Programozás 2 cheatsheet, 2. ZH

1. Osztály definiálása (`object` gyermekeként)

```
class <Osztálynév>:
    pass
```

2. Osztály definiálása (származtatással)

```
class <Osztálynév>(<Szülő1>[, <Szülő2>, ...]):
    pass
```

3. Adatosztály létrehozása

```
@dataclass
class MyClass:
    my_field: int = field()
```

4. A `field()` függvény paraméterezése:

   - `compare` – használjuk-e az attribútumot az `__eq__()` függvényben, logikai, alapértelmezettként `True`
   - `hash` – használjuk-e az attribútumot a `__hash__()` függvényben, logikai, alapértelmezettként `False`
   - `repr` – használjuk-e az attribútumot a `__repr__()` függvényben, logikai, alapértelmezettként `True`
   - `default` – az attribútumhoz tartozó alapértelmezett érték az `__init__()` metódus paraméterlistáján, egyszerű érték, opcionális
   - `default_factory` – az attribútumhoz tartozó alapértelmezett érték az `__init__()` metódus paraméterlistáján, paraméter nélküli függvény (mely visszaadja az alapértelmezett értéket), opcionális

5. Az Enum-osztály definiálása

```
class MyEnum(Enum):
    VALUE_1 = "value 1"
    VALUE_2 = "value 2"
```

6. List comprehension

```
[  visszaadott_érték
   for elem in szekvencia
   if szűrés_feltétele ]
```

7. Set comprehension

```
{  visszaadott_érték
   for elem in szekvencia
   if szűrés_feltétele }
```

8. Típus annotációk engedélyezése

```
from __future__ import annotations
```

11. Dictionary comprehension

```
{  kulcs: érték
   for elem in szekvencia
   if szűrés_feltétele }
```

12. Hasznos függvények

   - `min()` – az objektum legkisebb értéke
   - `max()` – az objektum legnagyobb értéke
   - `sum()` – az objektum/szekvencia összege
   - `next()` – az objektum/szekvencia első eleme
   - `chain.from_iterable()` – egymásba ágyazott szekvenciák "kibontása"

13. Kivétel eldobása

   - `assert` utasítással:

     ```
     assert person.age >= 0, "message"
     ```

   - `raise` utasítással:

     ```
     if person.age < 0:
         raise ValueError("message")
     ```

14. Kivételkezelés

```
try:
    pass # that can cause an exception
except ValueError:
    pass # handling the exception
```

15. Belépési pont, főprogram

```
def main() -> None:
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

16. Példa a `type_mapper()` metódusra:

```
@staticmethod
def type_mapper(values: dict[str, any]) \
    -> Route | Route.Flight:
    if "country" in values:
        route = Route(**values)
        route.operator = next(
            Route.Airline[entry.name]
            for entry in Route.Airline
            if entry.value == route.operator
        )
        return route
    else:
        return Route.Flight(**values)
```

17. Példa az `entities()` tulajdonságra:

```
@property
def entities(self) -> list[Route]:
    return cast(list[Route], super().entities)
```