**I.1) Co to jest lista, krotka, słownik i zbiór?**

Lista – czyli tablica, zmienne ustawione w szereg  
Krotka – tuple, podobnie jak tablica, ale ich zawartość jest zasadniczo stała (krotka może zawierać listę której elementy możemy edytować)  
Słownik – podobnie jak tablica, ale zamiast indeksów wykorzystuje klucze  
Zbiór – tablica w której elementy się nie powtarzają

**I.2) Jak się definiuje i wykorzystuje te obiekty?**  
Lista:  
 lista = [] lub list() #dodajemy elementy np przez .append()  
 lista = [1,2,3]  
Krotka:  
 dane = ("Aaa", 1, 2.13) #dane możemy tylko odczytać  
Słownik:  
 kontakty = {}  
 kontakty["Jan"] = 938477566 #odczytując klucz ‘Jan’ dostaniemy nr tel  
 .keys()–wypisze klucze słownika; .values()–wartości, .items()-to i to  
Zbiór:  
 A = set(["aaa", "bbb", "ccc"])

**I.3) Wyjątki i ich obsługa.**  
 try:

(polecenie)

except (nazwaWyjątku):

(polecenie gdy wystąpi wyjątek)  
  
**I.4) Operacje na plikach. Różnice między obsługą plików tekstowych, a binarnych.**

Zapisując plik binarnie musimy odpowiednio formatować to co chcemy zapisać, o wiele prościej jest skorzystać z modulu pickle  
 plik = open('ścieżka/do/pliku.txt', ‘(tryb)’)

#tryb – r-odczyt, w-zapis, a-dopis + t-tekstowy (domyślnie), b-binarny  
 plik.write( „coś” ) #tekstowo, tylko gdy zapis i dopis  
 print (plik.read()) #tekstowo, tylko gdy odczyt

plik.close()  
Inny sposób odczytu/zapisu linia po linii:

with open('plik.txt', ‘rt’) as plik: #odczyt linia po linii

for linia in plik:

print linia.strip().split()

**I.5) Serializacja i deserializacja danych przy zapisie do i odczycie z pliku binarnego (moduł pickle).**  
Serializacja: przekształcenie danych w strumień bajtów  
Deserializacja: proces odwrotny, zamiana strumienia bajtów w użyteczne dane  
  
 import pickle

output\_file = open("myfile.bin", "wb") #ustawienie do zapisu binarnego

myint = 42

mystring = "Hello, world!"

pickle.dump(myint, output\_file) #serializacja zapis inta jako bajty

pickle.dump(mystring, output\_file) #pickle.dump() do zapisu

output\_file.close()

input\_file = open("myfile.bin", "rb") #ustawienie do odczytu binarnego  
 myint = pickle.load(input\_file) #odczyt bajtów do zmiennej

mystring = pickle.load(input\_file) #pickle.load() do odczytu  
 input\_file.close()

**II.1 a) Krotki 1**  
 0: 1

1: 2.4

2: ABC

3: (3-2j)

4: (1, 3)

pierwszy element: 1 # 0 to pierwszy element

ostatni element: (1, 3) # -1 to pierwszy od końca element  
  
**II.1 b) Krotki 2** (('A', 'B'), (1, 2)) # cały pierwszy element krotki

(7.0, 600.0) # ostatni element (drugi) krotki

B # drugi element w pierwszej krotce pierwszej krotki

600.0 # czyli 6e2 = 6\*10^2 = 6\*100  
  
**II.2 a) Listy 1**  
 Długość listy imion: 10

Zawartość listy imion: ['P', 'a', 'w', 'e', 'ł', 'K', 'a', 's', 'i', 'a']  
  
**II.2 b) Listy 2**  
 Długość listy imion: 2

Zawartość listy imion: ['Paweł', 'Kasia']  
Dodanie nawiasów kwadratowych wpłynęło na potraktowanie tekstu jako całości do dodania do listy, wcześniej dodawano litera po literze.

**II.2 c) Listy 3, inne sposoby dodawania, usuwania z listy:**  
 imiona.append(imie1)  
 imiona.append(imie2)  
 imiona.pop(0) #usunie pierwszy element listy  
  
**II.3 a) Słowniki**  
 >>> s = {'Imie':'Józef', 'Nazwisko':'Nowak',5:13.2}

>>> s.keys()

dict\_keys(['Imie', 'Nazwisko', 5])

>>> s.values()

dict\_values(['Józef', 'Nowak', 13.2])

>>> s.items()

dict\_items([('Imie', 'Józef'), ('Nazwisko', 'Nowak'), (5, 13.2)])

>>> w = s.get('Imie')

>>> w

'Józef'

>>> s

{'Imie': 'Józef', 'Nazwisko': 'Nowak', 5: 13.2}

>>> w = s.pop(5)

>>> w

13.2

>>> s

{'Imie': 'Józef', 'Nazwisko': 'Nowak'}  
  
**II.3 b) Słowniki – dodawanie kluczy**  
 >>> s

{'Imie': 'Józef', 'Nazwisko': 'Nowak'}

>>> s["a"] = "b"

>>> s[5] = True

>>> s[2.33] = 3+2j

>>> s

{'Imie': 'Józef', 'Nazwisko': 'Nowak', 'a': 'b', 5: True, 2.33: (3+2j)}

**II.4 a) Zbiory**  
 >>> x = {'jeden','dwa','trzy'}  
 >>> y = {'dwa','trzy','cztery'}  
 >>> x.intersection(y)

{'dwa', 'trzy'}  
 >>> {'jeden','trzy'}.issubset(x)

True  
 >>> {'jeden','trzy'}.issubset(y)

False  
 >>> x|y

{'dwa', 'trzy', 'jeden', 'cztery'}

>>> x&y

{'dwa', 'trzy'}

>>> x^y

{'jeden', 'cztery'}

>>> x-y

{'jeden'}  
 >>> z={1,2,3,1,1,2,2,3,3,1,2,3,1,3,2,1,2,3}

>>> print('Długość:',len(z),'Zbiór:',z)

Długość: 3 Zbiór: {1, 2, 3}  
  
**II.4 b) Zbiory – usuwanie duplikatów z listy**  
 >>> lista = [1,2,3,4,5,6,1,2,3]

>>> zbior = set(lista)  
 >>> lista

[1, 2, 3, 4, 5, 6, 1, 2, 3]  
 >>> zbior

{1, 2, 3, 4, 5, 6}  
  
**II.5 a) Obsługa wyjątków** Podaj liczbę całkowitą: 1

Wprowadziłeś liczbę 1

Dziękuje!

= RESTART

Podaj liczbę całkowitą: 1.0

To nie jest liczba całkowita!

Dziękuje!  
  
**II.5 c) Obsługa wyjątków z ilorazem dwóch liczb**Zagnieżdżenie kolejnego try: z obsługą wyjątku ZeroDivisionError jeżeli podane liczby są poprawne.

**II.6 a) Operacje na plikach**  
 >>> f = open('plik1.txt', 'wt+') #otwarcie pliku tekstowego do w i r

>>> f.write('ABCDEFGHIJKLMNOP') #wpisanie danych do pliku

16

>>> f.tell() #sprawdzenie gdzie jest „karetka” w pliku

16

>>> f.seek(0) #przestawienie na karetki na pierwszą pozycje pliku

0

>>> f.read(6) #odczytanie 6 elementów od pozycji karetki w prawo

'ABCDEF'

>>> f.read() #odczytanie wszystkich elementów od karetki w prawo

'GHIJKLMNOP'

>>> f.write('''pierwszy #wpisanie wielolinijkowego tekstu do pliku  
 drugi

trzeci''')

23

>>> f.seek(0)

0

>>> f.readline() #odczytanie tekstu aż do \n (jednej linii)

'ABCDEFGHIJKLMNOPpierwszy\n'

>>> f.readline() #odczytanie kolejnej linii

'\tdrugi\n'

>>> f.seek(0)

0

>>> f.readlines() #odczytanie wszystkich linii pliku (zwraca tablice)

['ABCDEFGHIJKLMNOPpierwszy\n', '\tdrugi\n', '\ttrzeci']

>>> if not f.closed: f.close() #jeżeli plik nie zamknięty to zamknij  
  
**II.6 b) Operacje na plikach**  
  
 >>> with open('plik2.txt','wt') as f:

f.write('próba zapisu')

12

>>> f.closed

True

Ze względu na działanie wyrażenia with open() as f nie musimy zamykać pliku po wykonaniu działań na nim.