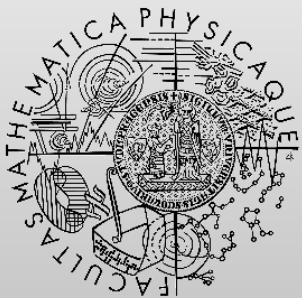


NPRG065: Programming in Python

Lecture 1

<http://d3s.mff.cuni.cz>



FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Tomas Bures

Jan Kofron

`{bures,kofron}@d3s.mff.cuni.cz`

Course information

- <https://d3s.mff.cuni.cz/teaching/nprg065/>
- All materials and announcements will be on Teams
- Course credits will be given for a homework project
 - Its qualities will determine the final grade
 - Comments, tests, overall code quality, ...

Approx. time-line of the course

- Introduction
- Core types
- Control structures
- Data structures
- Classes and objects
- Core parts of the std. library

About Python

- Dynamically-typed
 - *duck typing*
- Object-oriented language
 - there are classes but it is not a strictly class-based language
- Interpreted
 - no explicit compilation
 - “JIT” compilation to Python bytecode
- Started around 1990 by Guido Van Rossum
- Now in version 3.12
 - 2.7 – the last version of Python 2 still somewhat used
 - but unsupported since January 1, 2020
- One of the most popular languages today
 - mainly for data analysis and machine learning

"If it walks like a duck and it quacks like a duck, then it must be a duck."

About Python

- Name – why Python
 - Monty Python's Flying Circus ;-)
- Portable
 - Windows, Linux, *BSD,..., anywhere
- Installation <https://www.python.org/downloads/>
 - on Windows – download installer
 - on Linux – use a package manager
- License
 - Python Software Foundation license
 - BSD style license, can be used for anything
- PyPI – <https://pypi.python.org/>
 - Python Package Index
 - the repository of python packages

- PyCharm
 - <https://www.jetbrains.com/pycharm/>
 - Community edition – free
 - Professional edition – free for students/teachers
 - register via your university email
- Other IDEs

- Scripts
 - `my_script.py`
 - no explicit main – just start code
 - executable programs
 - `python my_script.py`
(or `python3 my_script.py`)
 - or
 - `my_script.py`
 - on unix systems
 - shebang line: `#!/usr/bin/env python3`

Shell

- Interactive shell
 - immediate evaluation
 - history (like in bash)
 - ...
 - run just **python**

```
>>> 1 + 2
```

```
3
```

```
>>>
```


Multiple Python implementations

- **CPython**
 - “the” Python
- **MicroPython**
 - a variant of CPython
 - runs on microcontrollers (pyboard, ESP32,...)
- **PyPy**
 - implementation in Python
 - JIT
- **Jython**
 - in Java, Python2 only
 - can be embedded in Java
- **IronPython**
 - in .NET
- ...

Python introduction...

- ...via examples

Hello world



No semicolons

```
print('Hello, world.')
```

No begin, no main method,...

Case sensitivity

Two variables

```
a = 1  
A = 2  
print(a)  
print(A)
```

Fibonacci numbers

```
def fib(a):  
    if a <= 1:  
        return 1  
    else:  
        return fib(a - 1) + fib(a - 2)  
  
print(fib(10))
```

No return type
No difference between
functions/procedures

No begin/end, no { }
Blocks by indentation

Multiplication table

No variable declaration

```
def multi(number):  
    print('Multiplication table of ', number)  
    for i in range(11):  
        print(i * number)
```

No “classical” **for** cycle

Fibonacci numbers v. 2

```
def Fib(k):  
    prev = 1  
    prevprev = 1  
    while k > 0:  
        tmp = prev + prevprev  
        prevprev = prev  
        prev = tmp  
        k -= 1  
    return prev
```

Command line arguments

```
import sys
```

```
print('Num. of args', len(sys.argv))  
for arg in sys.argv:  
    print(arg)
```

We will use elements
from the sys module

A list with command line
arguments

Max value in “array”

```
arr = [0, 9, 1, 8, 2, 7, 3, 6, 4, 5]
max = 0
i = 0
while i < len(arr):
    if arr[i] > max:
        max = arr[i]
    i += 1
print(max)
```

More examples #1

- Implement a function that returns a mean
 - the function takes an array as a parameter
 - returns a number which is the mean of the numbers

More examples #2

- Implement a function that prints a 2D multiplication table

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

More examples #3

- Implement a function that sorts an array in place
 - the function takes an array as a parameter
 - re-orders the values in the array to be sorted from smallest to largest
 - uses bubble-sort



The slides are licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/).