element:␣{}'.format(el)
    print(msg)
```

for-*Loop: Iteration over unordered collections*

Loop over keys of a `dict`

```python
# dictionary filled with arbitrary elements
my_dict = {'key': 'value', 1: 'text', (1, 2): 'text'}

# for-loop over keys of my_dict with control variable 'key'
for key in my_dict:
    val = my_dict[key]
    msg = 'key: {}, value: {}'.format(key, val)
    print(msg)
```

for-*Loop: Iteration over unordered collections*

Loop over items of a `dict`

```
# dictionary filled with arbitrary elements
my_dict = {'key': 'value', 1: 'text', (1, 2): 'text'}

# for-loop over items of my_dict with control variables 'key', 'val'
for key, val in my_dict.items():
    msg = 'key:␣{},␣value:␣{}'.format(key, val)
    print(msg)
```

UNIVERSITÄT BIELEFELD

For loops

While loops

UNIVERSITÄT BIELEFELD

Conditional iteration

Another type of loop in Python: `while`

- Loops until condition becomes False

```python
x = 5
while x > 0:
    print(x)
    x -= 1 # shorthand for x = x - 1
```

Special keywords in loops:

- `continue`: aborts current iteration and continues with the next
- `break`: aborts loop completely

UNIVERSITÄT
BIELEFELD

Quiz

- What does the instruction `tuple(range(3))` return?

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

- Let x be any integer, how many times is the `print` statement in the follwing `for`-loop executed?

```
for i in range(x):
    for j in range(i):
        print((i, j))
```

Quiz

- What does the instruction `tuple(range(3))` return?

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

- Let x be any integer, how many times is the `print` statement in the follwing `for`-loop executed?

```
for i in range(x):
    for j in range(i):
        print((i, j))
```

$\binom{x}{2}$ times

UNIVERSITÄT
BIELEFELD

Recap

Summary

- `for` and `while`