Python 2025.10. Forgalomszámlálás

Egy város közlekedéstervezési céllal kerékpárosforgalom-számlálót telepít az egyik főútjára. A próbaüzem reggel 6-tól 10-ig tart, amelynek során 15 percenként rögzítik a megelőző 15 percben áthaladó kerékpárosok számát. A rendszer még bizonytalan, ha technikai probléma – például áramszünet – történik a mérés során bármikor, akkor a rögzített érték abban az időintervallumban -1 lesz, különben az áthaladók száma nemnegatív egész szám.

A rögzített mérési értékek száma 16, és értékük például a következő:

```
36, 48, 39, -1, 30, 43, -1, 76, 67, 82, 73, 75, 64, 73, 69, 63
```

A mérés kezdőadatai: 6:15-kor 36, 6:30-kor 48 kerékpáros, és így tovább. 7:00-kor és 7:45-kor -1 került rögzítésre, mert az előző negyedórában mérőhiba történt a számlálórendszerben. Az utolsó adatot, 63 kerékpárost 10:00-kor jegyezték fel.

Készítsen programot, amely megválaszolja a mérési eredményekre vonatkozó kérdéseket!

A program forráskódját mentse *szamlalas* néven! A program megírásakor a mérési adatok számát és helyességét nem kell ellenőriznie. A programnak akkor is helyesen kell működnie, ha a programban tárolt adatokat más, megfelelő adatokra cseréljük.

A képernyőre írást igénylő részfeladatok esetén az ékezetmentes kiírás is elfogadott. A mintához tartalmában hasonlóan írja ki a képernyőre a feladat sorszámát (például: `2. feladat`), valamint utaljon a kiírt tartalomra is!

BLZS©

A feladat leírása

BLZS©

1. A megadott 16 számot tárolja el a program forrásában egy megfelelő adatszerkezetben! A 16 szám rendelkezésre áll a *meres.txt* állományban, amelyből a program kódjába átmásolhatók.
2. A forgalomszámláló adatai alapján határozza meg az áthaladt összes kerékpáros számát, és írassa ki a minta szerint! Ügyeljen arra, hogy a számítás során a mérőhibás adatok ne befolyásolják az összeget!
3. Írassa ki a képernyőre a mintának megfelelően, hogy óránként hány kerékpáros haladt át a számlálón!
4. Határozza meg a legnagyobb mérési értéket és rögzítésének időpontját! Az eredményt írassa ki a mintának megfelelő formátumban! Több maximális érték esetén az elsőt jelenítse meg! A feladat szempontjából most egyformán helyesnek tekintjük a 07:00, 7:00, 7:0 írásmódot is.

Minta a szöveges kimenet kialakításához:

```
2. feladat
Összesen 838 kerékpárost számoltak.

3. feladat
Óránkénti mérések:
6 órától 123 kerékpáros
7 órától 149 kerékpáros
8 órától 297 kerékpáros
9 órától 269 kerékpáros

4. feladat
Az áthaladók maximális száma: 82; a rögzítés időpontja: 8:30.
```

BLZS©

Az elkészített program

BLZS©

```python
# 1. FELADAT: Adatok eltárolása
meresek = [ 36, 48, 39, -1, 30, 43, -1, 76, 67, 82, 73, 75, 64, 73, 69, 63 ]

# Segédlet a 4. feladathoz kezdőknek:
# Mivel csak 16 időpontunk van, a legkönnyebb, ha ezeket is egyszerűen felsoroljuk!
idopontok = [ "6:15", "6:30", "6:45", "7:00",  "7:15", "7:30", "7:45", "8:00",
    "8:15", "8:30", "8:45", "9:00",  "9:15", "9:30", "9:45", "10:00" ]

# 2. FELADAT: Összes kerékpáros
print("2. feladat")
osszesen = 0
for adat in meresek:
    if adat != -1:
        osszesen = osszesen + adat

print(f"Összesen {osszesen} kerékpárost számoltak.")

# 3. FELADAT: Óránkénti mérések
print("3. feladat")
print("Óránkénti mérések:")

aktualis_index = 0

for ora in [6, 7, 8, 9]:
    ora_osszege = 0
    for lepes in range(4):
        adat = meresek[aktualis_index]
        if adat != -1:
            ora_osszege = ora_osszege + adat
        aktualis_index = aktualis_index + 1

    print(f"{ora} órától {ora_osszege} kerékpáros")

# 4. FELADAT: Legnagyobb mérési érték
print("4. feladat")
max_ertek = -1
max_index = 0

for i in range(len(meresek)):
    if meresek[i] > max_ertek:
        max_ertek = meresek[i]
        max_index = i

print(f"Az áthaladók maximális száma: {max_ertek}; a rögzítés időpontja: {idopontok[max_index]}.")
```

BLZS©

BLZS©

1. Adatok eltárolása

A feladat kéri, hogy a 16 megadott számot tároljuk el a forráskódban egy adatszerkezetben.

meresek = [...] Ezt a szerkezetet listának hívjuk. Képzeld el úgy, mint egy nagy iratrendezőt, amiben rekeszek vannak, és mindegyik rekeszbe bedobunk egy számot, amit a meres.txt-ből kaptunk.

idopontok = [...] Ahelyett, hogy a 4. feladatban bonyolult matematikával számolnánk ki az időt a sorszámokból, készítünk egy másik listát az időpontoknak. A mérések 6:15-kor kezdődtek , és 10:00-ig tartottak. Ha tudjuk hányadik mérés volt a legnagyobb, csak megnézzük ugyanezt a sorszámot az időpontok listájában.

```
1   # 1. FELADAT: Adatok eltárolása
2   meresek = [ 36, 48, 39, -1, 30, 43, -1, 76, 67, 82, 73, 75, 64, 73, 69, 63 ]
3
4   # Segédlet a 4. feladathoz kezdőknek:
5   # Mivel csak 16 időpontunk van, a legkönnyebb, ha ezeket is egyszerűen felsoroljuk!
6   idopontok = [ "6:15", "6:30", "6:45", "7:00",  "7:15", "7:30", "7:45", "8:00",
7       "8:15", "8:30", "8:45", "9:00",  "9:15", "9:30", "9:45", "10:00" ]
8
```

BLZS©

2. Összes áthaladó kerékpáros

Itt össze kell adnunk a számokat, de figyelnünk kell arra,
hogy a hibát jelző -1 -eket kihagyjuk.

print("2. feladat") Kiírja a képernyőre a szöveget.

osszesen = 0 Létrehozunk egy képzeletbeli "dobozt" (változót),
amibe gyűjtjük a kerékpárosokat. Kezdetben ez nulla.

for adat in meresek: Ez egy ciklus.
Menj végig a meresek listán, fogd meg sorban az összes elemet,
és hívjuk most az éppen aktuálisat adat-nak.

```
 9  # 2. FELADAT: Összes kerékpáros
10  print("2. feladat")
11  osszesen = 0
12  for adat in meresek:
13      if adat != -1:
14          osszesen = osszesen + adat
15
16  print(f"Összesen {osszesen} kerékpárost számoltak.")
```

BLZS©

2. Összes áthaladó kerékpáros

if adat != -1: Ez egy feltétel. A != jelenti azt, hogy "nem egyenlő".
Tehát: "HA az adat nem egyenlő -1-gyel...„

osszesen = osszesen + adat "...akkor az eddigi összeghez add hozzá az aktuális adatot." (Ha az adat -1 volt, a gép ezt a sort átugorja).

print(f"Összesen {osszesen} ...") Az f a szöveg előtt azt jelenti, hogy "formázott" szöveg. Lehetővé teszi, hogy a szövegen belül kapcsos zárójelek {} közé betehessük egy doboz tartalmát.

```
 9   # 2. FELADAT: Összes kerékpáros
10   print("2. feladat")
11   osszesen = 0
12   for adat in meresek:
13       if adat != -1:
14           osszesen = osszesen + adat
15
16   print(f"Összesen {osszesen} kerékpárost számoltak.")
```

BLZS©

3. Óránkénti mérések

Mivel a mérések 15 percenként történtek, egy órában pontosan 4 mérés van.

aktualis_index = 0 Ez lesz a mi "mutatóujjunk", amivel követjük, hol tartunk az iratrendezőben (a meresek listában).

for ora in [6, 7, 8, 9]: Négy óránk van, ezeken megyünk végig.

ora_osszege = 0 Minden új óra kezdetén lenullázzuk a számlálót.

```
18  # 3. FELADAT: Óránkénti mérések
19  print("3. feladat")
20  print("Óránkénti mérések:")
21
22  aktualis_index = 0
23
24  for ora in [6, 7, 8, 9]:
25      ora_osszege = 0
26      for lepes in range(4):
27          adat = meresek[aktualis_index]
28          if adat != -1:
29              ora_osszege = ora_osszege + adat
30          aktualis_index = aktualis_index + 1
31
32      print(f"{ora} órától {ora_osszege} kerékpáros")
```

BLZS©

3. Óránkénti mérések

BLZS©

for lepes in range(4): Minden órán belül pontosan 4-szer kell lépnünk (4 negyedóra).

adat = meresek[aktualis_index] A szögletes zárójel [] segítségével vesszük ki a listából azt az elemet, amire a mutatóujjunk (aktualis_index) épp mutat.

aktualis_index = aktualis_index + 1 Miután feldolgoztunk egy elemet, a mutatóujjunkat eggyel odébb toljuk.

```
18  # 3. FELADAT: Óránkénti mérések
19  print("3. feladat")
20  print("Óránkénti mérések:")
21
22  aktualis_index = 0
23
24  for ora in [6, 7, 8, 9]:
25      ora_osszege = 0
26      for lepes in range(4):
27          adat = meresek[aktualis_index]
28          if adat != -1:
29              ora_osszege = ora_osszege + adat
30          aktualis_index = aktualis_index + 1
31
32      print(f"{ora} órától {ora_osszege} kerékpáros")
```

BLZS©

4. Maximum keresés

Meg kell keresnünk a legmagasabb értéket és annak idejét.

max_ertek = -1 és max_index = 0 A maximumkeresés trükkje az, hogy kinevezünk egy nevetségesen alacsony számot a "jelenlegi legnagyobb" (bajnok) értéknek.

for i in range(len(meresek)): Végigmegyünk a mérések sorszámain. A Python a sorszámozást 0-tól kezdi (0, 1, 2... 15).

```
33
34  # 4. FELADAT: Legnagyobb mérési érték
35  print("4. feladat")
36  max_ertek = -1
37  max_index = 0
38
39  for i in range(len(meresek)):
40      if meresek[i] > max_ertek:
41          max_ertek = meresek[i]
42          max_index = i
43
44  print(f"Az áthaladók maximális száma: {max_ertek}; a rögzítés időpontja: {idopontok[max_index]}.")
```

BLZS©

4. Maximum keresés

BLZS©

if meresek[i] > max_ertek: "HA a most vizsgált érték nagyobb, mint az eddigi bajnok...„

max_ertek = meresek[i] és max_index = i "...akkor legyen ő az új bajnok, és jegyezzük meg a sorszámát is (max_index)!".

print(f"... {idopontok[max_index]}.") A végén az időpontok listájából egyszerűen elővesszük a nyertes sorszámához tartozó időt .

```
33
34  # 4. FELADAT: Legnagyobb mérési érték
35  print("4. feladat")
36  max_ertek = -1
37  max_index = 0
38
39  for i in range(len(meresek)):
40      if meresek[i] > max_ertek:
41          max_ertek = meresek[i]
42          max_index = i
43
44  print(f"Az áthaladók maximális száma: {max_ertek}; a rögzítés időpontja: {idopontok[max_index]}.")
```

BLZS©

Tipikus hibák!

BLZS©

A "behúzások" (Indentáció) hiánya vagy keverése: A Pythonban nagyon fontosak a sorok elején lévő szóközök (általában 4 szóköz vagy egy Tab). Ezek jelzik, hogy melyik utasítás tartozik bele a for ciklusba vagy az if feltételbe. Ha ezek nincsenek jó helyen, a program hibát fog dobni, vagy rosszul számol.

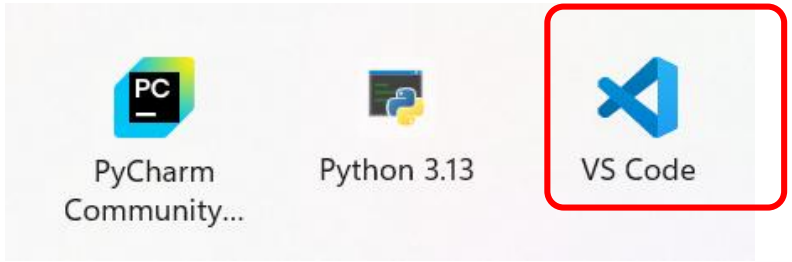
A nulla-alapú sorszámozás elfelejtése: Ne feledd, a számítógép nullától kezd számolni! A meresek lista legelső eleme a meresek[0], a második a meresek[1]. Ha ezt elrontod, minden adat elcsúszik egy hellyel.

Változónevek elgépelése: A gép nem érti a magyar nyelvet, csak az általad kitalált neveket. Ha valahol osszesen névvel hoztál létre egy változót, de később oszesen-ként hivatkozol rá, a gép összeomlik, mert szerinte olyan "doboz" nem létezik.

Kis- és nagybetűk felcserélése: A Python (és a legtöbb programnyelv) "case-sensitive". Ez azt jelenti, hogy a max_ertek és a Max_ertek számára két teljesen különböző dolog. Maradj a kisbetűk használatánál, úgy a legbiztonságosabb!

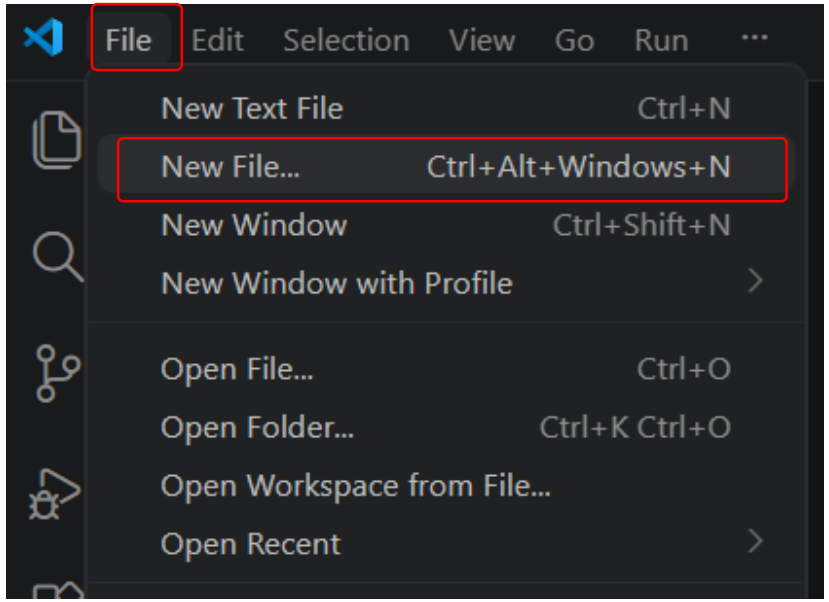
BLZS©

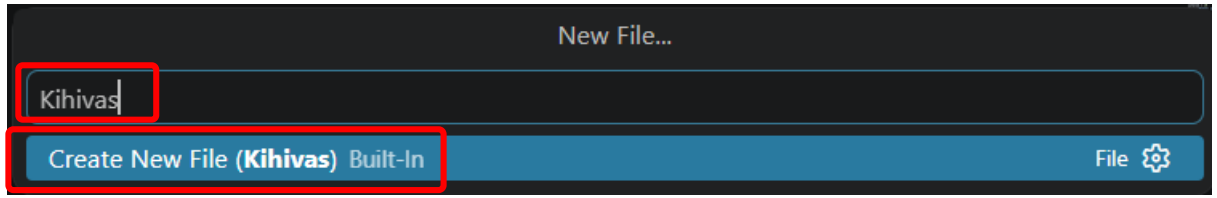
VS Code

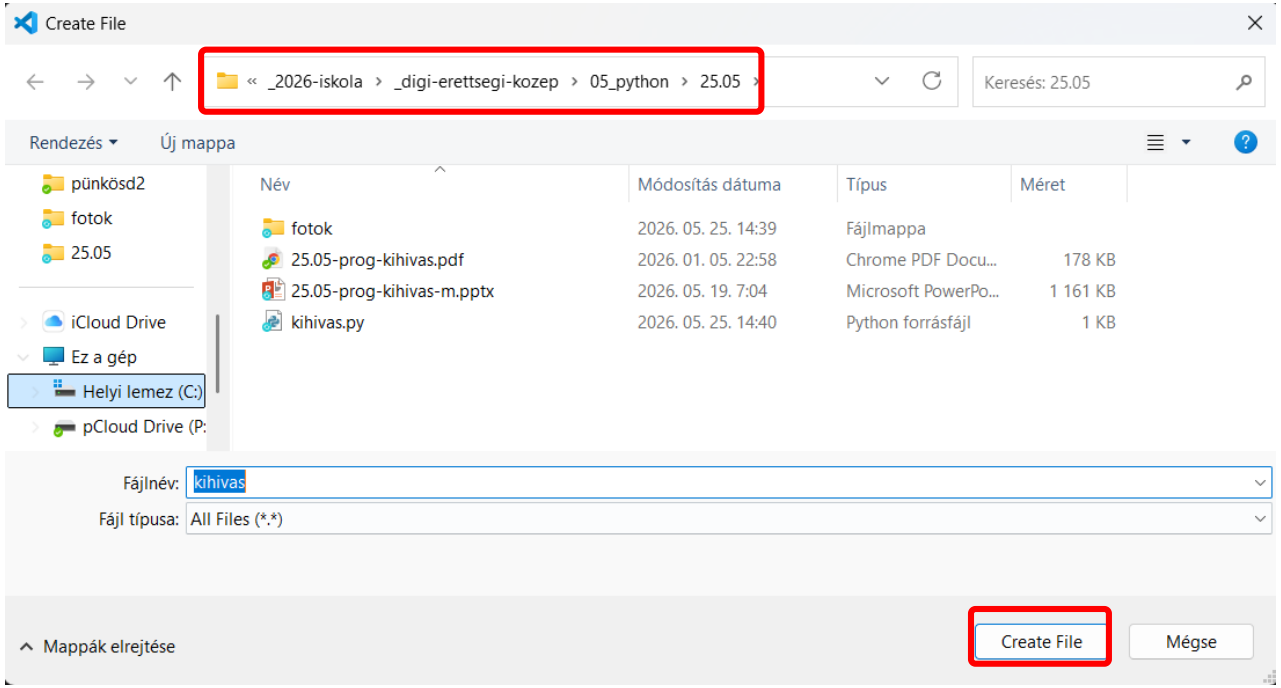


BLZS©

File -> New Project -> kihivas-> Create New File-> Mentés helye -> Create File

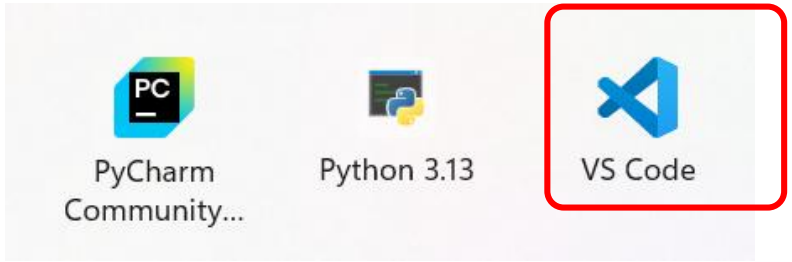






BLZS©

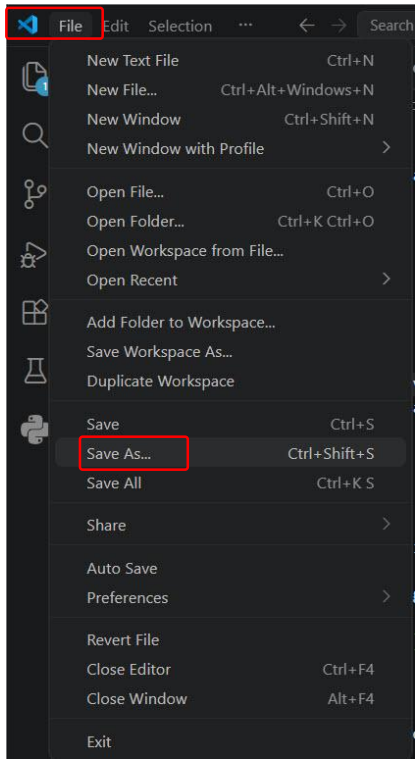
VS Code

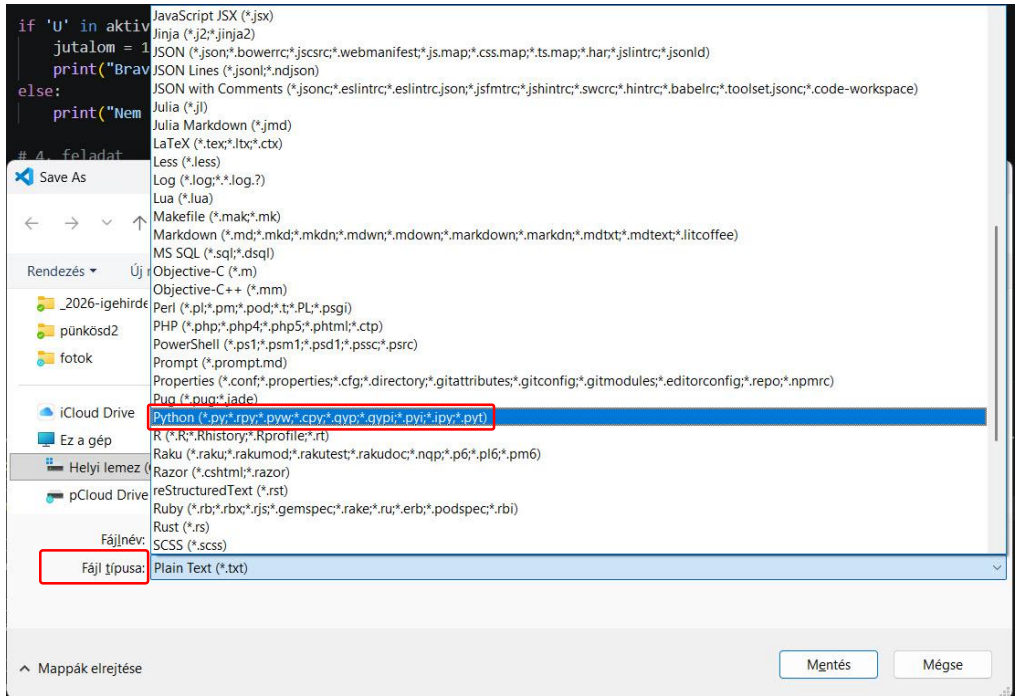


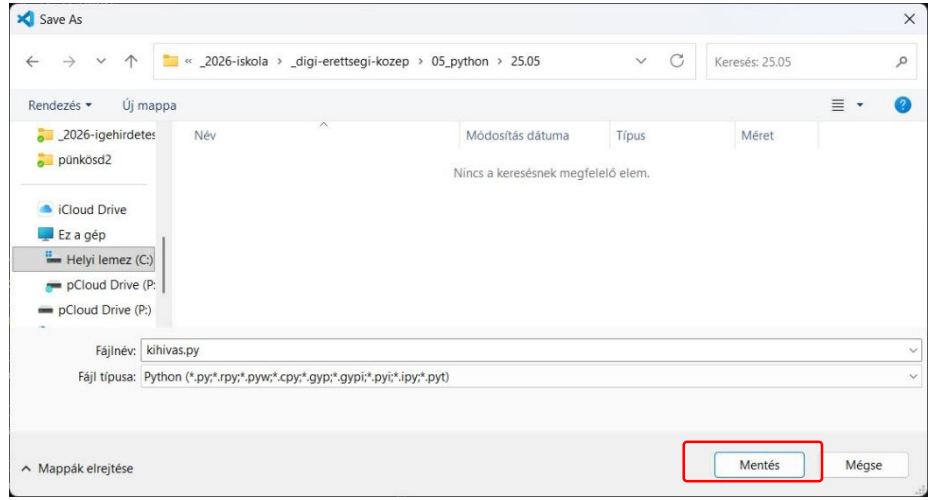
BLZS©

File -> New Project -> befozes-> Create New File-> Mentés helye -> Create File

File -> Save As... -> Fájl típusa: -> Python -> Mentés



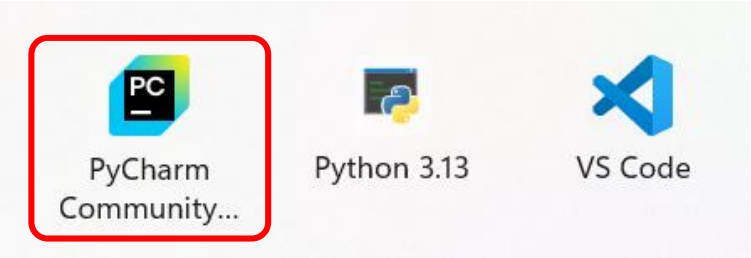




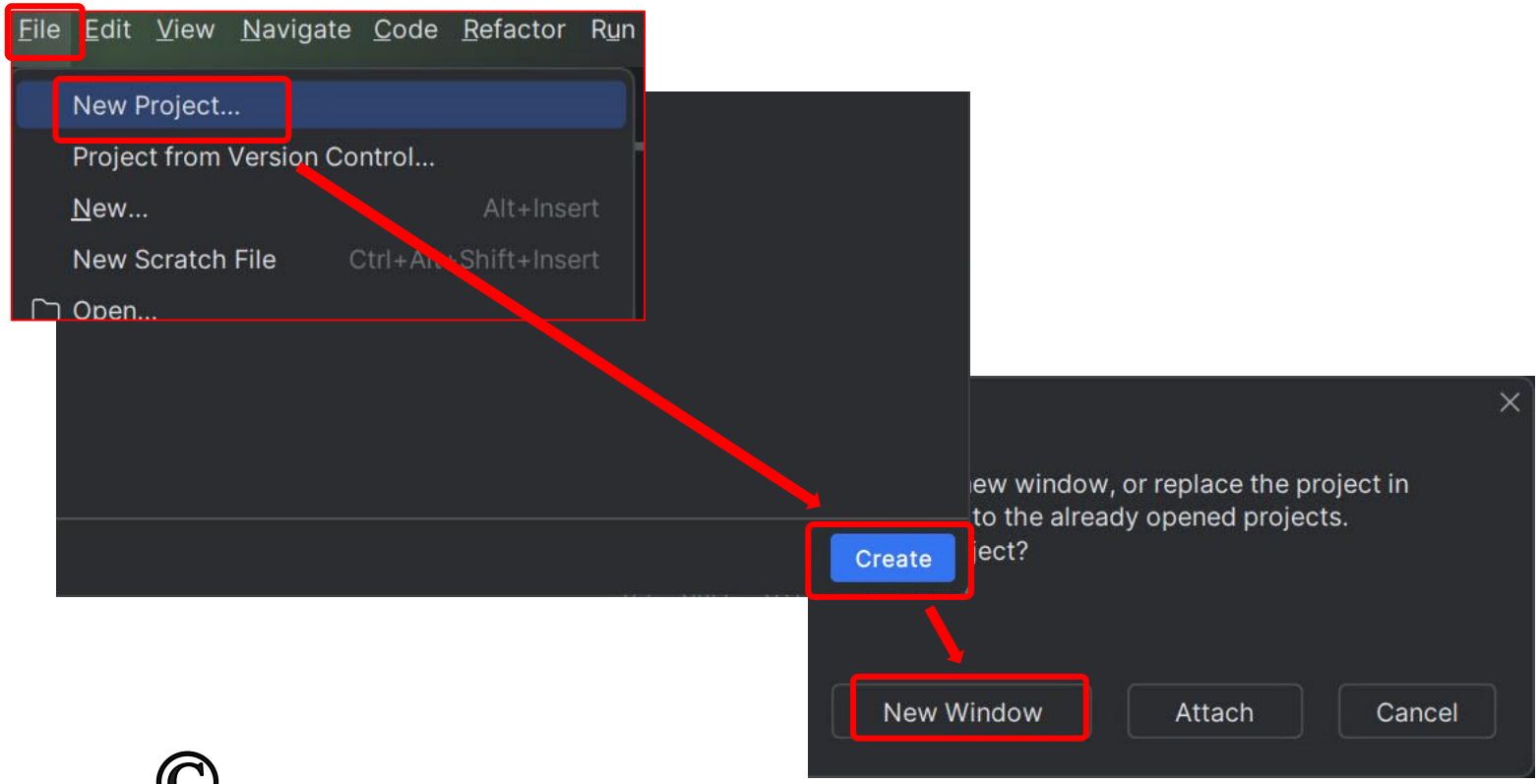
BLZS©

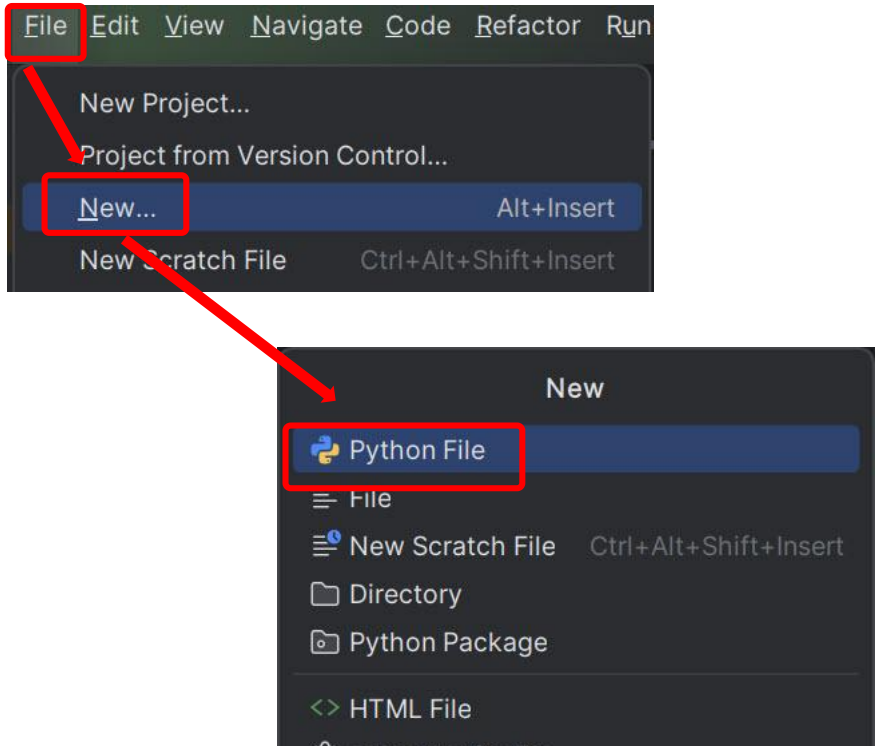
BLZS©

PyCharm



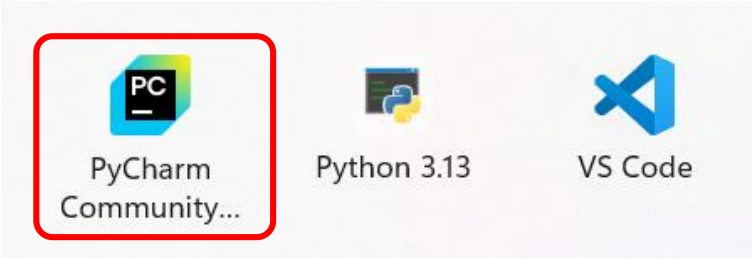
Suliban:
File -> New Project -> Create -> New Windows -> File -> New -> Python File

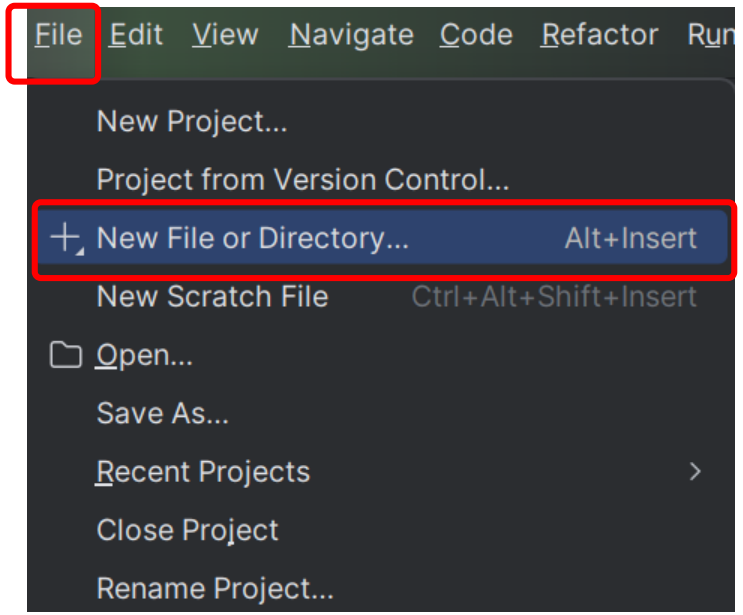


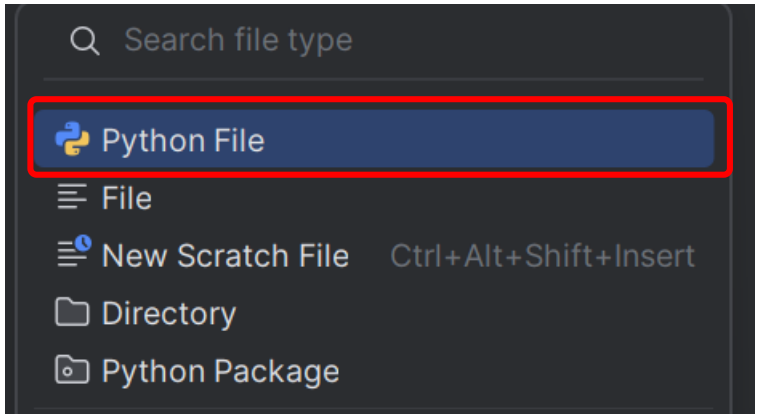


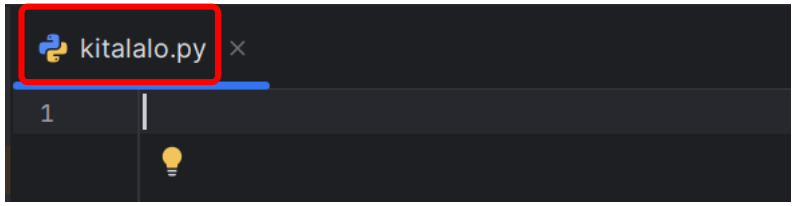
BLZS©

PyCharm









BLZS©