1. Jaki kod nazywamy czystym kodem?

2. Czym sa reguły PEP?

3. Co to znaczy, że Python jest językiem dynamicznie typowanym? Podaj przykład ręcznie typowanych zmiennych podawanych jako argumenty funkcji.

4. Dlaczego warto ręcznie typować zmienne?

5. Co to jest SOLID?

1. Łatwo rozszerzalny, łatwy do czytania – zrozumiały dla innych programistów i dostosowany do konkwencji kodowania.
2. Python Enhancement Proposals – zbiór zasad definiujący ilość linii w wierszu, przerwy pomiędzy klasami, sosób nazywania metod, itp.
3. Nie musimy ręcznie określać typów tworzonych zmiennych.

def funkcja(zmienna: str, inna\_zmienna: int, zmienna\_list: List[int]) -> int:

pass

def funkcja(zmienna: str, inna\_zmienna: int, zmienna\_list: List[int]) -> None :

pass

1. Poprawia się czytelność kodu, dobre IDE ostrzega jeśli próbujemy podać wartość innego typu do zmiennej
2. Zbiór reguł umożliwiających tworzenie aplikacji na najwyższym poziomie wymagań klienta.

S - Single Responsibility principle (Zasada jednej odpowiedzialności)

O - Open/Closed principle (Zasada otwarte/zamknięte)

L - Liskov Substitution principle (Zasada podstawienia Liskov)

I - Interface Segregation principle (Zasada segregacji interfejsów)

D - Dependency Inversion Principle (Zasada odwrócenia zależności)

S – nigdy nie powinno być więcej niż jednego powodu do istnienia klasy bądź metody – funkcja powinna realizować tylko jedno zadanie

O – klasa lub metoda powinna być otwarta na rozszerzenia ale zamknięta na modyfikacje – kod powinien pozwalać na dodanie nowych funkcjonalności bez naruszania istniejących

L – jeśli klasa s dzidziczy z klasy t to obiekty klasy t mogą być zastąpione obiektami klasy s bez zabirzania poprawności działania programu – klasa dziedzicząna nie pwoinna wykonywać mniej operacji niż klasa jej rodzica

I – należy możliwie często wprowadzać do programu interfejsy (klasy abstrakcyjne)

D – Moduły wysokopoziomowe nie powinny zależeć od modułów niskopoziomowych