```
from enum import *
from typing import Any
from typing import Self

class Genere(StrEnum):
    uomo = auto()
    donna = auto()

print("__name__ all'interno di mytypes.py: " + __name__)

class Voto(int):
    def __new__ (cls, v:int) -> Self:
        if v < 18 or v > 31:
            raise ValueError(f"Il voto v={v} deve essere tra 18 e 31")
        return int.__new__(cls, v)



'''class Voto:
    v: int
    def __init__(self, v: int):
        if v < 18 or v > 31:
            raise ValueError(f"Il voto v={v} deve essere tra 18 e 31")
        self.v = v

    def __eq__(self, other: Any) -> bool:
        return self.v == other.v'''



if __name__ == '__main__':
    print("Test di mytypes.py\n========\n")



    print(Genere.uomo)
    print(type(Genere.uomo))
    print(Genere.donna)


class Indirizzo:
    def __init__(self, via: str, civico: str, cap: str) -> str:
        print(f"Tipo di 'via': {type(via)}")
        if not isinstance(via, str) or not isinstance(civico, str) or not
(isinstance(cap, str) and len(cap) == 5 and cap.isdigit()):
            raise TypeError("Dati non validi per un indirizzo")

        self.via = via
        self.civico = civico
        self.cap = cap
        print(type(self.via))

    def get_via(self) -> str:
        return self.via
```

```
    def get_civico(self) -> str:
        return self.civico

    def get_cap(self) -> str:
        return self.cap

    def __hash__(self) -> int:
        return hash((self.get_via(), self.get_civico(), self.get_cap()))

    def __eq__(self, other:Any) -> bool:
        if other is None or not isinstance(other, type(self)) or
hash(self) != hash(other):
            return False
        return (self.get_via(), self.get_civico(), self.get_cap()) ==
(other.get_via(), other.get_civico(), other.get_cap())
    print("Via Roma", "12", "00100")
indirizzo = Indirizzo(123, True, 5000)
print(indirizzo)

print("------------")


class Email:
    c:str = "@"
    domini: list = [".it", ".com"]
    def __init__(self, c: str, i: str, f:str, domini: list, dominio: str)
-> None:
        if not isinstance(i, str) or not c =="@" or not isinstance(f, str)
or dominio not in domini:
            raise TypeError("Dati inseriti non validi")
        self.c = c
        self.i = i
        self.f = f
        self.dominio = dominio

    def get_c(self) -> str:
        return self.c
    def get_i(self) -> str:
        return self.i
    def get_f(self) -> str:
        return self.f
    def get_dominio(self) -> str:
        return self.dominio

    def __hash__(self) -> int:
        return hash((self.get_c(), self.get_i(), self.get_f(),
self.get_dominio()))
    def __eq__(self, other:Any) -> bool:
        if other is None or not isinstance(other, type(self)) or
hash(self) != hash(other):
            return False
```

```
        return (self.get_c(), self.get_i(), self.get_f(),
self.get_dominio()) == (other.get_c(), other.get_i(), other.get_f(),
other.get_dominio())

print("------------")

class Telefono:
    def __init__(self, numero: str) -> None:
        if len(numero) != 10 and numero.isdigit():
            raise TypeError("Telefono non valido")
        self.numero = numero

    def get_numero(self) -> str:
        return self.numero

    def __hash__(self) -> int:
        return hash((self.get_numero()))
    def __eq__(self, other:Any) -> bool:
        if other is None or not isinstance(other, type(self)) or
hash(self) != hash(other):
            return False
        return (self.get_numero())

print("------------")

class CF(str):
    def __new__(cls, valore: str) -> Self:
        if not valore.isalnum () or len(valore) != 16:
            raise ValueError("Codice Fiscale non valido")
        return str.__new__(cls, valore)

print("------------")
```