**Practical 5**

**Functions**

1. Գրեք ֆունկցիա, որը ստանում է 3 integer-ներ որպես argument-ներ ու վերադարձնում է (return) իրենց միջին արժեքը:

2. Գրեք ֆունկցիա, որը ստանում է integer-ների list որպես argument ու վերադարձնում է (return) տվյալ լիստում զույգ թվերի քանակը:

3. Գրեք ֆունկցիա, որը ստանում է **password** String-ը որպես argument ու ստուգում է արդյոք այն հանդիսանում է կանոններին համապատասխանող գաղտնաբառ՝ վերադարձնելով համապատասխանաբար True կամ False:

   Գաղտնաբառի կանոնները՝
   Գաղտնաբառի երկարությունը պետք է լինի առնվազն 10
   Գաղտնաբառը պետք է պարունակի առնվազն 2 թիվ (փորձեք ինքնուրույն գտնել թե ինչպես կարելի է անել այս ստուգումը)

4. Գրեք ֆունկցիա, որը ստանում է 1 պարտադիր string տիպի argument **name** ու 1 ոչ պարտադիր argument **greeting**, որի default արժեքն է “Welcome to our company!”: Ֆունկցիան տպում է “**name**, **greeting**”՝ օգտագործելով **name** և **greeting** argument-ների արժեքները:

5. Գրեք ֆունկցիա, որը վերցնում է 1 պարտադիր string տիպի argument **name** ու չորոշված քանակով ոչ պարտադիր ոչ-keyword argument-ներ: Ֆունկցիան պետք է հաշվի ոչ պարտադիր argument-ների միջին արժեքը (եթե տրված է որևէ ոչ պարտադիր argument) ու տպի հետևյալը՝եթե տրված է որևէ ոչ պարտադիր argument տպեք “**X**, your average grade is: **Y**”, որտեղ **X**-ը **name** argument-ի արժեքն է, իսկ Y-ը ոչ պարտադիր argument-ների միջին արժեքը: Հակառակ դեպքում տպեք “No grades availble for **X**”, որտեղ **X**-ը **name** argument-ի արժեքն է:

6. Գրեք ֆունկցիա, որը վերցնում է 1 պարտադիր string տիպի argument **user** ու չորոշված քանակով ոչ պարտադիր keyword argument-ներ: Եթե **user** argument-ի արժեքը “admin” է, Ֆունկցիան պետք է տպի բոլոր ոչ պարտադիր keyword argument-ները հետևյալ ֆորմատով՝
   “Argument1 name”: “argument1 value”
   “Argument2 name”: “argument2 value”
   “Argument3 name”: “argument3 value”
   և այլն:

Հակառակ դեպքում, եթե **user** argument-ի արժեքը “admin” չէ, ֆունկցիան պետք է տպի “access denied to the user X”, որտեղ X-ը **user** argument-ի արժեքն է:

## Modules

7. Ստեղծեք **calc.py** file-ը, որը պարունակում է 2 ֆունկցիա՝
   **calculate_cube(x) -** որն ընդունում է integer **x**-ը որպես argument ու վերադարձնում է x^3;
   **calculate_square(x) -** որն ընդունում է integer **x**-ը որպես argument ու վերադարձնում է x^2:
   Ստեղծեք **pretty_print.py** file-ը, որը պարունակում է 2 ֆունկցիա՝
   **simple_print(x)** - որն ընդունում է integer **x**-ը որպես argument ու տպում է “Result: **x**”, որտեղ **x**-ը **x** argument-ի արժեքն է;
   **pro_print(x)** - որն ընդունում է integer **x**-ը որպես argument ու տպում է “The result of the operation is **x**”, որտեղ **x**-ը **x** argument-ի արժեքն է:
   Դուք պետք է ունենաք Ձեր **main()** ֆունկցիան **calc.py** ֆայլի մեջ: **main** ֆունկցիայի ներսում պետք է հաշվեք 2-ի քառակուսին՝ օգտագործելով համապատասխան ֆունկցիա ու տպեք արդյունքը օգտվելով **simple_print(2)** ֆունկցիայից, ինչպես նաև հաշվեք 4-ի խորանարդը՝ օգտագործելով համապատասխան ֆունկցիա ու տպեք արդյունքը օգտվելով **pro_print(4)** ֆունկցիայից:

## Homework 5

1. Ստեղծեք **my_max** ֆունկցիան, որը ստանում է անորոշ քանակի թվեր որպես ոչ-keyword արգումենտներ, գտնում է այդ թվերի maximum-ը և վերադարձնում է այն: Եթե ֆունկցիան կանչվում է առանց արգումենտներ, այն պետք է վերադարձնի “no numbers given” արժեքը: Այս առաջադրանքը կատարելիս մի օգտագործեք առավերագույն արժեք հաշվելու build-in ֆունկցիաներ:

2. Ստեղծեք ֆունկցիա, որը ստանում է որպես արգումենտ list ու վերադարձնում է նոր list, որը պարունակում է տրված list-ի չկրկնվող (unique) արժեքները:

3. Ստեղծեք **my_fib** ֆունկցիան, որը ստանում է integer **n** որպես արգումենտ և վերադարձնում է n-րդ ֆիբոնաչիի թիվը՝ առանց օգտվելու ռեկուրսիայից:
   Նշում՝ fib(n) = fib(n-1) + fib(n-2)

4. Ստեղծեք **my_func** ֆունկցիան, որը ստանում է մեկ integer տեսակի արգումենտ **n** և վերադարձնում է **n+10, եթե n<10** կամ վերադարձնում է **n-10`** հակառակ դեպքում: Ստեղծեք ձեր ցանկացած integer արժեքներով **list1** լիստը: Օգտագործեք **map()** ֆունկցիան և կիրառեք **my_func** ֆունկցիան **list1**-ի ամեն արժեքի վրա՝ ստանալով նոր լիստ:

5. Ստեղծեք **Productcheck.py** ու **Customer.py** module-ները հետևելով ցուցումներին՝

**Productcheck.py**

Ֆունկցիաներից դուրս, ֆայլի սկզբում ստեղծեք հետևյալ dictionary-ն՝ products = {“candy”: 10, “juice”: 5, “pen”: 50}

**check(product, num):** Ստանում է պատվիրված ապրանքի անուն ու քանակ, ստուգում է թե արդյոք այդ քանակով ապրանք առկա է (ստուգելով թե արդյոք products dictionary-ում առկա է ապրանքի անունով key ունեցող ու արդյոք այդ key-ի տակ պահված արժեքը >= տրված ապրանքի քանակից (**num**)) ու վերադարձնում է համապատասխանաբար True կամ False:

**Customer.py**

**buy(product, num, price):** Ստանում է պատվիրված ապրանքի անուն ու քանակ, ստուգում է թե արդյոք այդ ապրանքն այդ քանակությամբ առկա է՝ օգտագործելով **Productcheck** module-ի **check(product, num)** ֆունկցիան: Եթե ապրանքն առկա է ապա տպում է “You bought ***product*** and spent ***num*price***”, հակառակ դեպքում տպում է “Sorry! We are out of this product.”:

**main():** Ֆունկցիայում պետք է կանչվի **buy(product, num, price)** ֆունկցիան որոշակի արժեքներով: