Introduction to Programming with Python

Dr. Anatol Wegner

Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg
Würzburg, Germany

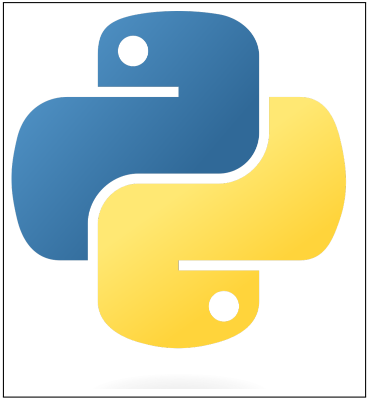
`anatol.wegner@uni-wuerzburg.de`

Lecture 02
Basic data types and control structures

November 8, 2024

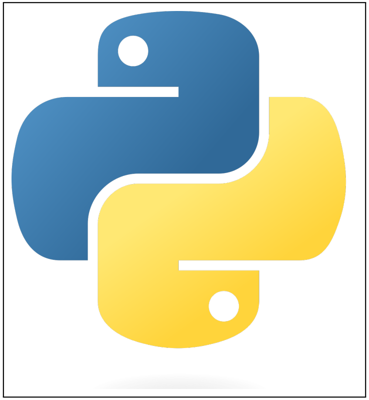
Recap

- Basics of programming
- Compiled vs interpreted programming languages
- Overview of the Python programming language
- Setting up Python and IDE
- We wrote our first program in Python
- We learned about **basic variable types and operations**



Recap

- Basics of programming
- Compiled vs interpreted programming languages
- Overview of the Python programming language
- Setting up Python and IDE
- We wrote our first program in Python
- We learned about **basic variable types and operations**



Today

- Basic data types and data structures
- Control structures

Variables and basic types

- Variables are containers for data
  - numbers, text, lists, dictionaries, files...
- In Python variables are created at assignment:
  - x = 5, y = 3.14, a = ’CS for Jurists’
- Python is dynamically typed:
  - Python assigns types to variables depending on their current value
  - Types of variables can change over time: a=1, a=’John’
  - need to keep track of variable types.

Basic types

- int() : 0, 1, -2, 1982 etc.
- float(): 3.14, -4.62131 etc.
- str(): ’Apples’, ’John’ etc.
- bool(): True, False.

Type conversion

- int(3.14) = 3
- float(3) = 3.0
- str(-4.45) = ’-4.45’
- bool(3.14) = True
- bool(0.0) = False
- int(’apple’)=?

Basic mathematical operations

- Arithmetic operations (+, -, *, /, **, %):
  - x**y (x^y): 3**2 = 9
  - % (mod) : 3 % 2 = 1
- Comparisons :
  - 3==4 (False)
  - 3!=4 (True)
  - 3<4 (True)
  - 3>4 (False)
  - 3<=4 (True)
  - 3>=4 (True)
- Logical operations (and, or, not):
  - False and True (False)
  - False or True (True)
  - False or not True (False)

Warning 1

Not all operations work with all types

Warning 2

Output of operations depend on types

- ’Alan’+’Turing’=’AlanTuring’
- 3 + True = 4
- True + True = 2
- ’Alan’+3=? (error)

Arrays

- Arrays are lists that can hold multiple values:
  - a=['John', 'Steven', 'Mark']
  - b=[3.5, True, 'Steven', -23]
- Elements of arrays can be accessed using their index:
  - indices start at '0'
  - a[0]='John',a[1]='Steven'
  - negative indices: a[-1]='Mark'
  - slicing: a[:2]=['John','Steven']

Arrays

- Arrays are lists that can hold multiple values:
  - a=['John', 'Steven', 'Mark']
  - b=[3.5, True, 'Steven', -23]
- Elements of arrays can be accessed using their index:
  - indices start at '0'
  - a[0]='John',a[1]='Steven'
  - negative indices: a[-1]='Mark'
  - slicing: a[:2]=['John','Steven']
- Adding and removing elements:
  - a.append['Alice'] (a=['John', 'Steven', 'Mark','Alice'])
  - a.remove['Mark'] (a=['John', 'Steven', 'Alice'])
  - a.pop(1) (removes a[1] from a and returns a[1])
- Useful methods:
  - len(a):length of a
  - max(a), min(a): largest/smallest value
  - a.sort() : a is now sorted (smaller values first)
  - a.index('Steven') = 1

Strings

- In Python strings are treated as lists of characters.
  - a=’To be, or not to be’
  - a[0]=’T’, a[:5]=’To be’...
- Some useful methods:
  - a.upper()–> ’TO BE, OR NOT TO BE’
  - a.lower()–> ’to be, or not to be’
  - a.split()–> [’To’, ’be,’ ,’or’, ’not’, ’to’, ’be’]
  - a.split(’,’)–>[’To be’, ’ or not to be’]
  - a.index(’o’)–> 1
  - a.index(’be’)–> 3
  - a.strip(): removes any leading, and trailing whitespaces.

Dictionaries

- Dictionaries are key value pairs:
  - D={’age’:30, ’height’:180, ’dob’: ’13 Jan 1992’}
- Dictionary values can be accessed via keys:
  - D[’age’]=30
  - D.keys()=[’age’, ’height’, ’dob’]
  - D.values()=[30, 180, ’13 Jan 1992’]
- Updating dictionaries:
  - D[’eyecolor’]=’brown’
  - D.update(D2): now D also contains all entries from D2.

Dictionaries

- Dictionaries are key value pairs:
  - D={'age':30, 'height':180, 'dob': '13 Jan 1992'}
- Dictionary values can be accessed via keys:
  - D['age']=30
  - D.keys()=['age', 'height', 'dob']
  - D.values()=[30, 180, '13 Jan 1992']
- Updating dictionaries:
  - D['eyecolor']='brown'
  - D.update(D2): now D also contains all entries from D2.
- Sets are unordered collection of unique elements.
  - set1={1, 2, 3, 4, 5}
  - set2={4, 5, 'Alice', 'Bob'}
  - Check if 1 is in set1: 1 in set1 –> True
- Set operations:
  - set1.add() : adds an element to the set
  - set1.update(set2/list2): add set2 or list2 to set1
  - set1.union(set2): union of sets
  - set1.intersection(set2): intersection of sets

Sets

- Sets are unordered collection of unique elements.
  - set1={1, 2, 3, 4, 5}
  - set2={4, 5, 'Alice', 'Bob'}
  - 1 in set1 –> True

Sets

- Sets are unordered collection of unique elements.
  - set1={1, 2, 3, 4, 5}
  - set2={4, 5, 'Alice', 'Bob'}
  - 1 in set1 –> True
- Set operations:
  - set1.add() : adds an element to the set
  - set1.update(set2/list2): add set2 or list2 to set1
  - set1.union(set2): union of sets (set1+set2)
  - set1.intersection(set2): intersection of sets
  - set1.difference(set2): set1-set2

Sets

- Sets are unordered collection of unique elements.
  - set1={1, 2, 3, 4, 5}
  - set2={4, 5, 'Alice', 'Bob'}
  - 1 in set1 –> True
- Set operations:
  - set1.add() : adds an element to the set
  - set1.update(set2/list2): add set2 or list2 to set1
  - set1.union(set2): union of sets (set1+set2)
  - set1.intersection(set2): intersection of sets
  - set1.difference(set2): set1-set2

Practice Session 1

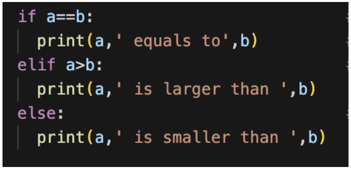
- Basic variable types and type conversions
- Arrays, dictionaries and sets

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_wise_infhaf_notebooks/-/blob/main/PythonIntroNotebooks/Lecture_02.ipynb`

Control structures

Control structures impose conditions on the execution bits of code:

- if statements:
  - condition
  - code that is executed if condition is satified/True
- elif (else if) statements:
  - can be used to impose multiple conditions
  - checked only if previous if condition is not satisfied
- else:
  - what to do if *none* of the if/elif statements are satisfied.

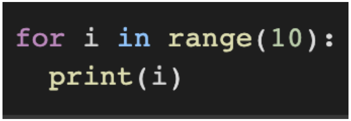


```
if a==b:
  print(a,' equals to',b)
elif a>b:
  print(a,' is larger than ',b)
else:
  print(a,' is smaller than ',b)
```

an if statement in Python

Loops

- Loops allow the repeated execution of bits of code.
  - indentation same as for if statements
- for loops are used to repeatedly execute commands over a range of values
  - range(n) : iterator from 0 to n-1
  - arrays and sets can also be used as iterators

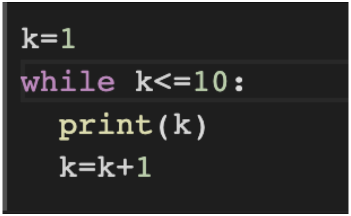


```
for i in range(10):
  print(i)
```

a for loop in Python

Loops

- while loops execute commands as long as a condition is satisfied
    - the condition should depend on the content of the loop
    - watch out for *infinite loops*!
- Loops can be nested and combined
- Control statements for loops:
    - break : stops the loop
    - continue: go back to the start of the loop

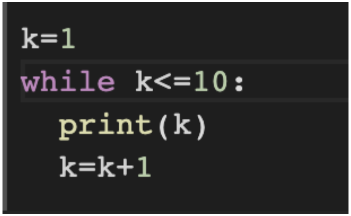


```
k=1
while k<=10:
  print(k)
  k=k+1
```

a while loop in Python

Loops

- while loops execute commands as long as a condition is satisfied
  - the condition should depend on the content of the loop
  - watch out for *infinite loops*!
- Loops can be nested and combined
- Control statements for loops:
  - break : stops the loop
  - continue: go back to the start of the loop



```
k=1
while k<=10:
  print(k)
  k=k+1
```

a while loop in Python

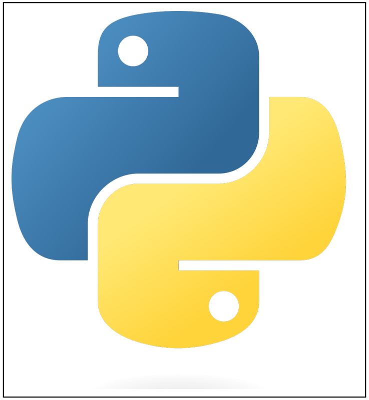
Practice Session 2

- If/elif/else statements
- for/while loops

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_wise_infhaf_notebooks/-/blob/main/PythonIntroNotebooks/Lecture_02.ipynb`

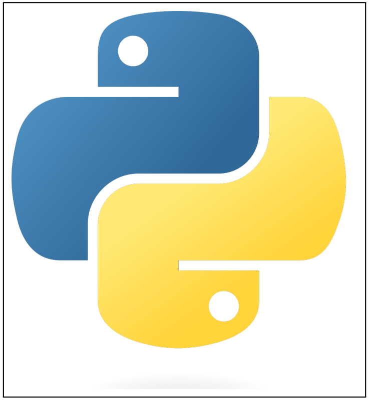
In summary

- we learned about **basic variable types and operations**
- we learned about basic Python objects such as **arrays, dictionaries and sets**



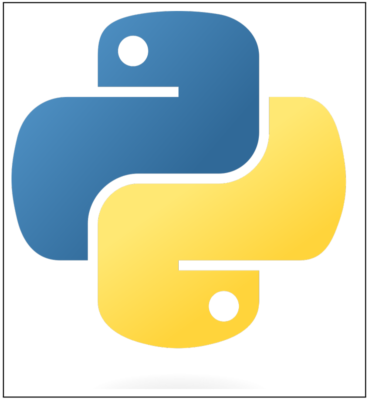
In summary

- we learned about **basic variable types and operations**
- we learned about basic Python objects such as **arrays, dictionaries and sets**
- we introduced **if, elif and else** statements
- we learned how to use **for** and **while** loops



In summary

- we learned about **basic variable types and operations**
- we learned about basic Python objects such as **arrays, dictionaries and sets**
- we introduced **if, elif and else** statements
- we learned how to use **for** and **while** loops



Exercise Session

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_wise_infhaf_notebooks/-/blob/main/PythonIntroNotebooks/ExerciseL02.ipynb`

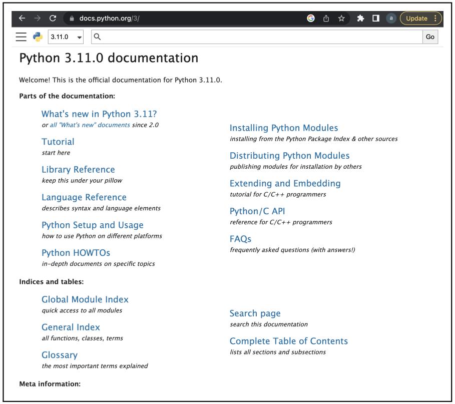
Self-study questions

1. What is a variable?
2. What are some variable types in Python?
3. What is the difference between a list, a set and a dictionary?
4. Give an example of an if statement.
5. What is the difference between a for and a while loop?

Literature

reading list

- F Kaefer, P Kaefer: **Introduction to Python Programming for Business and Social Science Applications**, SAGE Publications, 2020
- **Official Python documentation** https://docs.python.org/
- **Python tutorial**: https://docs.python.org/3/tutorial/



Python 3.11.0 documentation

Welcome! This is the official documentation for Python 3.11.0.

Parts of the documentation:

What's new in Python 3.11?
or all "What's new" documents since 2.0

Tutorial
start here

Library Reference
keep this under your pillow

Language Reference
describes syntax and language elements

Python Setup and Usage
how to use Python on different platforms

Python HOWTOs
in-depth documents on specific topics

Installing Python Modules
installing from the Python Package Index & other sources

Distributing Python Modules
publishing modules for installation by others

Extending and Embedding
tutorial for C/C++ programmers

Python/C API
reference for C/C++ programmers

FAQs
frequently asked questions (with answers!)

Indices and tables:

Global Module Index
quick access to all modules

General Index
all functions, classes, terms

Glossary
the most important terms explained

Search page
search this documentation

Complete Table of Contents
lists all sections and subsections

Meta information: