National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2024/2025

# Recitation 4
# Data Abstraction

## Python

1. *Tuple* - `(value1, value2, ...)`

   A tuple is an immutable sequence of Python objects enclosed in parentheses and separated by commas.

2. Operations on tuples:
   (a) *len(x)* - Returns the number of elements of tuple $x$.
   (b) *element* in *x* - Returns `True` is *element* is in $x$, and `False` otherwise.
   (c) for *var* in *x* - Will iterate over all the elements of $x$ with variable $var$.
   (d) *max(x)* - Returns the maximum element in the tuple $x$.
   (e) *min(x)* - Returns the miniimum element in the tuple $x$.

## Problems

1. Evaluate the following expressions:

```
tup_a = (10, 12, 13, 14) #Creating tup_a
print(tup_a)

tup_b = ("CS1010X", "CS1231") #Creating tup_b
print(tup_b)

tup_c = tup_a + tup_b #Creating tup_c
print(tup_c)

len(tup_c)

14 in tup_a

11 in tup_c

tup_d = tup_b[0] * 4

tup_d[0]

tup_d[1:]

count = 0
```

```
for i in tup_a:
    count = count + i
print(count)

max(tup_a)

min(tup_a)

max(tup_c)

min(tup_c)
```

2. Write expressions whose values will print out like the following.

```
(1, 2, 3)



(1, (2), 3)



(1, (2,), 3)



((1, 2), (3, 4), (5, 6))
```

3. Write expressions to that will return the value 4 when the `x` is bound to the following values:

```
(7, 6, 5, 4, 3, 2, 1)



(7, (6, 5, 4), (3, 2), 1)



(7, ((6, 5, (4,), 3), 2), 1)
```

4. You found a holiday assignment at the Registar's Office. Your job is to write a program to help students with their scheduling of classes. You are provided with an implementation of the records for each class as follows:

```
def make_module(course_code, units):
    return (course_code, units)

def make_units(lecture, tutorial, lab, homework, prep):
    return (lecture, tutorial, lab, homework, prep)

def get_module_code(course):
    return course[0]

def get_module_units(course):
    return course[1]

def get_module_total_units(units):
    return units[0] + units[1] + units[2] + units[3] + units[4]
```

Each class (course) has a course code and an associated number of credit unit, e.g. for CS1101S, that's 3-2-1-3-3. Your job is now to write a schedule object to represent the sets of classes taken by a student. **Note:** Since `class` is a keyword in Python, we will use `course` as the variable representing the current class of interest.

(a) Write a constructor `make_empty_schedule()` that returns an empty schedule.

```
def make_empty_schedule():
```

Order of growth in time, space?

(b) Write a function `add_class` that when given a class and a schedule, returns a new schedule including the new class:

```
def add_class(course, schedule):
```

Order of growth in time, space?

(c) Write a function `total_scheduled_units` that computes the total number of units in a specified schedule.

```
def total_scheduled_units(schedule):
```

Order of growth in time, space?

(d) Write a function `drop_class` that returns a new schedule with a particular class dropped from a specified schedule.

```
def drop_class(schedule, course):
```

Order of growth in time, space?

(e) Implement a credit limit by taking in a schedule, and returning a new schedule that has total number of units is less than or equal to `max_credits` by removing classes from the specified schedule.

```
def credit_limit(schedule, max_credits):
```

Order of growth in time, space?

(f) **Homework:** Implement an improved version of `credit_limit` that will return a schedule with a total number of units is less than or equal to `max_credits`, but with the maximal number of classes. What is the order of growth of your solution? Is that the best you can do?