*PYTHON DOLGOZAT*

***OOP a Python-ban***

***2022***

|  |  |
| --- | --- |
| ***Osztályzat*** | ***Pontok*** |
| ***1*** | ***00 - 13*** |
| ***2*** | ***14 - 19*** |
| ***3*** | ***20 - 26*** |
| ***4*** | ***27 - 33*** |
| ***5*** | ***34 - 37*** |

**A kidolgozott dolgozat dokumentumát, melynek a neve a tanuló saját neve (Vezetéknév Keresztnév) vissza kell másolni a hálózatra, az alábbi címre:**

**\\optimus\h\Vastag Atila\leadando**

**1** – **Az OOP mit helyez a középpontba?** **(1)**

**válasz: összetett adattagokat**

**2 – *Az OOP világában mit jelent az osztály*? (1)**

**válasz: Műveletek és adatok összessége, amivel egy dolog tulajdonságait vagy működését írjuk le**

**3 – Egy objektumnak az életciklusa során megváltozhat az állapota, tulajdonságai. Ezt az állapotot valahogy el kell tudnunk tárolni, illetve biztosítani kell a szükséges műveleteket a tulajdonságok megváltoztatásához. A tulajdonságokat tároló változókat: (1)**

**Húzza alá a helyes választ!**

1. adatoknak nevezzük.
2. **adattagonka nevezzük.**
3. függvényeknek nevezzük.
4. tulajdonságoknak nevezzük.

**4 – Egy objektumnak az életciklusa során megváltozhat az állapota, tulajdonságai. Ezt az állapotot valahogy el kell tudnunk tárolni, illetve biztosítani kell a szükséges műveleteket a tulajdonságok megváltoztatásához. A műveleteket: (1)**

**Húzza alá a helyes választ!**

1. akcióknak nevezzük.
2. adattagonka nevezzük.
3. **függvényeknek nevezzük**.
4. tulajdonságoknak nevezzük.
5. **- Hogyan különböztessük meg, hogy mikor adattag és mikor függvény (metódus)? (1)**

**válasz: Ha műveletet végzünk el vele akkor függvény, ha adatot adunk meg akkor adattag**

**6 – Hogyan definiálunk egy osztályt, mely kulcsszó segítségével? (1)**

**válasz: class**

1. **– A mellékelt *Négyzet* osztályt figyelembe véve, válaszoljon az alábbi kérdésekre: (3)**

class Negyzet:

    def \_\_init\_\_(self, a: float = 0):

        super().\_\_init\_\_()

        self.a: float = a

    def terulet(self) -> float:

        return self.a \* self.a

    def kerulet(self) -> float:

        return 4 \* self.a

* **adattag(ok): self.a**
* **függvény(ek): self.a\*self.a, 4\*self.a**

**7 – negyzet: Negyzet = Negyzet(10)**

**Ebben a kifejezésben mit jelent a Negyzet és mi a negyzet? (2)**

* **negyzet: változó neve**
* **Negyzet: osztály neve**

**9 – Mit kell tudni a példán látható *Test* osztály: (2)**

class Test:

    globalisValtozo: float = 0

def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.lokalisValtozo: float = 0

* **globalisValtozo-rol: statikus érték**
* **lokalisValtozo-rol: változó érték**

**10 – Hogyan lehet az osztály konkrét példányán elérni az adattagokat és a függvényeket*?* (1)**

**válasz: változóneve.adattagneve**

**11 – Egy osztály hány entitásért kell, hogy felelős legyen? (1)**

**Húzza alá a helyes választ!**

1. **egy**
2. **kettő**
3. **több**
4. **egyért sem**
5. **nincs is ilyen**

**12 – Egészítse ki a mondat hiányzó részét!**

**Minden esetben, amikor példányosítunk egy osztályt egy speciális metódus a konstruktor fut le, melynek feladata, hogy „beállítsa” az osztály adattag értékeit. (1)**

**13 – Mi a konstruktor feladata? (1)**

**válasz: inicializálja az értékeket,**

**14 – Hogyan jelöljük a konstruktort *python* programozási nyelvben? (1)**

**válasz: \_\_init\_\_**

**15 – Mely kulcsszóval kell ellátni egy osztály függvényét, hogy az statikus legyen? (1)**

**válasz: @staticmethod**

class Jarmu:

    def \_\_init\_\_(self, gyarto: str, model:str):

        self.name: str = name

        self.\_model: str = model

class Gepkocsi(Jarmu):

    def \_\_init\_\_(self, name: str, model:str, kw: int):

        Employee.\_\_init\_\_(self, name, model)

        self**.\_\_kw**: int = kw

**16 – Mit láthatunk a jobb oldali kódban? (1)**

**válasz: öröklődést**

**17 – Mit fog kiírni és miért a következő program, ha a jobb oldali kódrészletet vezzük figyelembe? (3+3)**

peugeot: Gepkocsi = Gepkocsi("Peugeot","208 1.6 e-HDI", 72)

* print(peugeot.gyarto): Peugeot , mert: Mert az érték minden szinten öröklődik
* print(peugeot.\_model): 208 1.6 e-HDI , mert: Még benne van az 1 öröklődési szintbe
* print(peugeot.\_\_kw):72 , mert: osztály saját értéke, nem örököli tehát hiába private

**18 - A python *három* módosítót ismer, melyek alkalmazhatóak adattagokra és függvényekre is. Mit kell tudni ezekről a módosítokról? (7)**

* **public: Minden szinten látható**
* **private:**
* **nem öröklődik**
* **csak a saját osztályán belül olvasható**
* **jele:\_\_**
* **protected:**
* **1 szintig öröklődik**
* **2. öröklődésnél már nem olvasható**
* **jele: \_**

**19 – Minek a bemutatását láthatjuk a jobb oldali kódon? (2)**

# person.py

class Person:

    def \_\_init\_\_(self, name: str, age: int = 0):

        self.name: str = name

        self.\_\_age: int = age

    def display(self):

        print(self.name)

        print(self.\_\_age)

    def getAge(self) -> int:

        return self.\_\_age

    def  setAge(self, age: int) -> None:

        self.\_\_age = age

**válasz: get és set metódus alkalmazását**

**20 – Miért van szükség a 19 feladat válaszában adott függvényekre? (2)**

**válasz: megváltoztassák az adattagok értékeit**

**JOKER:**

**Konstruktor írásának szabályai: (3)**

**nincs visszatérési értéke**

**több is lehet belőle de elnevezésben egyértelmű különbségnek kell lennie**