IB111

Základy programování

František Lachman
lachmanfrantisek@mail.muni.cz

cvičení 3

4. říjen 2017

Osnova

- kontrolní otázky
- jednoduché výpočty
- příklady
- zpětná vazba k domácím úlohám

Kontrolní otázky (I.)

- Jaký je rozdíl mezi číselnými typy `int` a `float`?
- Jak zapisujeme operace „celočíselné dělení“ a „dělení se zbytkem“?
- Co dělají funkce `round`, `math.floor`, `math.ceil`?

Kontrolní otázky (II.)

- Jaká je základní myšlenka algoritmu pro výpočet ciferného součtu?
- Jaká je základní myšlenka algoritmu pro výpočet odmocniny?
- Jaká je základní myšlenka algoritmu pro převod na binární zápis?

Kontrolní otázky (III.)

- Jaký je význam příkazu return? Jaký je rozdíl mezi tím, když na konci funkce zavoláme return a print?
- Jakým způsobem vygenerujeme náhodné číslo?

Návratová hodnota – `return`

```
def five():
    return 5
```

```
>>> a = five()
>>> print(a)
5
```

```
def double(a):
    b = a * 2
    return b # zkráceně return a * 2
```

```
>>> b = double(5)
>>> print(b)
10
```

Návratová hodnota – `return`

```
>>> x = five()
>>> y = double(x)
>>> print(x + y)
15
```

```
>>> print(double(five()))
10
```

```
>>> print(double(double(double(double(five())))))
80
```

Návrat z funkce pomocí `return`

```
def nothing():
    return
```

```
>>> nothing()
>>> a = nothing()
>>> type(a)
NoneType
```

```
def my_function():
    print("jsem tu")
    return
    print("sem se vypocet nikdy nedostane")
```

Návrat z funkce pomocí `return`

```
def my_max(number1, number2):
    if number1 == number2:
        print("jsou stejna")
        return number1
    if number1 < number2:
        print("prvni je vetsi")
        return number1
    print("druhe je vetsi")
    return number2

>>> a = my_max(2, 2)
jsou stejna
>>> b = my_max(2, 3)
prvni je vetsi
>>> c = my_max(3, 2)
druhe je vetsi
>>> print(a, b, c)
2 3 3
```

Prázdné tělo - `pass`

```
def without_return():
    pass
```

- spustitelný kód bez implementace
- explicitně vyjádřené "nedělání ničeho"

Cyklus `while`

```
# prvni mocnina je 1
power = 1

while power < 10000:
    # pokracujeme v cyklu jen pokud je mocnina
    # stale mensi nez 10 000
    print(power, end=" ")

    # vypocitame dalsi mocninu v poradi
    power = power * 3
```

Příklady - 3.1. Pokročilé počítání

- Součet čísel od 1 do n
  Napište funkci `series_sum(n)`, která vrátí součet čísel od `1` do `n`.
- 3.1.1. Faktoriál pomocí for
  Napište funkci `factorial(n)`, která vrací faktoriál čísla `n` a využívá cyklus `for`.
  (Připomeňme, že n! = 1·2·3· ... ·n a že 0! = 1.)
- 3.1.3. Ciferný součet
  Napište funkci `digit_sum(n)`, která vrátí ciferný součet čísla `n`.

Příklady - 3.2. Dělitelnost a prvočísla

- 3.2.1. Dělitelé
  Napište funkci `divisors(n)`, která vypíše všechny dělitele čísla `n`.
- 3.2.2. Počet dělitelů
  Napište funkci `divisors_count(n)`, která vrátí počet dělitelů čísla `n`.

Příklady - 3.2. Dělitelnost a prvočísla

- 3.2.3. Je prvočíslo
  Napište funkci `is_prime(n)`, která vrátí `True` pokud je číslo `n` prvočíslo, jinak `False`.
- 3.2.4. Prvočísla menší než n
  Napište funkci `primes_less_than(limit)`, která vypíše všechna prvočísla menší než `limit`.
- 3.2.12. Euklidův algoritmus
  Implementujte Euklidův algoritmus pomocí funkce `euklid(a, b)`.

Příklady - 3.3. Aproximace

- 3.3.1. Eulerovo číslo
  Napište funkci `e()` pro přibližný výpočet Eulerova čísla pomocí nekonečného součtu s přesností na miliontiny.

$$\mathrm{e} = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$

Příklady - 3.4. Převody mezi číselnými soustavami

- 3.4.1. Převod z desítkové soustavy do dvojkové
  Napište funkci `convert_10_to_2(n)`, která vrátí zadané číslo `n` ve dvojkové soustavě jako řetězec.
- 3.4.2. Převod z dvojkové soustavy do desítkové
  Napište funkci `convert_2_to_10(n)`, která vrátí zadaný řetězec `n` reprezentující binární číslo v desítkové soustavě jako int.

První domácí úloha (1)

```
def alternating_multiples(base):
    sign = 1
    for i in range(0, base*10):
        print(base * i * sign, end=" ")
        sign *= -1
```

První domácí úloha (2)

```
def crossing(n, lenght):
    for i in range(2*n +1):
        if i % 2 == 0:
            print((lenght+2)*'# ',)
        else:
            print('#   ', (lenght-2)*'  ','#')
```

```
def crossing(n, lenght):
    for i in range(0,n*2+1):
        print("#", end=" ")
        for j in range(lenght):
            if (i%2==0):
                print("#", end=" ")
            else:
                print(" ", end=" ")
        print("#")
```

První domácí úloha (3)

```
def mocniny(m,n):
    print('  ', '  ', end=" ")
    for i in range(1, m+1):
        print(i, end=" ")
    print()
    print('  ', '  ', end=" ")
    for i in range(m):
        print('-', end=" ")
    print()

    for j in range(1,n+1):
        print(j, '|', end=" ")
        for i in range(1, m+1):
            print(i ** j, end=" ")
        print()
```

První domácí úloha (..3)

```
def mocniny(m, n):
    print ("    ", end=" ")
    for x in range (1, m+1):
        print ('{0:3d}'.format(x), end=" ")
    print()
    print ("    ", end=" ")
    for z in range (1, m+1):
        print ("-".rjust(3), end=" ")
    print()
    for i in range(1, n+1):
        print (i, end=" ")
        print ("|", end=" ")

        for j in range(1, m+1):
            print ('{0:3d}'.format(j**i), end=" ")
        print()
```

První domácí úloha (4)

```
def tower(length):
    polygon(4, length)
    left(60)
    forward(length)
    right(120)
    forward(length)
    left(60)

def castle(towers, length, space):
    for _ in range(towers-1):
        tower(length)
        for _ in range(space):
            polygon(4, length)
            forward(length)
    if (towers > 0):
        tower(length)
```

První domácí úloha (5)

- výstavka příště

První domácí úloha

- Pozor na:
  - české názvy proměnných
  - žádné/přezbytečné komentáře
  - formátování
    - výstupy `pylint` (`pep8`) ve studijních materiálech