# Rozdělení programu do funkcí

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZPP1 Základy programování v Pythonu 1

# Rozdělení programu do funkcí

# Zadání

- Naprogramujme funkci, odstraňující duplicity ze zadaného seznamu.
- Funkci pojmenujeme

`remove_duplicates`

- a měla by fungovat následovně:

```
>>> remove_duplicates ([1 , 2, 1, 2, 2])
[1, 2]
>>> remove_duplicates ([])
[]
>>> remove_duplicates ([1 , 2, 3])
[1, 2, 3]
```

# Řešení

- Odstraňování duplicit budeme provádět tak, že do proměnné result budeme přidávat jen ty prvky vstupního seznamu, které se v ní ještě nenalézají.
- Proměnnou, jejíž hodnotu vracíme budeme nazývat výstupní.
- Zaceňme kostrou programu:

```
def remove_duplicates ( input_list ):
    result = []
    # ...
    return result
```

# Řešení

- Přidáme procházení prvku seznamu input_list:

```
def remove_duplicates ( input_list ):
    result = []
    for i in range ( len( input_list )):
        element = input_list [i]
        is_member = False
        # ...
        if not is_member :
            result = result + [ element ]
    return result
```

# Řešení

- Zbývá dodat test, zda je aktuální prvek element přítomen v seznamu result:

```
def remove_duplicates ( input_list ):
    result = []
    for i in range ( len( input_list )):
        element = input_list [i]
        is_member = False
        for j in range ( len( result )):
            list_element = result [j]
            if list_element == element :
                is_member = True
                break
        if not is_member :
            result = result + [ element ]
    return result
```

# Řešení

- Testování přítomnosti v seznamu umístíme do zvláštní funkce:

```
def is_element_member ( element , input_list ):
    is_member = False
    for i in range ( len( input_list )):
        list_element = input_list [i]
        if list_element == element :
            is_member = True
            break
    return is_member
```

- Funkci otestujeme:

```
>>> is_element_member (1, [1, 2, 1, 4])
True
>>> is_element_member (5, [1, 2, 1, 4])
False
```

# Řešení

- Nyní již můžeme nahradit ve funkci remove_duplicates problematický kód voláním funkce:

```
def remove_duplicates ( input_list ):
    result = []
    for i in range ( len( input_list )):
        element = input_list [i]
        if not is_element_member ( element , result ):
            result = result + [ element ]
    return result
```

- Jak je možné, že při volání funkce is_element_member nepřijdeme o hodnotu proměnné input_list funkce remove_duplicates, když prvně jmenovaná funkce má parametr také pojmenovaný input_list?

# Úkol 1

- Napište program, který bude umět sčítat, dva zlomky. Součet potom uvede do základního tvaru.
- Postup:
  1. Navrhnout reprezentaci zlomků a vytvořit funkci pro vytvoření zlomků.
  2. Napsat funkci pro součet zlomků.
  3. Uvést součet do základního tvaru.

# Řešení

- Zlomky budeme reprezentovat pomocí seznamu:

```
def make_fract (num , den ):
    if den == 0:
        return num // den
    else :
        return [num , den]
```

- Funkce pro získání čitatele a jmenovatele:

```
def get_num ( fract ):
    return fract [0]

def get_den ( fract ):
    return fract [1]
```

- Funkce pro tisk zlomku:

```
def print_fract ( fract ):
    print ('make_fract (', end ='')
    print ( get_num ( fract ), end=', ')
    print ( get_den ( fract ), end='')
    print (')')
```

# Řešení

- Funkce pro součet zlomků:

```
def add_fracts (fract1 , fract2 ):
    num1 = get_num ( fract1 )
    den1 = get_den ( fract1 )
    num2 = get_num ( fract2 )
    den2 = get_den ( fract2 )
    num_result = num1 * den2 + num2 * den1
    den_result = den1 * den2
    return make_fract ( num_result , den_result )
```

# Řešení

- Funkce pro výpočet NSD:

```
def compute_gcd (n, m):
    while m != 0:
        tmp = m
        m = n % m
        n = tmp
    return n
```

- Funkce pro zjednodušení zlomku:

```
def reduce_fract ( fract ):
    num = get_num ( fract )
    den = get_den ( fract )
    gcd = compute_gcd (num , den)
    return make_fract (num // gcd , den // gcd)
```

# Rozsah platnosti

# Příklad

```
def f1(b):
    b = b + 1
    return b

def f2(b):
    f1(b)
    return b

print (f2 (2))
```

- Co se vytiskne?

# Rozsah platnosti

- **Lokální proměnná**
  - deklarujeme ji ve funkci,
  - při každém vstupu do daného bloku je proměnná vytvořena,
  - při opuštění funkce pozbývá proměnná platnosti.
- **Globální proměnná**
  - deklarujeme ji mimo funkce,
  - platnost je v celém modulu,
  - ve funkci můžeme vytvořit globální proměnnou pomocí příkazu global,
  - pozor na špatnou práci s globálními proměnnými.
- **Pravidlo LEGB**
  - identifikátor se hledá v pořadí:
    - lokální rozsah (local),
    - neukončená funkce (enclosing),
    - globální rozsah (global),
    - vestavěné názvy (built-in),
  - pokud není nalezen ani v jednom rozsahu -> chyba.

# Příklad

```
x=99

def fun():
    x=88
    print(f"Ve funkci je hodnota X={x}.")

fun()
print(f"Mimo funkci je hodnota X={x}.")
```

# Příklad

```
x=99

def fun():
    global x
    x=88
    print(f"Ve funkci je hodnota X={x}.")

fun()
print(f"Mimo funkci je hodnota X={x}.")
```

# Seznamy různých hodnot

# Vytvoření seznamu

- Dosud jsme pracovali pouze se seznamy jejichž prvky byly celá čísla. Nic nám nebrání udělat seznam z libovolných hodnot. Můžeme tedy vytvořit seznam řetězců:

```
['jedna', 'dve', 'tri']
```

- pravdivostních hodnot:

```
[True, False, True]
```

- nebo desetinných čísel:

```
[0.1, 12.4, 1e-20]
```

- Dokonce můžeme i vytvořit seznam, kde každý prvek je jiného typu:

```
['Hrác jedna', 45, True]
```

# Seznam seznamů

- Jelikož seznam je také hodnota, muže být i on prvkem jiného seznamu.

```
>>> l = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>>
>>> l[0]
[1, 2, 3]
>>> l[0][1]
2
```

# Příklad

- Napíšeme program tisknoucí matici (bez obklopujících závorek).

```
def print_cell ( element , is_last ):
    print ( element , end='')
    if not is_last :
        print (' ', end ='')

def print_row (row ):
    for j in range ( len(row )):
        element = row [j]
        is_last = j == len(row) - 1
        print_cell ( element , is_last )
    print ()

def print_matrix ( matrix ):
    for i in range ( len( matrix )):
        row = matrix [i]
        print_row (row)
```

# Úkol 2

- Je dán seznam čísel l a přirozené číslo n. Vytvořte seznam délky n, kde prvek na indexu i je seznam obsahující všechna čísla z l jejichž zbytek po dělení číslem n je i.
- Například pro seznam [1, 2, 3, 3, 8, 5, 4] číslo 3 vraťte [[3, 3], [1, 4], [2, 8, 5]].

- Řešení:

```
seznam=[1, 2, 3, 3, 8, 5, 4]
n=3

s=[]
for i in range(n):
    s.append([])

for i in seznam:
    s[i%n].append(i)

print(s)
```

# Úkol

- Vytvořte počáteční stav šachovnice u hry česká dáma.
- Šachovnici osm krát osm reprezentujte jako seznam řádku, kde řádek je seznam polí.
- Nulou označte prázdné pole, jedničkou bílý kámen a dvojkou černý kámen.

- Napište funkci, která vytiskne šachovnici. Prázdná pole tisknete znakem tečka, bílý kámen písmenem malé o, černý kámen hvězdičkou. Za znakem pole nechávejte mezeru.
- Doplňte z obou stran čísla řádků a písmen sloupců.

```
  a b c d e f g h
1 . o . o . o . o 1
2 o . o . o . o . 2
3 . o . o . o . o 3
4 . . . . . . . . 4
5 . . . . . . . . 5
6 . * . * . * . * 6
7 * . * . * . * . 7
8 . * . * . * . * 8
  a b c d e f g h
```