Python Programming

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()`	Creates a new object but nested objects still reference the original.
`.deepcopy()`	Creates a new object with completely new copies of all nested objects.

Operators

Symbol	Name	Type
+	addition	Arithmetic
-	subtraction	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

```python
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
```

```python
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`.

Data Structures

list	`[e_1, ...]`	ordered, changeable, duplicates.
tuple	`(e_1, ...)`	ordered, unchangeable, duplicates.
set	`{e_1, ...}`	unordered, unchangeable, no duplicates, unindexed.
dictionary	`{a_1:b_1, ...}`	ordered, changeable, no duplicates.

Typing

`Sequence`	ordered container supporting indexing and slicing
`List`	mutable ordered sequence, supports indexing an
`Tuple`	immutable ordered sequence, fixed-size.
`Set`	unordered collection of unique elements.
`Dict`	mapping of keys to values.
`Mapping`	abstract read-only mapping interface.
`MutableMapping`	mapping that supports mutation.
`Iterable`	can be iterated with for-loops.
`Iterator`	yields items on demand.
`Sized`	supports len().
`Hashable`	can be used as dict key or set element.
`Optional[T]`	either T or None.
`Union[A,B]`	value may be of type A or B.
`Any`	accepts any type.

sets Methods

`.add(e)`	Add `e` in the set.
`.update(lst)`	Add all elements from `lst` in the set.
`.remove(e)`	Remove `e` in the set.
`.union(lst)`	add all elements form `lst`
`.intersection(lst)`	keep only the elements that are both in itse
`.difference(lst)`	remove all element of `lst` in the set
`.symetric_differencelst()`	

lists Methods

```python
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]
```

`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(n)`	Create a list with all integers from `0` to `n`.

dictionary Methods

```python
dic = {"max":22, "ugo":40, "cyp":21}
```

```python
    dic["max"] # -> 22
```

Object Oriented Programing(OOP)

Performence Tips

Basic Syntax

```python
    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

    def function(a:type, b:type = value, ...) -> rType:
        # code
```