Języki Symboliczne, laboratorium 04

Zadanie 1. Napisz funkcję liczącą *sin(x)* oraz *cos(x)*, zwracającą obie te wartości naraz (krotka, *tuple*) [spoiler: strona 3]. Wypisz wyniki dla kątów od 0 do 90 stopni:

```
>>> sincos(0)
(0.0, 1.0)
>>> zad1()
kąt: 0; sin: 0.0; cos: 1.0
kąt: 9; sin: 0.15643446504023087; cos: 0.9876883405951378
kąt: 18; sin: 0.3090169943749474; cos: 0.9510565162951535
kąt: 27; sin: 0.45399049973954675; cos: 0.8910065241883679
kąt: 36; sin: 0.5877852522924731; cos: 0.8090169943749475
kąt: 45; sin: 0.7071067811865475; cos: 0.7071067811865476
kąt: 54; sin: 0.8090169943749475; cos: 0.5877852522924731
kąt: 63; sin: 0.8910065241883678; cos: 0.4539904997395468
kąt: 72; sin: 0.9510565162951535; cos: 0.30901699437494745
kąt: 81; sin: 0.9876883405951378; cos: 0.15643446504023092
kąt: 90; sin: 1.0; cos: 6.123233995736766e-17
```

W jakich jednostkach funkcje *sin* oraz *cos* przyjmują miary kątów?

https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences

Zadanie 2. Napisz funkcję przyjmującą dwa argumenty, zwracającą parę/krotkę: drugi argument oraz sumę obu argumentów. Wykorzystaj tę funkcję do napisania funkcji zwracającej zadany wyraz ciągu Fibonacciego. Wykorzystaj operator * do krotek:

```
>>> a=(1, 2, 3)
>>> print(a)
(1, 2, 3)
>>> print(*a, sep=';')
1;2;3
```

Zadanie 3. Napisz funkcję 'rozmywającą' listę. Funkcja powinna jako argument przyjmować listę i wagi, utworzyć kopię przesłaniej listy, której każdy element (poza pierwszym i ostatnim) jest zamieniamy na średnią ważoną siebie oraz swoich bezpośrednich sąsiadów, z przesłanymi wagami. Funkcja powinna zwrócić nową listę – przesłana jako argument nie powinna być modyfikowana. Utwórz listę 20 losowych liczb i zaprezentuj działanie powyższej funkcji.

```
>>> blur([3, 1, 2, 0, 4], (1, 1, 1))
[3, 2.0, 1.0, 2.0, 4]
>>> blur([3, 1, 2, 0, 4], (2, -1, 2))
[3, 3.0, 0.0, 4.0, 4]
```

Zadanie 4. Utwórz słownik *imię=>ocena*, zawierający oceny czterech przykładowych osób. Napisz funkcję przyjmującą jako argument słownik, wypisującą listę osób w kolejności alfabetycznej.
Napisz funkcję przyjmującą jako argument słownik, wypisującą listę osób wraz z ocenami w dowolnej kolejności.

https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Zadanie 5. Utwórz słownik słowników, *imię*=>(*kategoria*=>*ocena*), przechowujący listę osób, oraz ocen z różnych kategorii (kolokwium 1, odpowiedź ustna 2 itp).
Napisz funkcję przyjmującą jako argument powyższy słownik, liczącą średnią arytmetyczną wszystkich ocen danej osoby, wypisującą listę osób których średnia ocen jest niższa od 3.0.
Napisz funkcję wypisującą średnią ocen wszystkich osób z podanej kategorii (np. średnią ocen wszystkich osób za 'python 1' itd.).

```
>>> dziennik
{'Mietek': {'python 1': 4, 'lisp': 2}, 'Kościsława': {'python 1': 2, 'odpowiedź ustna': 2, 'lisp': 3}}
```

Zadanie 6. Utwórz słownik *imię*=>*wiek*. Napisz funkcję przyjmującą jako argument słownik, zwracającą słownik *imię*=>*długość snu*, wyliczającą zalecaną dzienną długość snu dla każdej z osób w przesłanym słowniku:

- Osoby powyżej 200 roku życia uznawane są za nieumarłych, którzy powinni udać się na sen wieczny,
- Osoby z niedodatnim wiekiem powinny zostać pominięte,
- Dla pozostałych: $dlugość\ snu = \dfrac{8 godzin}{\log_{10}(wiek\ w\ latach)}$

Powyższy wzór został zmyślony – proszę nie stosować się do niego.

Przykładowo:

```
>>> osoby
{"Kel'thuzad": 357, 'Franek': 34, 'Wybraniec': -7, 'Autor': 4, 'Janek': 20}
[illegible]
{"Kel'thuzad": inf, 'Franek': 5.223708867929046, 'Autor': 13.287712379549449, 'Janek': 6.148974294721926}
```

Wykorzystaj mechanizm *Dictionary Comprehensions* [spoiler: strona 4] oraz:

```
>>> print(4 if True else 6)
4
>>> print(5 if False else 6)
6
```

Zadanie 7. Utwórz tablicę dwuwymiarową wielkości 10 na 10 elementów, wypełnij ją wartościami *False* oraz w losowych 20 miejscach wartościami *True* (upewnij się, że wylosowane pole nie ma wartości *True*). Napisz program pozwalający użytkownikowi wybrać element tablicy (użytkownik ma wpisać dwie liczby całkowite *x*, *y*). Wybrane współrzędne powinny zostać zapisane w pewnym zbiorze (*set*) jako krotka (*x*, *y*). Jeśli użytkownik podał drugi raz te same współrzędne, powinna zostać wypisana informacja o tym (np. "Już tam strzelano"), a program powinien wrócić do wyboru współrzędnych. Jeśli podany element tablicy ma wartość *True*, wypisany powinien zostać tekst "Trafiony!", w przeciwnym razie "Pudło!". Program ma prosić o podanie współrzędnych do momentu zestrzelenia wszystkich wartości *True*.
Aby dodać element do zbioru można wykorzystać operator | (pionowa kreska, OR) lub metodę *add*. Jak usunąć element ze zbioru? Jak uzyskać różnicę lub część wspólną (przecięcie, iloczyn) dwóch zbiorów? [spoiler: strona 5]

https://docs.python.org/3/tutorial/datastructures.html#sets

ad. Zadanie 1. Utworzenie krotki, pobranie jednego elementu, utworzenie krotki jednoelementowej i pustej:

```
>>> a=(1, 2, 5)
>>> a
(1, 2, 5)
>>> a[1]
2
>>> a=4,
>>> a
(4,)
>>> a=()
>>> a
()
```

Krotki są niemodyfikowalne:

```
>>> a=(1, 'bla', lambda x: x**0.5)
>>> a
(1, 'bla', <function <lambda> at 0x03626858>)
>>> a[1]
'bla'
>>> a[1]='ble'
Traceback (most recent call last):
  File "<pyshell#64>", line 1, in <module>
    a[1]='ble'
TypeError: 'tuple' object does not support item assignment
```

Operator * można użyć do rozbicia krotki na kilka argumentów funkcji. Wartości krotki można też przypisać do kilku zmiennych:

```
>>> def funkcja(a, b, c):
        print(a, b, c)


>>> funkcja( *('krotka', 'jako', 'kilka argumentów') )
krotka jako kilka argumentów
>>> a, b, c=(1, 2, 5)
>>> a
1
>>> b
2
>>> c
5



>>> a=(1, 7)
>>> funkcja(*a)
Traceback (most recent call last):
  File "<pyshell#85>", line 1, in <module>
    funkcja(*a)
TypeError: funkcja() missing 1 required positional argument: 'c'
>>> a=(2, 1, 3, 7)
>>> funkcja(*a)
Traceback (most recent call last):
  File "<pyshell#87>", line 1, in <module>
    funkcja(*a)
TypeError: funkcja() takes 3 positional arguments but 4 were given
```

ad. Zadanie 6.

```
>>> slownik={'a': 1, 'b': 2, 'c': 4, 'd': 1337, 'e': -3}
>>> wynik={'nowy_'+key: value if value>0 else -value for key, value in slownik.items() if value<10}
>>> wynik
{'nowy_a': 1, 'nowy_c': 4, 'nowy_b': 2, 'nowy_e': 3}
>>>
```

Słownik *slownik* został wykorzystany do utworzenia nowego, którego klucze mają dopisany na początku tekst *'nowy_'*, złożony z wartości dodatnich (*value if value>0 else -value*), mniejszych niż 10 (*if value<10*).

ad. Zadanie 7. Użycie zbiorów:

1. Utworzenie zbioru *a*
2. Utworzenie zbioru *b*
3. Suma zbiorów (zbiór złożony z elementów zbiorów *a* i *b*)
4. Różnica zbiorów (zbiór *a* pozbawiony elementów występujących w *b*)
5. Różnica zbiorów (zbiór *b* pozbawiony elementów występujących w *a*)
6. Część wspólna zbiorów (elementy występujące w *a* i *b*)
7. Różnica symetryczna zbiorów (elementy występujące w *a* lub *b*, ale nie w obu jednocześnie) Innymi słowy, różnica sumy zbiorów i ich części wspólnej: *(a|b)-(a&b)*

```
>>> a={1, 2, 4, 7}
>>> b={2, 7, 8, 3}
>>> a|b
{1, 2, 3, 4, 7, 8}
>>> a-b
{1, 4}
>>> b-a
{8, 3}
>>> a&b
{2, 7}
>>> a^b
{1, 3, 4, 8}
```