KIT100 Programming Preparation

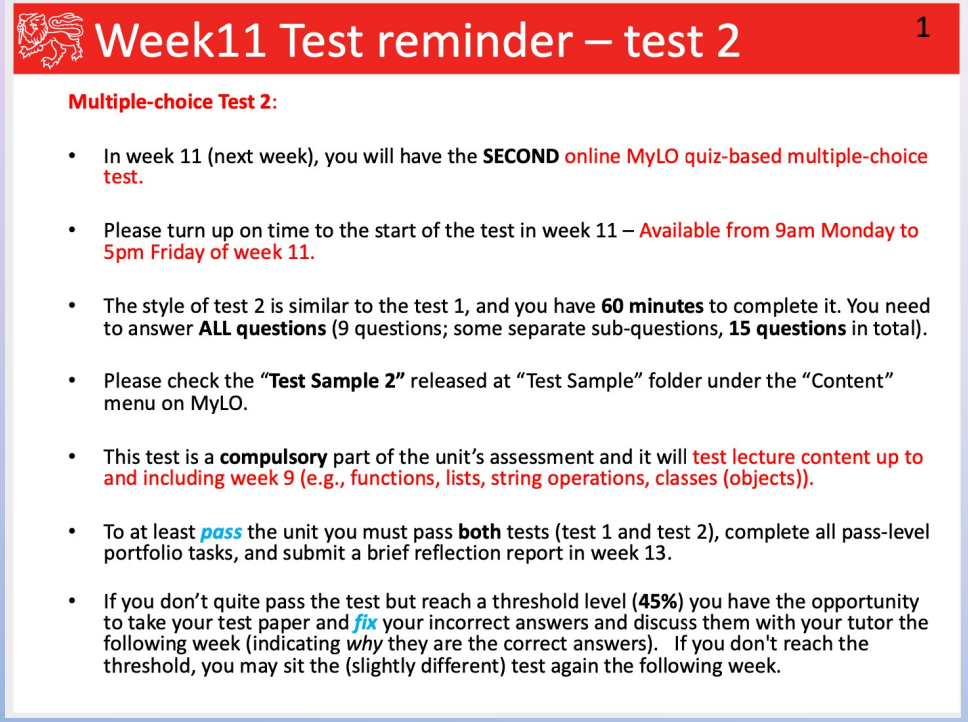
Tutorial Nine – Week 10

Today's Flow

- Test 2 Reminder
- Revise Python class and objects
- Walk through Portfolio Tasks 10.1CR, 10.2CR, and 10.3DN
- Walk through Test 2 Sample
- One-on-one session with me
  - to help you to resolve the programming concerns from the previous weeks

Test 2

- Don’t forget we will have Test 2 **next week** in MyLO.
- The style of this quiz will be very similar with the **Test 2 Sample**.



Week11 Test reminder – test 2

1

Multiple-choice Test 2:

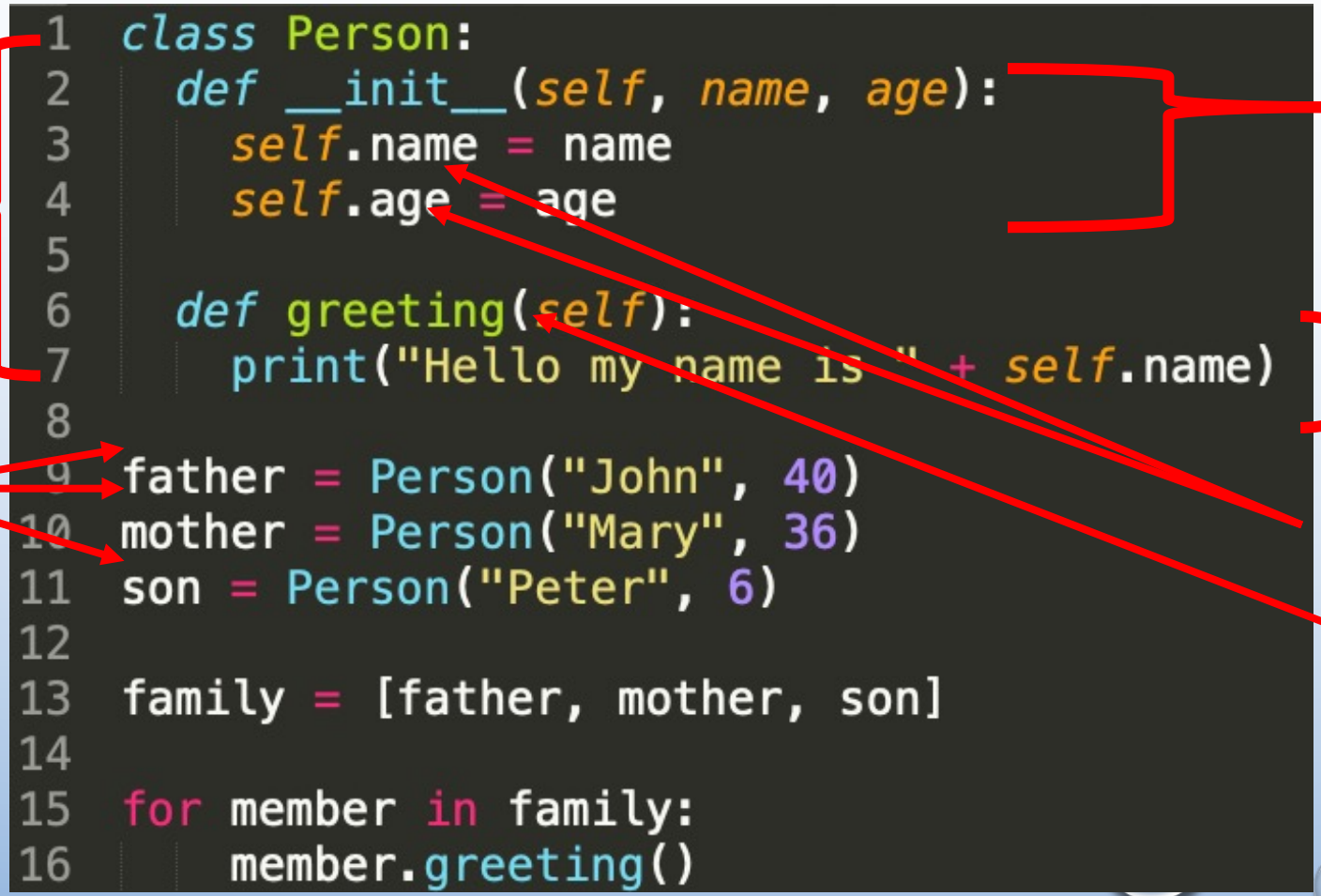
- In week 11 (next week), you will have the **SECOND** online MyLO quiz-based multiple-choice test.
- Please turn up on time to the start of the test in week 11 – Available from 9am Monday to 5pm Friday of week 11.
- The style of test 2 is similar to the test 1, and you have **60 minutes** to complete it. You need to answer **ALL questions** (9 questions; some separate sub-questions, **15 questions** in total).
- Please check the “**Test Sample 2”** released at “Test Sample” folder under the “Content” menu on MyLO.
- This test is a **compulsory** part of the unit’s assessment and it will test lecture content up to and including week 9 (e.g., functions, lists, string operations, classes (objects)).
- To at least ***pass*** the unit you must pass **both** tests (test 1 and test 2), complete all pass-level portfolio tasks, and submit a brief reflection report in week 13.
- If you don’t quite pass the test but reach a threshold level (**45%**) you have the opportunity to take your test paper and ***fix*** your incorrect answers and discuss them with your tutor the following week (indicating *why* they are the correct answers). If you don't reach the threshold, you may sit the (slightly different) test again the following week.

Python Class and Objects

- Python support object oriented programming
- Almost everything in python is an object.
- Class – a blueprint to create objects, object constructor. *(See Lecture9 pp.20-30 for more details on the syntax of classes)*.

```
1  class Person:
2      def __init__(self, name, age):
3          self.name = name
4          self.age = age
5
6      def greeting(self):
7          print("Hello my name is " + self.name)
8
9  father = Person("John", 40)
10 mother = Person("Mary", 36)
11 son = Person("Peter", 6)
12
13 family = [father, mother, son]
14
15 for member in family:
16     member.greeting()
```

Python Class and Objects



```
1  class Person:
2    def __init__(self, name, age):
3      self.name = name
4      self.age = age
5
6    def greeting(self):
7      print("Hello my name is " + self.name)
8
9  father = Person("John", 40)
10 mother = Person("Mary", 36)
11 son = Person("Peter", 6)
12
13 family = [father, mother, son]
14
15 for member in family:
16     member.greeting()
```

Class definition

Objects

__init__() function

- Constructor of a class
- Automatically executed when the class is being initiated.
- Use to assign value to class attributes

Class function

Class attributes

self parameter

> The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

Python Class and Objects

```
1  class Person:
2    def __init__(self, name, age):
3      self.name = name
4      self.age = age
5
6    def greeting(self):
7      print("Hello my name is " + self.name)
8
9    def setName(self, newName):
10     self.name = newName
11
12   def getName(self):
13     return self.name
14
15   def __str__(self):
16     return '{obj.name}, {obj.age}'.format(obj=self)
```

Set function

Get function

Override built-in string function

Python Class and Objects

```
1  class Dog:
2      def __init__(self, name, age):
3          self.name = name
4          self.age = age
```

The correct way to instantiate the `Dog` class is:

A. `bobby = Dog.__init__("Bobby", 3)`

B. `bobby = Dog()`

C. `bobby = Dog("Bobby", 3)`

D. `bobby = Dog.create("Bobby", 3)`

Python Class and Objects

```
1  class Dog:
2      def __init__(self, name, age):
3          self.name = name
4          self.age = age
```

The correct way to instantiate the `Dog` class is:

A. `bobby = Dog.__init__("Bobby", 3)`

B. `bobby = Dog()`

✓ C. `bobby = Dog("Bobby", 3)`

D. `bobby = Dog.create("Bobby", 3)`

Python Class and Objects

```
1  class Sales:
2      def __init__(self, id):
3          self.id = id
4          id = 100
5
6  val = Sales(123)
7  print (val.id)
```

What will be the output of this code snippet?

A. Syntax Error

B. 100

C. 123

D. val.id

Python Class and Objects

```
1  class Sales:
2      def __init__(self, id):
3          self.id = id
4          id = 100
5
6  val = Sales(123)
7  print (val.id)
```

What will be the output of this code snippet?

A. Syntax Error

B. 100

C. 123 ✓

D. val.id

10.1CR – Code Completion

Question 1: Your single line of code is to be inserted at line 2:

Code:

```
1 i = 0
2
3     print("Do you sell cheese cake",i,"here?")
4     i = i + 1
```

Output:

```
Do you sell cheese cake 0 here?
Do you sell cheese cake 1 here?
Do you sell cheese cake 2 here?
Do you sell cheese cake 3 here?
Do you sell cheese cake 4 here?
```

Question 2: Your single line of code is to be inserted at line 1:

Code:

```
1
2     print("This is a cheese cake shop sir",i)
```

Output:

```
This is a cheese cake shop sir 0
This is a cheese cake shop sir 1
This is a cheese cake shop sir 2
This is a cheese cake shop sir 3
This is a cheese cake shop sir 4
```

Question 3: Your single line of code is to be inserted at line 7:

Code:

```
1 cheeses = ["Original","Lemon","Strawberry","Peanut"
2 replies = ["Sorry", "No", "No", "Yes sir.. No"
3
4 j = 0
5 for i in cheeses:
6     print("Do you have any ",i,"?",sep='')
7
8     j = j + 1
```

Output:

```
Do you have any Original?
Sorry
Do you have any Lemon?
No
Do you have any Strawberry?
No
Do you have any Peanut?
Yes sir.. No
```

Question 4: Your two lines of code are to be inserted at lines 4 and 12:

Code:

```
1  cheeses = ["Original","Lemon","Strawberry","Peanut"]
2  print("Welcome to the cheese cake shop.")
3  choice = input("Enter cheese: ").lower()
4
5  while found != True:
6      for cheese in cheeses:
7          if choice.find(cheese) != -1:
8              found = True
9
10     if found == False:
11         print("No, we don't have that.")
12
13     else:
14         print("Yes, we have that!")
```

Output: (example user input shown after the `Enter cheese:` prompts)

```
Welcome to the cheese cake shop.
Enter cheese: Chocolate
No, we don't have that.
Enter cheese: Banana
No, we don't have that.
Enter cheese: Lemon
Yes, we have that!
```

Submission Details

Upload the following to the MyLO submission folder for this task:

1. Your python source code containing your answers to the questions above

- For easier marking, please add comment ("#code inserted") at the end of your inserted line.

10.2CR – Class Definition

Description:

The animal rescue facility next to the pharmacist in task 8.1PP has asked around and you have been recommended as good programmer to them. They need an animal inventory system to help keep track of which animals they have, and whether they have been adopted.

Task:

Write a program that defines a class called `Animal`.

The class should contain the following attributes (also known as *instance* variables):

- `name` (a string, the animal's adopted name)
- `animalType` (a string, the sort of animal, for example, *dog*, *cat*)
- `age` (an integer, the animal's estimated age)
- `adopted` (a boolean, whether the animal has been accepted for adoption yet)

Firstly, the Animal class should have an `__init__` method that creates and assigns values to these 4 attributes (`name,animalType,age, adopted`).

Then, the Animal class must also define the following 'set' and 'get' methods:

'set' methods:

- `setName(self,value)`
  Assign the animal's `name` to be `value`
- `setAnimaleType(self,value)`
  Assign the animal's `animalType` to be `value`
- `setAge(self,value)`
  Assign the animal's `age` to be `value`
- `setAdopted(self,value)`
  Assign the animal's adoption status, `adopted` to be `value` (`True or False`)

'get' methods:

- `getName(self)`
  returns the animal's `name`
- `getAnimalType(self)`
  returns the animal's type, i.e. `animalType`
- `getAge(self)`
  returns the animal's `age`
- `getAdopted(self)`
  returns the animal's adoption status, i.e. `adopted`

Finally, the `Animal` class should have an `__str__` method that asks the object to print itself regarding these 4 attributes (`name, animalType, age, adopted`).

Submission Details

Save the class to a file called `animal.py` and upload it to the MyLO submission folder for this task.

Assessment Criteria & Hints

A completed submission **must**:

1. Include comments about the program purpose and the author of the program (your name)
2. Define a class `Animal` at the start of the program
3. Define all 'set' and 'get' methods, as well as a correct __init__ method and a __str__ method
4. Use meaningful names for instance variables, starting with a lower case
5. Submit the source file

(See Lecture9 pp.22-30 for example code and explanation)

10.3DN – Class Definition and Usage

Description:

The Galactic Government learned of your amazing work with NASA and rockets, and you were their number one choice to develop an employee database for Galactic employees.

Task:

Write a class named `Employee` that holds the following data about an employee in four attributes: Employee name (`name`), staff identification number (`idNumber`), department (`department`), and the staff member's job title (`jobTitle`). Once you have written the Employee class, write **another** program that then imports and uses the Employee class to create **five** Employee objects to hold the following data:

Name	ID Number	Department	Job Title
Darth Vader	123456	Empire	Supreme Leader
Luke Skywalker	246810	Rebellion	Red 5 Leader
Han Solo	135790	Acquisitions	Chief Smuggler
Leia Organa	888888	Royalty	Rebel Leader
Chewbacca	111112	IT	CTO

The program should then display the employee information in a formatted, tabular style (similar to the example output below). Your output must query each of the Employee objects for their data, for example, use the `getName()` method for the employee's name. Recall to left-justify a string using print placeholders, use **"%-widths"**.

The `Employee` class should have an `__init__()` method that creates and assigns values to these 4 attributes (`name, idNumber, department, jobTitle`).
The `Employee` class must also define the following 'set' and 'get' methods:

'set' methods (even if you don't use them all):

- `setName(self,value)`
  Assign the employee `name` to be `value`
- `setEmployeeID(self,value)`
  Assign the employee's `idNumber` to be `value`
- `setDepartment(self,value)`
  Assign the employee's `department` to be `value`
- `setTitle(self,value)`
  Assign the employee's `jobTitle` to be `value`

'get' methods:

- `getName(self)`
  get the employee `name`
- `getEmployeeID(self)`
  get the employee's `idNumber`
- `getDepartment(self)`
  get the employee's `department`
- `getTitle(self)`
  get the employee's `jobTitle`

Finally, create **another** program called `employeeDatabase.py` that imports and uses the `Employee` class to create **five** Employee objects and print the output as shown below:

Example Output

```
Name            ID Number Department    Job Title
----            --------- ----------    ---------
Darth Vader     123456    Empire        Supreme Leader
Luke Skywalker  246810    Rebellion     Red 5 Leader
Han Solo        135790    Acquisitions  Chief Smuggler
Leia Organa     888888    Royalty       Princess
Chewbacca       111112    IT            CTO
```

Submission Details

Save the class source code to a file called `employee.py`, and the code that creates and displays employees to a file called `employeeDatabase.py` and upload them to the MyLO submission folder for this task.

Test 2 Sample

Student ID number:

_____

Pages: 6

Questions: 9

UNIVERSITY OF TASMANIA

KIT100 Programming Preparation

Test 2, Semester x, xxxx

SAMPLE

Time allowed: 60 minutes

Instructions:

- Write your student number in the box above. Answer ALL 9 questions by circling the correct answers provided in this test paper with a black or blue pen.
- This is a closed-book test and do not allow using Python software.
- This is a pass/fix/fail test, so there are no numerical marks. If you only make a small number of mistakes you will have the opportunity to correct them and explain your corrections to your tutor.