Programozás 2 cheatsheet

A *Programozás 2* tantárgy első zárthelyi dolgozatán használható, hivatalos segédeszköz.

1. Osztály definiálása (`object` gyermekeként)

```
class <Osztálynév>:
    pass
```

2. Osztály definiálása (származtatással)

```
class <Osztálynév>(<Szülő>):
    pass
```

3. Láthatóságok

```
apple: int      # publikus
_banana: int    # védett
__lemon: int    # privát
```

4. Tulajdonság definiálása

```
@property
def foo(self) -> int:
    return self.__foo

@foo.setter
def foo(self, value: int) -> None:
    self.__foo = value
```

5. Tulajdonság kezdőértékkel (osztályszintű)

```
class Foo:
    an_integer: int = 5
    an_integer_list: list[int] = []
```

6. Speciális azonosítók
   - `self`: aktuális példány hivatkozása
   - `super()`: szülőosztály hivatkozása

7. Példányszintű metódus definiálása

```
def hello(self) -> str:
    return "Hello, World!"
```

8. Osztályszintű (statikus) metódus definiálása

```
@staticmethod
def hello() -> str:
    return "Hello, World!"
```

9. Konstruktor definiálása

```
def __init__(self, foo) -> None:
    self.__foo = foo
```

10. Reprezentáció megadása

```
def __repr__(self) -> str:
    return f"representation for debugging"
```

11. Sztring reprezentáció megadása

```
def __str__(self) -> str:
    return f"pretty representation"
```

12. Operátorok túlterhelése:

```
def __eq__(self, o: object) -> bool:
    pass
```

   - Visszatérési érték:
     - `True`, ha az operátor igaz értéket állít elő
     - `False`, ha az operátor hamis értéket állít elő
     - `NotImplemented`, ha az operátor nem támogatott
   - Operátor és metódus párosítások:
     - `<`: `__lt__()`
     - `<=`: `__le__()`
     - `>`: `__gt__()`
     - `>=`: `__ge__()`
     - `!=`: `__ne__()`
   - Teljes rendezettség:

```
from functools import total_ordering

@total_ordering
class Foo:
    pass
```

13. Hash-függvény definiálása

```
def __hash__(self) -> int:
    return self.__repr__().__hash__()
```

14. Parancssori argumentumok használata

```
import sys

sys.argv[1] # első parancssori argumentum
```

15. Kivétel eldobása
   - `assert` utasítással:

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

   - `raise` utasítással:

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

16. Kivételkezelés

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

17. Belépési pont, főprogram

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

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

18. Típus annotációk engedélyezése

```
from __future__ import annotations
```