Laboratorul 1

În cadrul acestui laborator vom lucra cu elementele de bază folosite în limbajul de programare Python.

Variabile

```
>>> intreg = 1
>>> real = 7.2
>>> sirCaractere = "Primul laborator la ISupB"
>>> sirCaractere = 'Primul laborator la ISupB'
```

Afișarea unei variabile se face cu funcția print().

```
>>> print(intreg)
>>> print (sirCaractere)
```

Exerciții

1. Creați două variabile de tip întreg și afișați suma, diferența și produsul lor.
2. Creați două șiruri de caractere care conține textul "Bioinformatician" și "in devenire". Afișați conținutul șirurilor concatenate folosind operatorul +.
3. Creați o variabilă de tip întreg care conține o valoare și un șir de caractere care conține o propoziție. Afișați variabila și șirul de caractere concatenate.
4. Creați un șir de caractere care să conțină "Bioinformatician2017". Extrageți valoare 2017 din șir și atribuiți-o unei variabile de tip întreg. Extrageți "informatician" și atribuiți-l unui alt șir de caractere. Folosiți [1:3].

Liste

```
>>> lista  = []
>>> lista  = [1, 2, 3]
>>> lista  = [a, b, c]
>>> lista  = [genomica, proteomica]
```

Funcții aplicabile unei liste:

- ccmp(lista1, lista2) — Compară elementele celor două liste.
- len(lista) — Returnează numărul de elemente din listă.
- max(lista) — Returnează elementul din listă cu valoarea maximă.
- min(lista) — Returnează elementul din listă cu valoarea minimă.

Metode de lucru cu o listă:

>>> lista.append(element)	Inserează elementul la finalul listei.
>>> lista.count(element)	Returnează numărul de apariții al elementului în listă.
>>> lista.index(element)	Returnează indexul primei apariții a elementului în listă.
>>> lista.insert(index, element)	Inserează elementul la indexul indicat în listă.
>>> lista.pop()	Elimină și returnează ultimul element din listă.

>>> lista.remove(element)	Elimină elementul indicat din listă.
>>> lista.reverse()	Inversează ordinea elementelor din listă.
>>> lista.sort([funcție])	Sortează elementele din listă conform funcției indicate.

Exerciții

1. Creați o listă vidă.
2. Adăugați cinci elemente în listă folosind funcția append().
3. Folosiți funcția count() pentru a verifica că aveți cinci elemente.
4. Aflați pe ce poziție se află elementul 121 în listă.
5. Dacă elementul 121 nu este în listă atunci adăugați acest element în listă.
6. Inserați la poziția patru valoarea 23.
7. Afișați și eliminați ultimul element din listă.
8. Afișați toate elementele din listă.
9. Inversați ordinea elementelor din listă și afișați-le.
10. Sortați elementele și listă afișați-le.

Dicționare

```
>>> sala = {
        'Bioinformatica' : 31,
        'Proteomica' : 29,
        'Genomica' : 37
}
```

Funcții aplicabile unui dicționar:

- cmp(dictionar1, dinctinar2) Compară elementele celor două dicționare.
- len(dictionar) Returnează numărul de elemente din dicționar.
- str(dictionar) Produce o versiune afișabilă a dicționarului

Metode de lucru cu dicționarul:

>>> dictionar.clear()	Elimină toate elementele din dicționar
>>> dictionar.copy()	Returnează o copie a dicționarului
>>> dictionar.get(cheie, default=None)	Returnează valoarea conținută de dicționar în compartimentul "cheie", în ipoteza că există ceva acolo, iar în caz contrar, valoarea setată în mod implicit
>>> dictionar.has_key(cheie)	Returnează valoarea "True", dacă există o înregistrare în dicționar cu cheia "cheie", iar în caz contrar, returnează valoarea "False"
>>> dictionar.items()	Returnează o listă cu toate perechile de tipul (cheie, valoare) din interiorul dicționarului
>>> dictionar.keys()	Returnează o listă cu toate cheile dicționarului

>>> dictionar.setdefault(cheie, default=None)	Similară funcției get(), verifică existența cheii "cheie" în dicționar, iar în lipsa acesteia, o introduce cu valoarea implicită
>>> dictionar1.update(dictionar2)	Adaugă conținutul dicționarului "dictionar2" în dicționarul "dictionar1"
>>> dictionar.values()	Returnează o listă cu valorile regăsite în dicționar

Exerciții

1. Creați un dicționar.
2. Verificați dacă dicționarul are o anumită cheie care introdusă la punctul 1.
3. Verificați dacă dicționarul are cheia 121.
4. Extrageți valoare unei chei care a fost introdusă la punctul 1.
5. Afișați lista valorilor din dicționar.
6. Afișați lista cheilor din dicționar.
7. Creați un alt dicționar. Comparați cele două dicționare.
8. Adăugați valorile din al doilea dicționar în primul dicționar.

Structura decizională

Permite alegerea unei opțiuni dintre două disponibile, pe baza unui test logic

```
If (test):
        calea 1
else:
        calea 2
```

Structură repetitivă

Permite efectuarea unor operații de mai multe ori.

```
for i in range(1, 6):
        print("Aceasta este operatia care e repeta")

while (a < b):
        print("Aceasta este operatia care e repeta")
```

Exemple

În primul exemplu vom parcurge o secvență de ADN și vom afișa fiecare bază azotată de pe fiecare poziție.

```
adn = "ATGCTTACTGAAA"
for i in range(0, len(adn)):
        print(adn[i])
```

În următorul exemplu vom afișa toate pozițiile în care se găsește o adenină.

```
adn = "ATGCTTACTGAAA"
for i in range(0, len(adn)):
```

```
    if (adn[i] == 'A'):
        print(i)
```

În următorul exemplu vom înlocui toate bazele de timină (T) cu litera X iar dacă nu este timină atunci le vom înlocui cu semnul "-".

```
adn = list("ATGCTTACTGAAA")
for i in range(0, len(adn)):
    if (adn[i] == 'T'):
        adn[i] = 'X'
    else:
        adn[i] = '-'
print(''.join(adn))
```

Exerciții

1. Creați o structură repetitivă care parcurge o secvență de ADN și creează pe baza acesteia o secvență de ARN. ARN-ul se formează prin concatenarea bazelor complementare secvenței de ADN și prin schimbarea timinei cu uracilul. (A -> U, T -> A, C -> G, G -> C). Exemplu: ACTTAG -> UGAAUC.
2. Creați o buclă repetitivă care transformă o secvență de pe catena negativă într-o secvență pe catena pozitivă (inversa-complementată, reverse-complement). Această operație presupune complementarea bazelor azotate (A -> T, T -> A, C -> G, G -> C) și inversarea elementelor astfel încât ultimul element să devină primul. Exemplu: ACTGGA -> TGACCT -> TCCAGT.
3. Creați o aplicație care transformă o secvență de ADN într-o secvență de aminoacizi. Exemplu: ATCGAGTCC -> IES.

Amino Acid	single letter code	3-letter code	DNA codons
Isoleucine	I	Ile	ATT, ATC, ATA
Leucine	L	Leu	CTT, CTC, CTA, CTG, TTA, TTG
Valine	V	Val	GTT, GTC, GTA, GTG
Phenylalanine	F	Phe	TTT, TTC
Methionine	M	Met (start)	ATG
Cysteine	C	Cys	TGT, TGC
Alanine	A	Ala	GCT, GCC, GCA, GCG
Glycine	G	Gly	GGT, GGC, GGA, GGG
Proline	P	Pro	CCT, CCC, CCA, CCG
Threonine	T	Thr	ACT, ACC, ACA, ACG
Serine	S	Ser	TCT, TCC, TCA, TCG, AGT, AGC
Tyrosine	Y	Tyr	TAT, TAC
Tryptophan	W	Trp	TGG
Glutamine	Q	Gln	CAA, CAG
Asparagine	N	Asn	AAT, AAC
Histidine	H	His	CAT, CAC
Glutamic acid	E	Glu	GAA, GAG
Aspartic acid	D	Asp	GAT, GAC
Lysine	K	Lys	AAA, AAG
Arginine	R	Arg	CGT, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	termination	TAA, TAG, TGA

Sursă: http://fourier.eng.hmc.edu/bioinformatics/intro/node7.html