ECE326 – TUTORIAL 5TH WEEK

PREPARED BY MARTIYA ZARE JAHROMI

UNIVERSITY OF TORONTO

BOUNDLESS

AGENDA

- Exercise 2
- Exercise 3

EXERCISE 2 – 1. TRUE OR FALSE

1. Assignment in Python is always by reference.  **F**
    - Mutable objects
        - You can change the value
            - When you change the value of the assigned object you are changing the same object
    - Immutable objects
        - You cannot change the value
            - When you change the value after assignment, a new object will be created

EXERCISE 2 – 1. TRUE OR FALSE

2. Dynamically typed languages do not perform type checking. **T** **F**

- E.g.

  When you concatenate string and integer

EXERCISE 2 – 1. TRUE OR FALSE

3. global keyword is required to read a global variable from inside a function. **T** **F**

- global keyword is needed to reassign global variables.

EXERCISE 2 – 2. MULTIPLE ANSWERS

1. Which of the following operations are allowed inside a pure function?

- ✓ (a) Read from a constant global variable
- ✗ (b) Read from a static function variable (are mutable unless specified const.)
- ✓ (c) Modify a local variable
- ✗ (d) Call print function to write to console (I/O is part of side effects)
- ✓ (e) Call another pure function

EXERCISE 2 – 2. MULTIPLE ANSWERS

None

2. In Python 3, print returns the string it printed to screen.

Which of the following is true?

✗ (a) print is an statement

✓ (b) print is an expression

✓ (c) You can assign print to a variable, e.g. a = print

✓ (d) You can pass print to a function, e.g. foo(print)

✓ (e) You can assign a value to print, e.g. print = 5

Functions, even built-in ones, are objects in Python

EXERCISE 2 – 3. SHORT ANSWERS

1. What does this expression evaluate to?

>> { b : a for a, b in enumerate(“HELLO”) }

{'H': 0, 'E': 1, 'L': 3, 'O': 4}

EXERCISE 2 – 3. SHORT ANSWERS

2. What slice of the word “washington” will give the result of “ogisw”? (Give answer in the form [i:j:k])

```
>>test[-2:-11:-2]
```

```
>>test[8::-2]
```

EXERCISE 2 – 4. PROGRAMMING QUESTION

- Write a function reverse_dict() that will reverse keys and values such that the original values become the new keys to lists of one or more values that were the original keys. For example:
- {“bob”: 2, “greg”: 3, “joe”: 2, “tom”: 1, “dave”: 2, “stu”: 3, “mike”: 5}
- becomes:
- { 1: [“tom”], 2: [“bob”, “joe”, “dave”], 3: [“stu”], 5: [“mike”] }

EXERCISE 2 – 4. PROGRAMMING QUESTION

- Write a function reverse_dict() that will reverse keys and values such that the original values become the new keys to lists of one or more values that were the original keys.

```
def reverse_dict(input):
    result = defaultdict(list)
    for item, key in input.items():
        result[key].append(item)
    return result
```

EXERCISE 3 – 1. TRUE OR FALSE

1. Overloading virtual functions in C++ is an example of multiple dispatch. **T** **F**

   - Function overloading in C++ is done statically (static dispatch), at compile time.

EXERCISE 3 – 1. TRUE OR FALSE

2. You can declare an opaque data type on the stack.

T **F**

- You must know the size of a data type to be able to declare it on the stack, but an opaque data type is not defined, making it impossible to calculate its size.

EXERCISE 3 – 1. TRUE OR FALSE

3. Pure virtual functions are not necessarily pure functions.

T **F**

- Pure virtual functions can still have side effects such as modifying global variables, which makes it not always a pure function.

EXERCISE 3 – 1. TRUE OR FALSE

4. A virtual function overloading an operator is an example of dynamic dispatch. **T** **F**

- for example, virtual int operator+(int val) const;

EXERCISE 3 – 1. TRUE OR FALSE

5. Dynamically-typed interpreted language cannot implement early binding. **T** **F**

- Only compiled languages can legitimately implement early binding (name to address translation)

EXERCISE 3 – 1. SHORT ANSWERS

1. 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(food):
                print(str(food) + “ is delicious”)
```

EXERCISE 3 – 2. SHORT ANSWERS

1. …

```
class Animal:
        # may change this function in the future
        def eat(food):
                print(str(food) + " is delicious")
class Dog(Animal):
        def eat(self, food):
                super().eat(food)
                print("Wags its tail")
```

EXERCISE 3 – 2. SHORT ANSWERS

2. Write an expression with equivalent result as the following list comprehension, using only the map and filter function for control flow.

Expression: [str(v) for v in range(10, 100) if not (v//10+v%10)%7]

Result: ['16', '25', '34', '43', '52', '59', '61', '68', '70', '77', '86', '95']

EXERCISE 3 – 2. SHORT ANSWERS

2. Write an expression with equivalent result as the following list comprehension, using only the map and filter function for control flow.

```
def filterFunction(x):
    return not (x//10+x%10)%7
filteredList = filter(filterFunction, range(10,100))
mappedList = map(str, filteredList)
```

EXERCISE 3 – 2. SHORT ANSWERS

3. In Prototype-based programming, all objects have the type Object. The base object initially has no attribute. We wish to program with this paradigm in Python. Create a Person object out of the base object, followed by inheriting from the Person object to create a Student object. Finally, make an instance of Student. A Person object should have the data attributes: name, age, and gender, with a method called birthday() that increments age. A Student object should have the data attributes: id, gpa, and year. Create an instance of Student named “me” with your credential (can be fake). Choose suitable defaults for the prototypes.

EXERCISE 3 – 2. SHORT ANSWERS

```
class Object:
        pass

base = Object()
```

```
import copy
person = copy.deepycopy(base)
person.name = “”
person.age = 26
person.gender = “unknown”
def bday(self):
   self.age += 1
Object.birthday = bday
student = copy.deepycopy(person)
student.id = 0
student.gpa = 0
student.year = 1900
me = copy.deepcopy(student)
me.name = “jack”
me.gender = “male”
me.id = 123456789
me.gpa = 3
me.year = 2015
```

Questions?