Question 1. True or False

Circle **T** if the statement is true, otherwise circle **F** if the statement is false.

1. A virtual function overloading an operator is an example of dynamic dispatch. **(T)** **F**
2. Dynamically typed interpreted language cannot implement early binding. **(T)** **F**
3. `decltype` in C++ is an example of static reflection. **T** **(F)**
4. The instance variables specified in the `__slots__` special variable must be initialized before they can be referenced. **(T)** **F**
5. You can call any method defined in the super class by replacing `self` with `super()` in the methods of a subclass. **(T)** **F**
6. You can overload the assignment operator in Python. **T** **(F)**
7. The `__str__` special method is an example of ad-hoc polymorphism. **(T)** **F**
8. The `__dict__` special attribute is an example of attribute reification. **(T)** **F**
9. A class method cannot be used to change instance variables. **(T)** **F**
10. If an attribute name starts with an underscore, it can only be accessed by instance methods of the same class. **T** **(F)**

Question 2. Multiple Choices

Pick all answer(s) that are correct.

a) Which of the following functions in Python are examples of introspection?

i. `globals`

ii. `getattr`

iii. `setattr`

iv. `delattr`

v. `hasattr`

b) Given the following code snippet:

```
class A:
    foo = [1, 2]

x = A()
y = A()
```

Which one of the following statements are true? (Note: the previous statements do not affect the next)

i. After running `x.foo.append(3)`, the value of `y.foo` is `[1, 2, 3]`.

ii. After running `x.foo = [1, 2, 3]`, the value of `y.foo` is `[1, 2, 3]`.

iii. After running `A.foo = [1, 2, 3]`, the value of `y.foo` is `[1, 2, 3]`.

iv. After running `setattr(x, 'foo', [1, 2, 3])`, the value of `y.foo` is `[1, 2, 3]`.

v. After running `getattr(x, 'foo').append(3)`, the value of `y.foo` is `[1, 2, 3]`.

Question 3. Short Questions

a) Describe type erasure, and its pros and cons.

Type erasure is the removal of type information by the compiler by doing all the necessary type checking at compile time. The benefit of type erasure is that it improves performance of the program. The issue with type erasure is that since type is not checked at runtime, a hacker could potentially inject malicious input that can compromise the program.

b) Override the eat method in Animal so that the eat method in Dog will, *in addition to what* Animal.eat *already does*, print “Wags its tail” at the very end. Show the entire class definition for Dog.

```
class Animal:
   …
   # may change this function in the future
   def eat(self, food):
        print(str(food) + " is delicious")
```

```
class Dog(Animal):
    def eat(self, food):
        super().eat(food)        # or Animal.eat(self, food)
        print("Wags its tail")
```

c) Show that functions in Python are first-class citizens.

1. Function can be used.

```
def foo():
    pass
f = foo     # can be assigned to variable
print(foo)  # can be passed to a function
```

2. Function can be constructed:

```
def foo(x):
    def bar():      # function constructed locally
        print(x)
    return bar
```

3. Function has a type. It is the class 'function'.

d) In Python, what is the difference between the class 'function' and the class 'method'?

A function is not bounded to an instance whereas a method is. The other way of saying it is that a method already has it's first argument bounded to an instance but a function does not. E.g.

```
class A:
    def foo(self, x):
        …

a = A()
```

In this case, `A.foo` is an instance of a function but `a.foo` is an instance of a method. Calling `A.foo` requires 2 arguments (e.g. `A.foo(a, 5)`) whereas calling `a.foo` requires only 1 more argument (e.g. `a.foo(7)`).

Question 4. Programming Question

Create a Sheet class that works like a spreadsheet with a fixed dimension. The constructor should take 3 arguments, *width*, *height*, and *type*, which will create a *width* by *height* matrix filled with the default value of that type. For example:

```
>> sh = Sheet(3, 2, int)
>> print(sh)
0, 0, 0
0, 0, 0
```

The Sheet class must overload the index operator so that it takes a tuple of 2 elements in the form (*x*, *y*). If the value being set is not the type specified in the constructor, a `TypeError` must be raised. E.g.:

```
>> sh[2,1] = 5
>> sh[0,0] = 7
>> print(sh)
7, 0, 0
0, 0, 5
```

```
>> print(sh[1,1])
0
>> sh[0,1] = 3.2
TypeError: type must be int
```

Lastly, printing a Sheet object should output the matrix in comma separated format (see first example).

```
class Sheet:
    def __init__(self, w, h, t):
        self.type = t
        self.matrix = []
        row = [ t() ] * w
        for _ in range(h):              # _ means “don’t care”
            self.matrix.append(row[:]) # row[:] makes a new copy

    def __getitem__(self, idx):
        return self.matrix[idx[1]][idx[0]]

    def __setitem__(self, idx, val):
        if not isinstance(val, self.type):
            raise TypeError("type must be %s"%self.type.__name__)
        self.matrix[idx[1]][idx[0]] = val

    def __str__(self):
        out = []
        for row in self.matrix:
            temp = []
            for val in row:
                temp.append(str(val))
            out.append(", ".join(temp))
        return "\n".join(out)
```