# IB111

## Základy programování

František Lachman lachmanfrantisek@mail.muni.cz

cvičení 2

25. září 2017

# Osnova

- kontrolní otázky
  - opakování z minulého cvičení
  - základní konstrukce
- příklady
  - posloupnosti
  - tabulky
  - textová grafika
- zadání 1. domácí úlohy

# Docházka

?

# Kontrolní otázky

?

# Jak v Pythonu deklarujeme proměnné?

```
my_cool_variable = 1
print(my_cool_variable)

second_variable = my_cool_variable
print(second_variable)

second_variable = 2
print(my_cool_variable, second_variable)
```

```
1
1
1 2
```

# Jak se určuje typ proměnné?

# Jak zjistíme typ proměnné?

# Může se změnit typ proměnné?

# Jak lze přetypovat číslo na řetězec?

```
my_variable = "Simple is better than complex."
print(my_variable)
print(type(my_variable))

my_variable = 42
print(my_variable)
print(type(my_variable))

my_text = str(my_variable)
print(my_variable)
print(type(my_text))
```

```
Simple is better than complex.
<class 'str'>

42
<class 'int'>

42
<class 'str'>
```

# Jaký je rozdíl mezi

`x = 5` a `x == 5`?

```
>>> number = 5

>>> print(number = 5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'number' is an invalid keyword argument for thi

>>> print(number == 5)
True

>>> print(number == 6)
False
```

# Co znamená `var += 1`?

```
number = 5
number += 3
print(number)
```

```
8
```

# Jak zapisujeme operaci "umocňování"?

```
power = 8
base = 2
print(base ** power)
```

```
256
```

# Jak v Pythonu vyznačujeme blok kódu?

**Jak oddělujeme jednotlivé příkazy?**

**Jak oddělujeme řídící příkaz (`if` / `while` / `for`) od následujících příkazů?**

# Jaký je rozdíl mezi cykly `for` a `while`?

```
known_count = 3
for i in range(known_count):
    print(i)
```

```
0
1
2
```

```
max_count = 3
i = 0
while max_count > i:
    print(i)
    i += 1
```

```
0
1
2
```

# Cyklus

```
for i in range(10):
    print(i, end=" ")
print()
# 0 1 2 3 4 5 6 7 8 9
```

```
for i in range(5, 9):
    print(i, end=" ")
print()
# 5 6 7 8
```

```
for i in range(1, 10, 2):
    print(i, end=" ")
print()
# 1 3 5 7 9
```

# Range ✕ seznam

- `range(n)` - objekt "sekvenčního" typu
- `list(range(n))` - seznam vygenerovaný z prvků `range(n)`

```
my_range = range(10)
print(my_range)
print(list(my_range))
```

```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

# Proč při programování používáme funkce?

# Jak zapisujeme definici funkce?

# Funkce

```
def numbers(n):
    for i in range(1, n + 1):
        print(i, end=" ")
    print()
```

```
numbers(5) # 1 2 3 4 5
```

```
numbers(10) # 1 2 3 4 5 6 7 8 9 10
```

# Jak zapíšeme parametry s defaultní hodnotou?

```python
def numbers(n=5):
    for i in range(1, n + 1):
        print(i, end=" ")
    print()
```

```python
numbers() # 1 2 3 4 5
```

```python
numbers(10) # 1 2 3 4 5 6 7 8 9 10
```

# Jak do zdrojového kódu zapisujeme komentáře?

```
var = 5 # my one-line comment

"""
My block comment
can have
multiple lines.
"""

def hello(person):
    """
    Function description.

    :param person: description of parameter
    """
    print("Hello", person)
```

# Co znamená PEP8?

- https://pep8.org/
- https://github.com/PyCQA/pylint

```
$ pylint code.py
************* Module code
C: 23, 0: Missing class docstring (missing-docstring)
C: 24, 0: Constant name "logger" doesn't conform to UPPER_
C: 27, 8: Variable name "im" doesn't conform to snake_case
W: 32, 8: Unreachable code (unreachable)
C: 59, 0: Wrong continued indentation (remove 1 space).
                                    )
                                   |^ (bad-continuation)
R: 91, 0: Too many arguments (6/5) (too-many-arguments)
C: 91, 0: No space allowed before bracket
def _get_checks (target_type, tags=None,
                ^ (bad-whitespace)
C: 93, 0: Exactly one space required before assignment
    ruleset= Ruleset(ruleset_name=ruleset_name,
           ^ (bad-whitespace)
```

**Jaké jsou příklady konvencí, které nejsou „povinné“, ale je velmi vhodné je dodržovat?**

# Příklady - 2.1. Posloupnosti

- 2.1.1. Sudá čísla
  Napište funkci `even_numbers(n)`, která vypíše prvních n sudých čísel větších než 0.

```
def even_numbers(n):
    for i in range(1, n + 1):
        print(2 * i, end=" ")
    print()
```

- 2.1.2. Mocniny
  Napište funkci `powers(base, n)`, která vypíše prvních n mocnin o daném základu.

```
def powers(base, n):
    current = 1
    for i in range(n):
        print(current, end=" ")
        current = current * base
    print()
```

- 2.1.4. Fibonacci
  Napište funkci `fibonacci(n)`, která vypíše prvních n prvků Fibonacciho posloupnosti.

```
def fibonacci(n):
    current_number = 1
    next_number = 1
    for i in range(n):
        print(current_number, end=" ")
        next_next = current_number + next_number
        current_number = next_number
        next_number = next_next
    print()
```

# Výpis bez přechodu na nový řádek

```
print(text, end=" ")
```

```
>>> print("1"); print("2")
1
2
```

```
>>> print("1", end=" "); print("2")
1 2
```

# 2.3.1. Vyplněný čtverec

Napište funkci `square(n)`, která v textové grafice vykreslí vyplněný čtverec o straně n.

```
>>> square(3)
# # #
# # #
# # #
```

```
def square(n):
    for i in range(n):
        for j in range(n):
            print('#', end=' ')
        print()
```

# 2.3.3. Pyramida

Napište funkci `pyramid(n)`, která v textové grafice vykreslí pyramidu o velikosti n.

```
>>> pyramid(3)
    #
  # # #
# # # # #
```

```
def pyramid(n):
    for i in range(n):
        for j in range(2 * n - 1):
            if j >= n - 1 - i and j <= n - 1 + i:
                print('#', end=" ")
            else:
                print(' ', end=" ")
        print()
```

# 2.3.6. Kříž

Napište funkci `cross(n)`, která v textové grafice vykreslí kříž, jehož ramena budou mít délku n.

```
>>> cross(2)
#   #
  #
#   #
```

```
def cross(n):
    for i in range(2 * n - 1):
        for j in range(2 * n - 1):
            if i == j or (2 * n - 2) - i == j:
                print('#', end=" ")
            else:
                print(' ', end=" ")
        print()
```

# 2.2.1. Násobilka

Napište funkci `table_products(n)`, která vypíše tabulku s daným počtem řádků a sloupců (+ popisný řádek a sloupce), kde v každé buňce se nachází součin čísla řádku a čísla sloupce.

```
>>> table_products(5)
    1  2  3  4  5
    -  -  -  -  -
1 | 1  2  3  4  5
2 | 2  4  6  8  10
3 | 3  6  9  12 15
4 | 4  8  12 16 20
5 | 5  10 15 20 25
```

```
def table_products(n):

    print(' ', ' ', end=" ")
    for col in range(1, n + 1):
        print(col, end=" ")
    print()

    print(' ', ' ', end=" ")
    for col in range(n):
        print('-', end=" ")
    print()

    for row in range(1, n + 1):
        print(row, '|', end=" ")
        for col in range(1, n + 1):
            print(row * col, end=" ")
        print()
```

# Příklady - 2.2. Tabulky

- 2.2.3. Sčítání `table_additions(n)`:

```
>>> table_additions(5)

    1 2 3 4 5
    - - - - -
1 | 2 3 4 5 6
2 | 3 4 5 6 7
3 | 4 5 6 7 8
4 | 5 6 7 8 9
5 | 6 7 8 9 10
```

```
def table_additions(n):
    print(' ', ' ', end=" ")
    for col in range(1, n + 1):
        print(col, end=" ")
    print()
    print(' ', ' ', end=" ")
    for col in range(n):
        print('-', end=" ")
    print()
    for row in range(1, n + 1):
        print(row, '|', end=" ")
        for col in range(1, n + 1):
            print(row + col, end=" ")
        print()
```

- maximum, mocniny, zbytek po dělení, …

# 1. Domácí úloha

- Deadline: 2. 10. 2018 11:59
- Softdeadline: 30. 9. 2018
- Celkem 25 bodů (~3-5 na úlohu)
- Odevzdání do odevzdávárny v ISe:
  `Skupina 03 (Lachman)/Domácí úkol 1`
  odkaz na složku

# Jeden soubor `UČO.py`

- Prohlášení a sebehodnocení (Pokyny k domácím úlohám)

```
"""
Autor: Jméno Příjmení, UČO

Prohlašuji, že celý zdrojový kód jsem zpracoval(a) zcela s
Jsem si vědom(a), že  nepravdivost tohoto tvrzení může být
"""

def my_hw_function():
    """
    Description.

    Známé nedostatky: Rámcově funguje, ale nevykresluje př
    Styl: Příliš kryptická jména funkcí a proměnných.
    """
```

# 1. Domácí úloha (1)

Napiště funkce `alternating_multiples(n)` vypisující alternující násobky daného čísla:

```
>>> alternating_multiples(2)
0 -2 4 -6 8 -10 12 -14 16 -18 20
```

```
>>> alternating_multiples(3)
0 -3 6 -9 12 -15 18 -21 24 -27 30
```

# 1. Domácí úloha (2)

- Navrhněte funkci `crossing(n, length)` vykreslující přechod o `n` bílých pruzích délky `length`.

```
>>> crossing(4, 8)
# # # # # # # # #
#               #
# # # # # # # # #
#               #
# # # # # # # # #
#               #
# # # # # # # # #
#               #
# # # # # # # # #
```

- Mezi vodorovnými `#` jsou mezery.

# 1. Domácí úloha (3)

Napište funkci pyramid(n), která vykreslí textovou grafikou pyramidu podle následujícího vzoru. Parametr n udává výšku pyramidy (počet řádků). Příklad:

```
def pyramid(n):
```

```
>>> pyramid(5)
        #
      # . #
    # . . . #
  # . . . . . #
# # # # # # # # #
```

# 1. Domácí úloha (4)

- Napiště funkci `mocniny(m, n)`, která vypíše tabulku pro `m` čísel s prvními `n` mocninami.

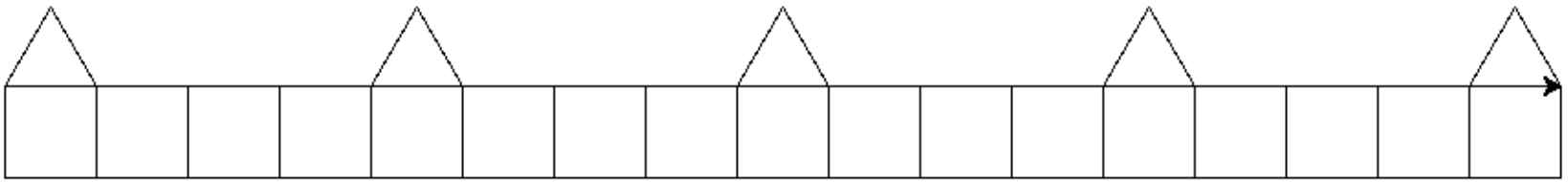
```
def mocniny(m, n):
    pass
```

```
>>> mocniny(5, 3)
    1  2  3
    -  -  -
1 | 1  1  1
2 | 2  4  8
3 | 3  9  27
4 | 4  16 64
5 | 5  25 125
```

# 1. Domácí úloha (5) 🐢

- Navrhněte funkci `castle(towers, length, space)`, která pomocí želví grafiky nakreslí hrad, který má daný počet věží (`towers`) a mezer mezi nimi (`space`). Hranu základních čtverců a trojúhelníků lze definovat parametrem `length`.

```
>>> castle(5, 50, 3)
```



# 1. Domácí úloha (6) 🐢

- Nakreslete pomocí želví grafiky hada
  - ideálně krajtu (=python..;-)

```
def draw_python()
    pass
```

- Mělo by jít poznat, že se jedná o hada.
- Za pěkné provedení budou bonusové body.

# Osnova

- kontrolní otázky
  - opakování z minulého cvičení
  - základní konstrukce
- příklady
  - posloupnosti
  - tabulky
  - textová grafika
- zadání 1. domácí úlohy