00_MODULES

Ինչպես որ հուշում է անվանումը դիրեկտորիան պարունակում է հավելվածի աշխատանքը և ֆունկցիոնալությունը ապահովող հիմնակ մոդուլները:

- **database** - այն կկապվի ՏԲ հետ և կկարդա ինֆորմացիան այնտեղից:
- **commands** - կապում է մի քանի մոդուլները միասին, կատարում կոլեկտիվ-ընդհանուր լոգիկան: Հանդիսանում է կապող օղակ:
- **excel_table** - կկատարի ստացված տվյալների վերափոխումը էկզել սանդղակի: Սանդղակը կգեներացնելով աշխատանքի ավարտից հետո: նախատեսված վերջնական օգտատիրոջ համար:
- **factory** - այն ունի Ինֆորմատիկ-ճարտարագիտական նշանակություն, տարանջատոլու համար ապրանքի տեսակը և դրա հաշվարկելու լոգիկան:

database

Նպատակը և Առաջադրված խնդիրներ

> Մոդուլի այս հատվածը հավելվածի աբստրակցիայի **առաջին** կամ եթե կարելի է այդպես ասել հավելվածի շրջանակներում
> "ամենացածր" մակարկան է, քանի որ *անմիջականորեն կապ է ստեղծում տվյալների բազզայի հետ* ապահովելով
> ամբողջ հավելվածի աշխատանքը վերջինիս հետ:

ՆԵՐՄՈՒԾՎԱԾ ԳՐԱԴԱՐԱՆՆԵՐ

> sqlite3, shelve, abc

ԴԱՍԵՐ ԵՎ ԴՐԱՆՑ ՆԵՐԿԱՌՈՒՑՎԱԾ ՖՈՒՆԿՑԻԱՆԵՐԸ:

`DataBaseManager` | ԴԱՍԻ ՆՇԱՆԱԿՈՒԹՅՈՒՆԸ

Դասը աբստրակտ է այն հանդիսկանում է բազզային դաս որում ռեալիզացված են նրա ժառանգ դասերի համար պարտադիր համարվող ֆունցիաները: Իր գոյությամբ այն երաշխավորում է որ, ոչ-մի ժառանգ բաց չի թողնի պարտադիր համարվող ֆունցիաների ռեալիզացումը: Այստեղ այդպիսի ֆունկիցաներ են հանդիսանում **_connection write_to** ֆունկցիաները: Սակայն կարելի է նկատել որ, ֆունկցիաների ռեալիզացիան յուրաքանչյուր դասում յուրօրինակ է, տվյալ ՏԲ կարիգի և նշանակությանը համաձայն: Պարտադիր համարվող ֆունկցիաները դեկորացված են **@abstarctmethod**-ով:

`SQLDataBase` | **SQL** ՏՎՅԱԼՆԵՐԻ ԲԱԶԶԱ

Ժառայնգ դաս՝ Այն ապահովվում է **sqlite3** ՏԲ հետ անխափան աշխատանքը:

SQLDataBase.`_connection(str, values=False)` | ԿԱՊԻ ՀԱՍՏԱՏՈՒՄ:

Այն կկատարի ապահով կապի հաստատում ՏԲ հետ, կապահովվի տվյալների մուտքագրումը ՏԲ, դրանց հարցումը հարկ եղած դեպքում, ստացված տվյալենրի համար ֆոմատավորումը(dict, tuple փոխհարեն) իսկ վերջում կվերադարձնի կատիվ կուրսոր: Այն ունի մեծ նշանակություն ՏԲ հետ աշխատանքի ժամանակ, այս մոդուլի ներքոհիշյալ ֆունցիաները (մասնավորապես **select** և **add_to**) անիջականորեն օգտվում են այս ֆունկցիայից:

SQLDataBase.`select(str, tare=False)` | ՏՎՅԱԼՆԵՐԻ ՀԱՐՑՈՒՄ:

Այս ֆունկցիան տրամադրում է տվյալենրի հարցման շաբլոններ ծրագրում պլանվավորված հարցման բոլոր սցենարների համար, ըստ տրված լոգիկայի պայմաններում: Այն ընդունում է բարկոդ և կկատարի հարցում ստացված բարկոդի տվյալների համաձայն: Վերջում կվերադարձնի ստացած տվյալները, դրանց առյաության պարագայում Հակառակ դեպքում՝ None կամ Null:

SQLDataBase.`wirite_to(dict, tare=False)` | ՏՎՅԱԼՆԵՐԻ ՄՈՒՏՔԱԳՐՈՒՄ:

Ֆունկցիան օգտակար է այն պարագայում երբ օգտատերը բախվել այնպիսի իրավիճակի որ, ապրանքատեսակը առկա չէ ցուցադրված ՏԲ-ում և պետք է ավելացնել այն կամ անհրաժեշտ փոխել տվյալների առկա ցուցանիշները: Այն ապահովվում է տվյալների մուտքագրումը կամ դրանց փոփոխությունը ՏԲ-ում: Այն ընդունում է տվյալները մեկ հասանելի ֆորմատով բառարանի տեսքով(dict) և պատասխանատու է դրանց ապահով մուտքագրման համար ՏԲ:

`ShelveDataBase` | ԴԱՍԻ ՆՇԱՆԱԿՈՒԹՅՈՒՆԸ

Ժառայնգ դաս՝ Այն ապահովվում է **shelve** սերիալիզացիոն նշանակություն ունեցիող գրադարանի հետ անխափան աշխատանքը: Այն ունի բուֆֆերային նշանակություն, ստեղծվում է ժամանակավոր ֆայլ որում գրացնվում են օգտատոիրոջ բոլոր հարցումները և արդյունքում ստացված զտաքաշի տվյալները ծրագրի ավարտի դեպքում դրանցից **excel** ֆայլ գեներացնելու համար:

ShelveDataBase.`_connection()` | ԿԱՊԻ ՀԱՍՏԱՏՈՒՄ:

Այն կապ է հաստատում իսկ բացակայության դեպքում ստեղծում է **buffer.db** ժամանակավոր տվյալների պահոց-ֆայլը:

ShelveDataBase.`wirite_to(str, float)` | ՏՎՅԱԼՆԵՐԻ ՄՈՒՏՔԱԳՐՈՒՄ:

Ֆունցիան կապ կհաստատի **buffer.db** հետ և ստացված տվյալները կգրանցի այնտեղ, վերջում փակելով այն տվյալների կորուստից խուսափելու համար:

factory

Նպատակը և Առաջադրված խնդիրներ

> Մոդուլի այս հատվածը առաջարկում է ապրանքի զտաքաշը հաշվարկելու համապատասխան լոգիկա կախված վերջինիս տեսակից: Խմիչքի դեպքում այն ընդհանուր քաշից հանում է տարայի քաշը ապա մնացորդը բաժանում խտության վրա իսկ այլ ապրանքի դեպքում այն ընդհանուր քաշից կհանի միայն տարայինը: Ծրագրում այս երկու լոգիկան տարանջատելու համար էլ կիրառվում է այս մոդուլը որը ներկայացնում է իրենից ինֆորմատիկ-ճարտարագիտական պատտերն որը, կոչվում է factory, այստեղից էլ մոդուլի անունը:

ՆԵՐՄՈՒԾՎԱԾ ԳՐԱԴԱՐԱՆՆԵՐ

> shelve, colorama
> include.validations, include.symbols

ԴԱՍԵՐ ԵՎ ԴՐԱՆՑ ՆԵՐԿԱՌՈՒՑՎԱԾ ՖՈՒՆԿՑԻԱՆԵՐԸ:

`Factory` | ԲԱԶԱՅԱԿԱՆ ԴԱՍ

Այս դասը ներկայացնում է իրենից բազմայական դաս, այն նկարագրում է ֆունկցիաների հիմնական բաղադրիչները որոնք պետք է կիրառվեն իրենից ժառանգող երկու՝ **PowderFactory, DrinkFactory** դասերը:

Factory. `calculation()` | ՀԱՇՎԻՉ

Դասի շրջանակներում այս ֆունկցիայի ռեալիզացիան իրենից ներկայացնում է *տեղապահ*(placeholder) ֆունկցիա, այսինքն այն երաշխավորում է որ, այս դասի յուրաքանչյուր ժառանգ պետք է ունենա այս ֆունկցիայի ռեալիզացիան իր մեջ, հակառակ դեպքում կհանգի ինտերպրետատորի **NotImplementedError** սխալի: Նաև տարբերակ է օգտագործել աստեղ abc գորադարանի **@abstarctmethod** դեկորատորից սակայն այս մեթոդը նույնպես ընդունելի է:

Factory. `full_weight_validation()` | ՔԱՇԻ ՎԱԼԻԴԱՏՈՐ

ֆունկցիան իր հերթից կանչ է կատարում *full_weight* ֆունկցիային որը որը երաշխավորում է որ օգտատիրոջ կողմից մուտքագրված քաշի տվյալները հանդիսանում են ամբողջական թվիային արժեք: Այս կպահանջի մուտքագրել վայլիդային արժեք այնքան ժամանակ մինչև այն կմուտքագրվի ճիշտ կամ կկատարվի ծրագրի ավարտ:

Factory. `string_output_design(float, str)` | ՌԵՆԴԵՐԻՆԳ

Ֆունկցիան ունի ավելի վիզուալ նշանակություն: Այն ստանում է ապրանքի քաշի ցուցանիշը և չափման միավորը(գրամ կամ միլիգրամ) և այն գեղեցիկ և ընթերնելի ձևով ներկայացնում է տերմինալում:

`PowderFactory` | ԽՄԻՉՔՆԵՐԻ ԴԱՍ

Դասը պատասխանատու է բոլոր հեղուկ չհանդիսացող ապրանքների (այսինքն երբ պետք չէ նաև հաշվի առնելհեղուկի խտություն)հետ մանիպուլացիաների համար:

PowderFactory. `calculation()` | ՀԱՇՎԻՉ

Ֆունկցիան կատարում է պարզագույն մաթեմատիկական հաշվարկ, այն օգտակար է այն ապրանքների պարագայում երբ օգտագործված մի քանի տեսակի ապրանքներ, սակայն դրանց տարաների քաշը միանման են: Այս պարագայում ծրագիրը մեկ սկսպասի բարկոդի մուտքագրում, իսկ մնացած դեպքերի համար պահպանելով տարայի քաշը կսպասի միայն ընդանուր քաշի ցուցանիշին, մինչև ծրագրի ավարտը: Քանի որ այս տեսակի ապրանքը օժտված չէ յուրօրինակ անվանումներով դրանք չեն գրանցվում վերջնական *excel* սանդղակի մեջ:

DrinkFactory. `calculation()` | ՀԱՇՎԻՉ

Դասը պատասխանատու է բոլոր խմիչքների հետ մանիպուլացիաների համար:

DrinkFactory. `calculation()` | ՀԱՇՎԻՉ

Այս ֆունկցիան վերոհիշվյալ համանուն ֆունցիայից տարբերվում է նրանով որ, հաշվարկ կատարելիս այն նաև տարաից բացի հաշվի է առնում կոնկրետ հեղուկի խտությունը, մնացորդը բաժանելով դրա վրա: Հաջորդիվ այն ստացված զտաքաշը և ապրանքի անունը գրանցում է բուֆերային ֆայլի մեջ որից էլ հետագայում պետք է գեներացվի *excel* սանդղակը:

commands

Նպատակը և Առաջադրված խնդիրներ

> Ավանդաբար այս մոդուլը համարվում է կապող օղակ, այդ ծրագրի պայմաններում այն կապում է նախորդող *database factory* մոդուլները բերելով դրանց աշխատանքը մեկ ընդհանուր լոգիկայի: Մոդուլը առաջարկում է ամբողջ ծրագիրի հավաքական ֆունկցիոնալը և լրացնում այն մասերը որոնք պակասում էին իր նախորդդների նեջ:

ՆԵՐՄՈՒԾՎԱԾ ԳՐԱԴԱՐԱՆՆԵՐ

> include.modules.factory, include.modules.database, from include.symbols, include.validations

ԴԱՍԵՐ ԵՎ ԴՐԱՆՑ ՆԵՐԿԱՌՈՒՑՎԱԾ ՖՈՒՆԿՑԻԱՆԵՐԸ:

`Command` | ՀԱՄԱԿԱՐԳՈՂ ԴԱՍ

Մոդուլի ամբողջ նկարարգրությունը վերաբերում է այս դասին քանի որ, այն իր վրա է վերձնում մոդուլի ամբողջ ֆունկցիոնալությունը:

Command. `__create(object)` | ՖԱԲՐԻԿԱ

Ֆունցիան ստանում է factory դասի օբյեկտ ըստ նրան տրված օբյեկտի տվյալների: տարբերություն չկա այն կստանա **PowderFactory**, թե **DrinkFactory** դասի օբյեկտը: Օգտվելով նրանից որ, այդ դասերը ունեն նույն ինտերֆեյսը այսինքն երկուսն էլ ունեն *calculation* ֆունկցիան: Ֆունկցիան կվերադարձնի այն:

Command. `filter_(db=object)` | ՏՎՅԱԼՆԵՐԻ ՖԻԼՏՐԱՑԻԱ

Այն նախ ստուգում է բարկոդի վալիդությունը, ապա ըստ բարկոդի հարցում է կատարում ՏԲ: Եթե արդյունքը դրական է այն **define*factory()* ֆունկցիայի օգնությամբ որոշում է թե տվյալները ինչ ֆաբրիկայի են պատկանում, ստեղծում է այդ դասի օբյեկտը և փոխանցում է այն** _create()_ ֆունցիային: Հակառակ դեպքում բացասական պատասխանի դեպքում, զգուշացնում է *__not_found_handle()* ֆունկցիայի օգնությամբ այդ մասին և առաջարկում նորից մուտքագրել:

Command. `_get_by_barcode(object, str)` | ՀԱՐՑՈՒՄ ՏԲ

ֆունկցիան ընդունում է բարկոդը և ՏԲ հացում է կատարում, դիական պատասխանի դեպքում վերդարձնում է տվյալների բառարան հակառակ դեպքում None:

Command. `__not_found_handle()` | ՍԽԱԼԻ ԱՐՏԱԲԵՐԻՉ

Ֆունցիայի ուղղակի տպում է *Սխալ Բարկոդ | Բարկոդը չի գտնվել* հաղորդագրությունը:

Command. `_define_factory(dict)` | ՖԱԲՐԻԿԱՅԻ ՈՐՈՇԻՉ

Ֆունկցիայի ընդունում է տվյալների բառարանը որը ստացել էինք և ըստ նրա **type** բանալու, որոշում թե որ ֆաբրիկային է այն պատկանում վերադարձնելով համապատասխան ֆաբրիկայի օբյեկտը զետեղելով նրա մեջ ստացված բառարանը:

excel_table

Նպատակը և Առաջադրված խնդիրներ

> Մոդուլի նպատակն է բուֆերային տվյալներից գեներացնել վերջնական excel ֆայլը:

ՆԵՐՄՈՒԾՎԱԾ ԳՐԱԴԱՐԱՆՆԵՐ

> openpyxl, datetime
> include.modules.database

ԴԱՍԵՐ ԵՎ ԴՐԱՆՑ ՆԵՐԿԱՌՈՒՑՎԱԾ ՖՈՒՆԿՑԻԱՆԵՐԸ:

`ExcelTable` | **DESCRIPTION**

Դասը կնախապատրաստի excel ֆայլը, ստեղծի դրա նախնական կմախքը, և բուֆերային ֆայլից տվյալները կտեղափոխի դրա մեջ պահպանելով այն::

ExcelTable.`convert_and_save(str, values=False)` | **DESCRIPTION**

Ֆունկցիան ակտիվ excel ֆայլում կավելացնի "Անվանում", "Զտաքաշ" սանդղակները: Կկապվի բուֆֆերային ֆայլի հետ, կստուգի արդյ՞ք այն դատարկ չէ, եթե դատարկ է ապա տվյալներ կարդալու կամ պահպանելու կարիգ չկա: եթե ֆայլը պարունակում է առնվազն մեկ տվյալ ապա այն կկարդա տվյալները և կմուտքագրի համապատասխանող սանդղակներում: Վերջում կպահպանի մուտքագրված տվյալները excel ֆայլում, տալով նրան արդիական ամսաթիվը պարունակող անուն:

01_INCLUDE

Դիրեկտորիան իր մեջ պարունակում է օգտակար ֆունկցիաներ պարունակող մոդուլներ որոնք սակայն չեն մասնակցում ծրագրի հիմնական լոգիկայի կատարմանը այլ ունեն հավելյալ օգնող են կամ վիզուալ ֆունկցիաներ են կատարում

- **validations** - պարունակում է վալիդացնող ֆունկցիաներ:
- **symbols** - պարունակում է UTF-8 վիզուալ emoji սիմվոլներ:

validations

Նպատակը և Առաջադրված խնդիրներ

> ֆայլը պարունակում է վալիդացիոն ֆունկիցաներ որոնք ստուգում են առաջին մուտքագրվող քաշի թվային արժեքը իսկ երկրորդ բարկոդի թույլատրելի երկարությունը և սիմվոլները:

ՆԵՐՄՈՒԾՎԱԾ ԳՐԱԴԱՐԱՆՆԵՐ

> colorama

> include.symbols

ՖՈՒՆԿՑԻԱՆԵՐ

`full_weight()` | **ՔԱՇԻ ՎԱԼԻԴԱՑԻԱ**

Ֆունցիան առաջարկում է մուտքագրել ընդանուր քաշի տվյալները և ստուգում որ դրանք լինեն թվային արժեք, եթե չեն բավարարվել պահանջները կառաջարկի նորից մուտքագրել այնքան ժամանակ մինչև որ, ստանա բավարարող պատասխան:

`check_barcode_length()` | **ԲԱՐԿՈԴԻ ՎԱԼԻԴԱՑԻԱ**

Ֆունցիան առաջարկում է մուտքագրել բարկոդի տվյալները և ստուգում որ դրանք լինեն թվային արժեք և լինին ոչ պակաս քան տանսըմեկ նիշ, եթե չեն բավարարվել պահանջները կառաջարկի նորից մուտքագրել այնքան ժամանակ մինչև որ, ստանա բավարարող պատասխան:

symbols

Նպատակը և Առաջադրված խնդիրներ

> ֆայլը պարունակում է UTF-8 վիզուալ emoji սիմվոլներ համրմար ֆորմատավորմամբ փոփոխականներ որոնք կիրառվում են ամբողջ ծրագրի տարբեր մոդուլներում:

ՆԵՐՄՈՒԾՎԱԾ ԳՐԱԴԱՐԱՆՆԵՐ

>

ՖՈՒՆԿՑԻԱՆԵՐ

`print_centered(str)` | **ԹԵՔՍՏԻ ԿԵՆՏՐՈՆԱՑՈՒՄ**

Ֆունկցիան ստանում է տեքստային արժեք և գեղեցիկ ֆորմատավորել տեղադրում համեմատական կենտրոնում: