# Rozdział 5 - Pytania

**Pytanie 1**
Wybierz poprawne stwierdzenia dotyczące klas i obiektów w języku Python.

A. Klasy, to schematy, na bazie których tworzymy obiekty.
B. Obiekty, to instancje klas.
C. W języku Python klasy również są obiektami.
D. Można utworzyć tylko jedną instancję danej klasy.

**Pytanie 2**
Dany jest kod źródłowy:

```
class Klasa:
      pass

k = Klasa()
print(type(k))
```

Co zostanie wyświetlone w konsoli po uruchomieniu programu?

A. `<class '__main__.Klasa'>`
B. `<__main__.Klasa object at …>`
C. `<class 'Klasa'>`
D. `<instance of class 'Klasa'>`

**Pytanie 3**
Dany jest kod źródłowy:

```
class Klasa:
      def powitanie(self):
      return "Witaj!", self

k = Klasa()
print(k.powitanie()[0])
```

Co zostanie wyświetlone w konsoli po uruchomieniu programu?

A. Nic
B. `Witaj!`
C. `["Witaj!"]`
D. Interpreter zgłosi błąd `NoneTypeError: return of function powitanie() was not assigned to any variable (2 required)`

**Pytanie 4**
Jakiego dekoratora należy użyć celem zadeklarowania wybranej metody, jako statycznej?

A. `@static`
B. `@staticmethod`
C. `@staticfunction`
D. `@staticobject`

**Pytanie 5**
Dany jest kod źródłowy:

```
class Math:
   @classmethod
   def set_values(cls, x, y):
       cls.x = x
       cls.y = y

M1 = Math()
M1.set_values(1, 1)
M2 = Math()
print(M2.x, M2.y)
```

Co zostanie wyświetlone w konsoli po uruchomieniu programu?

A. `None None`
B. `0 0`
C. `1 1`
D. Interpreter zgłosi błąd `AttributeError: 'Math' object has no attribute 'x'`

**Pytanie 6**
Czy język Python obsługuje dziedziczenie i polimorfizm?

A. Dziedziczenie tak, ale polimorfizm nie.
B. Polimorfizm tak, ale dziedziczenie nie.
C. Język Python obsługuje oba te mechanizmy.
D. Język Python nie obsługuje tych mechanizmów.

**Pytanie 7**
Dany jest kod źródłowy:

```
class Program:
    memory = '0011'

    def __init__(self, memory):
        self.memory = memory

program = Program('1100')
print(Program.memory)
```

Co zostanie wyświetlone w konsoli po uruchomieniu programu?

A. Interpreter zgłosi błąd `ClassInstanceError: 'Program' class has existence instance 'program'. Use it instead.`
B. `None`
C. `1100`
D. `0011`

**Pytanie 8**
Dany jest kod źródłowy:

```
class Computer:
    def __init__(self, cpu):
        self.cpu = cpu

    def get(self):
        return self.cpu

class MacBook(Computer):
    def __init__(self, cpu, os):
        super().__init__(cpu)
        self.os = os

computer = MacBook('M2', 'macOS')
```

Które z poniższych linii kodu po dodaniu pod kodem przytoczonym powyżej spowodują, że po uruchomieniu programu ujrzymy w konsoli `M2`?

A. `print(Computer.cpu)`
B. `print(MacBook.cpu)`
C. `print(computer.cpu)`
D. `print(computer.get())`

**Pytanie 9**
Dany jest kod źródłowy:

```
class Firma:
    def __init__(self):
        self._projekt = 'TR3B'

class Pracownik(Firma):
    def __init__(self, pracownik):
        self.pracownik = pracownik
        Firma.__init__(self)

    def wyswietl_informacje(self):
        print("Pracownik:", self.pracownik, end=', ')
        print("pracuje nad:", self._projekt)


p = Pracownik("Bob")
p.wyswietl_informacje()
```

Co zostanie wyświetlone w konsoli po uruchomieniu programu?

A. `Pracownik: Bob, pracuje nad: TR3B`
B. `Pracownik: Bob, pracuje nad: None`
C. `Pracownik: Bob, pracuje nad:`
D. Interpreter zgłosi błąd `ProtectedAccessError: Object assigned to '_projekt' variable is protected`

**Pytanie 10**
Dany jest kod źródłowy:

```
class Damage:

    def __init__(self, dmg: int):
        self.__dmg = dmg

    def __add__(self, other):
        if isinstance(other, Damage):
            return Damage(self.__dmg + other.__dmg)

    def overall_dmg(self):
        return self.__dmg

dmg_1, dmg_2 = Damage(34), Damage(47)
print((dmg_1 + dmg_2).overall_dmg())
```

Co zostanie wyświetlone w konsoli po uruchomieniu programu?

A. `34`
B. `47`
C. `81`
D. Interpreter zgłosi błąd `AttributeError: 'Damage' object has no attribute 'dmg'`