PRAHACODINGSCHOOL

Python první kroky

Datové typy, operace s proměnnými

Základní programové konstrukce

Lektor: Martin Rosický

Python: cíle v tomto kurzu

- Seznámit se se základy programovacího jazyka
  - Základní datové typy
  - Práce s proměnnými
  - Strukturované datové typy
  - Cykly, podmínky
  - Funkce
- Využití Python jako univerzálního nástroje pro
  - Čtení dat z různých zdrojů
  - Základní manipulaci s daty
  - Export dat v různých formátech

Proč Python …

… jako první?

- Je univerzální
  - od malých jednoúčelových skriptů po rozsáhlé enterprise aplikace
  - od skriptování přes GUI a webové aplikace k vědeckým výpočtům a strojovému učení
- Je dostupný na mnoha platformách
- Má širokou komunitu vývojářů
- Má rozsáhlou množinu knihoven

… a ne jiný programovací jazyk?

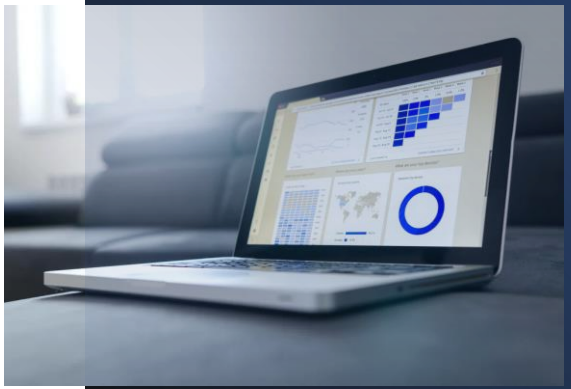
- Jako skriptový jazyk je jednodušší než kompilované jazyky
- Dává volnost různým programovacím technikám
- Dynamické typování proměnných s kontrolou kompatibility typů při běhu programu
- Je zdarma včetně knihoven i pro komerční využití

Základní stavební kameny

Práce s čísly a řetězci

Proměnné a operace s nimi

Datové struktury



Práce s řetězci

```
>>> # uvozovky
>>> "Praha Coding School"
'Praha Coding School'
>>> # apostrof
>>> 'Praha Coding School'
'Praha Coding School'
>>> # kombinace uvozovky/apostrof
>>> "Praha 'Coding' School"
"Praha 'Coding' School"
>>> # escape
>>> "Praha \"Coding\" School"
'Praha "Coding" School'
>>> """Praha "Coding" School"""
'Praha "Coding" School'
```

```
>>> # spojovování řetězců
>>> "Praha " + "Coding School"
'Praha Coding School'
>>> " ".join(['Praha','Coding','School'])
'Praha Coding School'
>>> # malá, velká
>>> "Praha Coding School".upper()
'PRAHA CODING SCHOOL'
>>> "Praha Coding School".lower()
'praha coding school'
>>> # délka řetězce
>>> len("Praha Coding School")
19
>>> # formátování
>>> “{} {} School".format("Praha","Coding")
'Praha Coding School'
```

Python jako kalkulačka

```
>>> # sčítání
>>> 5 + 3
8
>>> # odčítání
>>> 5 - 3
2
>>> # násobení
>>> 5 * 3
15
>>> # dělení
>>> 5 / 3
1.6666666666666667
>>> # celočíselné dělení
>>> 5 // 3
1
>>> # zbytek po dělení
>>> 5 % 3
2
>>> # mocnina
>>> 5 ** 3
125
```

- Číselné typy

int	celá čísla
float	čísla s desetinnou čárkou
complex	komplexní čísla complex(„2+3j“)

```
>>> complex("j") ** 2
(-1+j)
>>> complex("3.2+4.15j") + complex(2.3+5.14j)
(5.5+9.29j)
```

- Převody mezi int a float

```
>>> float(9)
9.0
>>> float("-200")
-200.0
>>> int(9.6)
9
>>> int("-200")
-200
```

Typ ‘bool’ a podmínky

```
>>> int(False)
0
>>> int(True)
1
>>> bool(0)
False
>>> bool(1)
True
>>> bool("")
False
>>> bool("Praha")
True
>>> bool([])
False
>>> bool(['Praha','Coding','School'])
True
```

```
>>> 5 > 3
True
>>> 5 < 3
False
>>> 5 >= 3
True
>>> 5 <= 3
False
>>> 5 == 5
True
>>> 5 != 5
False
>>> "a" == "a"
True
>>> "ab" >= "a"
True
```

Proměnné a operace s nimi

```
>>> city = 'Praha'
>>> action = 'Coding'
>>> school = city+' '+action+' School'
>>> school
'Praha Coding School'
>>> f'{city} {action} School'
'Praha Coding School'
>>> school = '{city} {action} School'
>>> school.format(city=city,action=action)
'Praha Coding School'
>>> city = 'Lipno'
>>> action = 'Yachting'
>>> f'{city} {action} School'
'Lipno Yachting School'
>>> school.format(city=city,action=action)
'Lipno Yachting School'
```

```
>>> a = 1
>>> b = 3
>>> c = 5
>>> a + b + c
9
>>> c = a + b + c
>>> c
9
>>> f'{c}.{a}-{b}'
'9.1-3'
>>> c == 3 * b
True
>>> city[a:b]
'ip'
>>> city[0]
'L'
>>> city[-1]
'o'
```

Datové struktury

- List
  - Uspořádaný seznam hodnot /i různých typů/
- Set
  - Neuspořádaná množina unikátních hodnot /i různých typů/
- Tuple
  - Uspořádaný neměnitelný seznam hodnot /i různých typů/
- Dictionary
  - Kolekce „slovník“ párů klíč - hodnota

List a operace s ním

```
>>> # vytvoření
>>> empty = []
>>> numbers = [6,5,4,3,2,1]
>>> cities = ['Brno','Ostrava','Plzeň','Praha']
>>> ["dohromady",empty,numbers,cities]
['dohromady', [], [6, 5, 4, 3, 2, 1],
  ['Brno', 'Ostrava', 'Plzeň', 'Praha']]
>>> 'Praha Coding School'.split('')
['Praha', 'Coding', 'School']
>>> numbers[2:4]
[4, 3]
>>> # existence
>>> 5 in numbers
True
>>> 'Zlín' in cities
False
>>> cities[2]
'Plzeň'
```

```
>>> # sčítání
>>> numbers + cities
[6, 5, 4, 3, 2, 1,
  'Brno', 'Ostrava', 'Plzeň', 'Praha']
>>> # seřazení
>>> numbers.sort()
>>> numbers
[1, 2, 3, 4, 5, 6]
>>> # změna položky
>>> numbers[5] = 33
>>> numbers
[1, 2, 3, 4, 5, 33]
>>> # přidání položky
>>> cities.append('Zlín')
['Brno', 'Ostrava', 'Plzeň', 'Praha', 'Zlín']
>>> # délka
>>> len(cities)
5
```

Tuple a operace s ním

```
>>> # vytvoření
>>> nums_tup = (6,5,4,3,2,1)
>>> cts_tup= tuple(cities)
>>> cts_tup
('Brno', 'Ostrava', 'Plzeň', 'Praha', 'Zlín')
>>> throw_tup = (6,6,3,3,2,1)

>>> # délka
>>> len(cts_set)
5
>>> # existence
>>> 2 in nums_tup
True
>>> throw_tup.count(3)
2
>>> throw_tup.index(3)
2
```

```
>>> # výběr hodnot
>>> cts_tup[2]
'Plzeň'
>>> cts_tup[2:4]
('Plzeň', 'Praha')

>>> # sčítání
>>> nums_tup + cts_tup
(6, 5, 4, 3, 2, 1,
  'Brno', 'Ostrava', 'Plzeň', 'Praha', 'Zlín')

>>> # změna hodnoty = chyba
>>> cts_tup[2] = 'Liberec'
Traceback (most recent call last):
  File „<input>,  line 1,  in <module>
    cts_tup[2] = 'Liberec'
TypeError: 'tuple' object does not support item
assignment
```

Set a operace s ním

```
>>> # vytvoření
>>> empty_set = set()
>>> nums_set = {6,5,4,3,2,1}
>>> cts_set = set(cities)
>>> cts_set
{'Ostrava', 'Plzeň', 'Zlín', 'Brno', 'Praha'}
>>> throw = {6,6,3,3,2,1}
>>> throw
{1, 2, 3, 6}

>>> # existence
>>> 2 in throw
True
>>> # délka
>>> len(cts_set)
5
```

```
>>> # sjednocení
>>> throw | cts_set
{1, 2, 3, 'Ostrava', 6, 'Plzeň', 'Zlín',
   'Brno', 'Praha'}
>>> # průnik
>>> throw & {5,4,3,2}
{2, 3}
>>> # rozdíl
>>> nums_set - {7,4,3,2}
{1, 5, 6}
>>> # symetrický rozdíl
>>> nums_set ^ {7,4,3,2}
{1, 5, 6, 7}
>>> # přidání položky
>>> cts_set.add('Liberec')
{'Brno', 'Ostrava', 'Plzeň', 'Praha',
'Liberec', 'Zlín'}
```

Dict a operace s ním

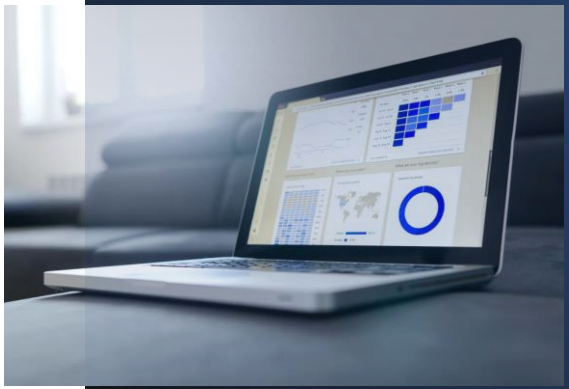
```
>>> # vytvoření
>>> empty_dict = {}
>>> person = {'name': 'Martin Rosický', 'city': 'Černolice', 'id': 321}
>>> course = dict(name='Datová analýza', students=8, lector=person)
{'name': 'Datová analýza', 'students': 8, 'lector': {'name': 'Martin Rosický', 'city': 'Černolice', 'id': 321}}
>>> # délka
>>> len(person)
3
>>> 'name' in person
True
>>> course.get('name')
('Datová analýza')
>>> course.get('id', 63)
63
>>> course['id'] = 636
>>> course.get('id', 63)
636
```

```
>>> course.keys()
dict_keys(['name', 'students', 'lector', 'id'])
>>> person.values()
dict_values(['Martin Rosický', 'Černolice', 321])
>>> person.items()
dict_items([('name', 'Martin Rosický'),
            ('city', 'Černolice'), ('id', 321)])
```

Základní programové konstrukce

Cykly

Podmínky a větvení



Cykly v Pythonu

FOR

```
for loop_var in iterable:
    # opakuj blok pro všechny položky
    # můžeš využít <loop_var>
    if break_condition:
        break # opusť cyklus
    if continue_condition:
        continue # pokračuj další položkou
    # zbylý kód bloku
else:
    # proveď, pokud cyklus nebyl přerušen

# zbytek programu
```

WHILE

```
while expression:
    # opakuj blok, dokud <expression> je True
    if break_condition:
        break # opusť cyklus
    if continue_condition:
        continue # pokračuj od začátku
    # zbylý kód bloku
else:
    # proveď, pokud cyklus nebyl přerušen

# zbytek programu
```

Náš první prográmek

```
MAX_ITER = 10
''' maximalni pocet iteraci ,‘’


print("for ... in ...")
for ii in range(0,MAX_ITER):
    print(ii)


print("\nwhile ...")
ii = MAX_ITER
while ii > 0:
    print(ii)
    ii -= 1 #změna vstupu podmínky!
```

```
for ... in ...
0
1
2
3
4
5
6
7
8
9
while ...
10
9
8
7
6
5
4
3
2
1
```

Větvení programu – „if“

```
MAX_ITER = 10
''' maximalni pocet iteraci '''
SPLIT1 = MAX_ITER // 3
''' prvni rozhodovaci bod '''
SPLIT2 = 2 * SPLIT1
''' druhy rozhodovaci bod ',‘

for ii in range(0,MAX_ITER):
    if ii < SPLIT1:
        print("skupina 1")
    elif ii < SPLIT2:
        print("skupina 2")
    else:
        print("skupina 3")
    print(ii)
```

```
for ... in ...
skupina 1
0
skupina 1
1
skupina 1
2
skupina 2
3
skupina 2
4
skupina 2
5
skupina 3
6
skupina 3
7
skupina 3
8
skupina 3
9
```

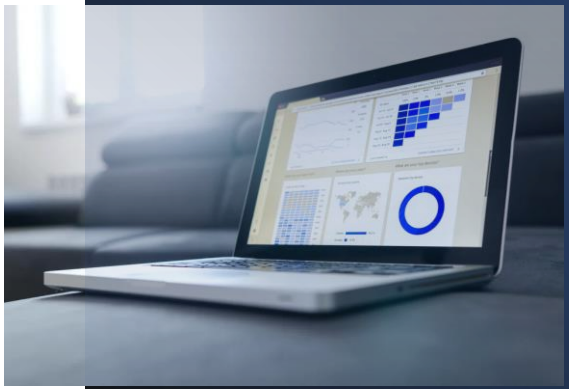
Větvení programu - „match … case“

```
ii = MAX_ITER
while ii > 0:
    match ii:
        case 1:
            print("jednicka")
        case 2:
            print("dvojka")
        case 3:
            print("trojka")
        case 4:
            print("ctyrka")
        case 5:
            print("petka")
        case 6:
            print("sestka")
        case _:
            print(f"{ii} neni na kostce")
   ii -= 1
```

```
while ...
10 neni na kostce
9 neni na kostce
8 neni na kostce
7 neni na kostce
sestka
petka
ctyrka
trojka
dvojka
jednicka
```

Základní programové konstrukce

Procedury a funkce



Funkce – vrací výsledek

```
def get_group(val):
    output = ""
    if val < SPLIT1:
        output = "skupina 1"
    elif val < SPLIT2:
        output = "skupina 2"
    else:
        output = "skupina 3"
    return output


print("for ... in ...")
for ii in range(0,MAX_ITER):
     print(f"{ii} {get_group(ii)}")
```

```
for ... in ...
0 skupina 1
1 skupina 1
2 skupina 1
3 skupina 2
4 skupina 2
5 skupina 2
6 skupina 3
7 skupina 3
8 skupina 3
9 skupina 3
```

Procedura – nevrací výsledek

```python
def print_name(val):
    match val:
        case 1:
            print("jednicka")
        case 2:
            print("dvojka")
        ⋮
        case 5:
            print("petka")
        case 6:
            print("sestka")
        case _:
            print(f"{val} neni na kostce")


print("\nwhile ...")
ii = MAX_ITER
while ii > 0:
     print_name(ii)
     ii = ii - 1
```

```
while ...
10 neni na kostce
9 neni na kostce
8 neni na kostce
7 neni na kostce
sestka
petka
ctyrka
trojka
dvojka
jednicka
```

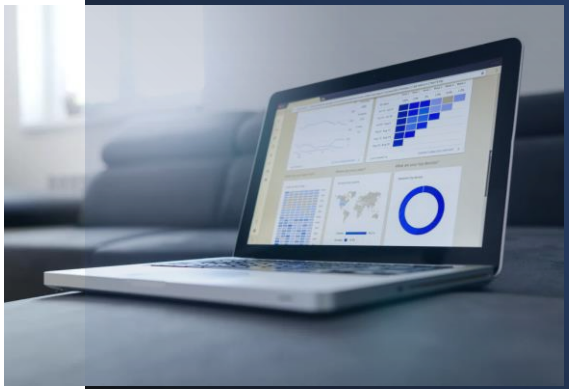
Komplexní argumenty funkce

```
def funkce(a,b,*l,**d): # Argumenty funkce
  print("a:",type(a))   # „a“ poziční
  print("b:",type(b))   # „b“ poziční
  print("l:",type(l))   # „l“ nejmenované
  print("d:",type(d))   # „d“ jmenované

  print("\nsoucet a+b",a+b)
  print("\nObsah 'l':")

  for ii in l:
    print(ii)

  print("\nObsah 'd':")
  for k,v in d.items():
    print(f"{k} => {v}")
```

```
print("#"*20,"\nfunkce(a,b)")
funkce(3,2)

l = range(10,20)
print("\n"*2,"#"*20,"\nfunkce(a,b,l)")
funkce(5,6,l)

print("\n"*2,"#"*20,"\nfunkce(a,b,*l)")
funkce(5,6,*l)

d = {'test':'text'}
print("\n"*2,"#"*20,"\nfunkce(a,b,*l,d)")
funkce(5,6,*l,d)

print("\n"*2,"#"*20,"\nfunkce(a,b,*l,**d)")
funkce(5,6,*l,**d)

print("\n"*2,"#"*20,
  "\nfunkce(a,b,*l,test='text',test2='text2')")
funkce(5,6,*l,test='text',test2='text2')
```

Základní programové konstrukce

Cykly – přerušení běhu



Loop – continue – else

```
BREAK_POINT = 7
''' preruseni zpracovani '''

print("for ... in ...")
for ii in range(0,MAX_ITER):
    if ii % 2 == 0:
        continue
    print_name(ii)
    print(f"{ii} {get_group(ii)}")
else:
    print("dosli jsme na konec")
```

```
for ... in ...
jednicka
1 skupina 1
trojka
3 skupina 2
petka
5 skupina 2
7 neni na kostce
7 skupina 3
9 neni na kostce
9 skupina 3
dosli jsme na konec
```

Loop – continue – break – else

```
BREAK_POINT = 7
''' preruseni zpracovani '''

print("for ... in ...")
for ii in range(0,MAX_ITER):
    if ii == BREAK_POINT:
        break
    if ii % 2 == 0:
        continue
    print_name(ii)
    print(f"{ii} {get_group(ii)}")
else:
    print("dosli jsme na konec")
```

```
for ... in ...
jednicka
1 skupina 1
trojka
3 skupina 2
petka
5 skupina 2
```