Python Programming

Class: COSC 1800

General Reminder

`r.randint(x,y)`	Generate a random integer between **x** and **y**(import random as r).
`input(msg)`	Prompt the user with **msg** and take the user input.
`type(var)`	Returns the type of var.
`type(n, b, d,)`	Dynamicaly create a class named **n**. This class inerits all classes in **b** (tuple). **d** is a dictionary containing attributes and member method.
`int(var)`	Convert `var` to a integer.
`float(var)`	Convert `var` to a float.
`str(var)`	Convert `var` to a string.
`len(var)`	Returns the length of a string or a list.
`pass`	Used to keep an indentention empty avoiding `IndentationError`.
`.copy(obj)`	
`.deepcopy(obj)`	

Basic Syntax

```
if conditon_1:
    # code if conditon_1 is true
elif conditon_2:
    # code if conditon_2 is true and conditon_1
    # is false
else:
    # code
```

Lists

```
lst1 = [e_1, e_2, e_3] # [e_1, e_2, e_3]
lst2 = 5 * [a] # [a, a, a, a, a]
lst3 = [a for i in range(3)] # [a, a, a]
print([1,3,5,7]) # [1, 3, 5, 7]
```

`list(var)`	Convert a set or tuple to a list.
`lst[i]`	Access the `i`th element in the list.
`.apend(a)`	Adds `a` to the end of the list.
`.insert(i,e)`	Insert element `e` at index `i`.
`.pop()`	Remove the last element, return the removed value.
`.pop(i)`	Remove element at index `i`, return the removed value.
`range(i)`	

Operators

Symbol	Name	Type
`+`	addition	Arithmetic
`-`	substraction	Arithmetic
`*`	multiplication	Arithmetic
`/`	division	Arithmetic
`%`	modulo	Arithmetic
`**`	power	Arithmetic
`//`	div	Arithmetic
`and`	logical and	Boolean
`or`	logical or	Boolean
`not`	logical not	Boolean
`in`	**in**	Membership
`==`	equal	Comparison
`!=`	not equal	Comparison
`>`	greater than	Comparison
`<`	less than	Comparison
`>=`	greater than or equal	Comparison
`<=`	less than or equal	Comparison

Error Handling

```
try:
    # risky operation
except ex:
    # runs if an exception of type ex is raised
else:
    # runs if no exception is raised
finally:
    # Runs regardless of what happens
```

```
for i in lst:
    # for each element in lst
while condition:
    # runs while condition is true
```

`raise exception`	Throw an error of type `exception`.
`assert c, msg`	Throw and error with the message `msg` if the condition `c` is false.
`BaseException`	Base class for exception.
`.add_note(note)`	add a note to an exeption, it is a member function of `BaseException`.

Object Oriented Programing(OOP)

Performence Tips