Python3 SL Review LIJIA XU (1/2)

Floats can be created by division on ints

2**5 → 32 | 9**(1/2) → 3.0 | 20//6 → 3

1,25%0,5 → 0,25 | use Backslashes for escape characters

""" ⇒ new lines automatically escaped

4*'2' → '2222' | int("2") + int("3") = 5 | a>Z → True

"" must be Int | use del remove var | float(input("Enter")) | 9==9.0 True

num=7 | and, or, not ← operators | == has higher than or

```
if num==5:
  print("is5")
elif num==7:
  print("is7")
else:
  print(not 5or7)
```

```
i=0
while True
  i=i+1
  if i==2:
    print('skip 2')
    continue
  if i==5:
    print("breaking")
    break
  print(i)
print('finished')
```

→ 1 skip 2 3 4 breaking finished

```
list=[1,'2',['1',2]]
print(list[2][0])
```
→ '1'

num=[1,2,3] num=[] ✱ List can have several different types

✱ in operator to chek item in list

print(1 in list) → True | print(4 not in list) → True

nums.index('one') | max(list) min(list) | list.count(obj) | list.remove(obj) | list.reverse

```
nums.append(4)
print(len(nums))
index=1 | num.insert(index,4)
#insert infront of index
```

nums = list(range(10)) → [0 ..... 9] or range(0,10)

list(range(2,6,2)) → [2,4] ✱ not include 6

```
for num in nums
  print(num)
for i in range(5)
  print("Hi")
```

DRY → Don't Repeat Yourself
WET → Write Everything twice or We Enjoy Typing

```
def add(x,y):
  return x+y
a=5
b=10
def do_twice(func,x,y)
  return func(func(x,y),func(x,y))
print(do_twice(add,a,b))
# outputs 30
```

```
import random #module
for i in range(5): #0to4
  value=random.randint(1,6)
  print(value)
# 5 vals from 1to6 #1to6
```

from math import pi,sqrt # import multi from module

from math import pi as pie
print(pie)

Raise Exceptions ⇒ raise ValueError

Exceptions: ImportError, IndexError, NameError, SyntaxError, TypeError, ValueError, ZeroDivisionError, OSError, AssertionError

```
try
  print(2/0)
except ZeroDivisionError:
  print('div zeroerror')
except ValueError, TypeError:
  print("error")
except: # catch all errors
  print('error')
finally: # run even with error
  print('must run')
```

assertion is a sanity-check

```
temp=-10
assert(temp>=0),"colder than absolute zero"
```

✱ file.close() ✱ allway close file

mytile= open("filename.txt","w") | cont=mytile.read()

"r" read mode, default
"w" write mode, rewrite
"a" append mode
"b" b in any mode, non-txt

len(open("text.txt").readlines())

```
for line in mytile
  print(line)
```

mytile.readlines() # return a list of lines

file.read(4) return another 4 bytes

"w" mode will create a file, if it does not already exist.

```
msg="Hello World!"
file= open("test.txt","w")
amount_written=file.write(msg)
print(amount_written)
file.close() # output 12
```

two safe ways to open file:

```
try:
  f=open('test.txt')
  print(f.read())
finally:
  f.close()
```

with open(test.txt) as f:

None, 0, [] and "" is False

```
ages={"A"=20,"B"=25}
print(ages["A"]) # 20
```

```
pairs={1:"apple",
  "A":[2,3,4],
  True:"okay"}
pairs.get(1,"A")
 # prints "okay"
pairs.get(87,"A")
 # prints [2,3,4]
pairs.get(55,"No result")
 # print "No result"
```

Tuples

```
words=("A","B","C") or ="A","B","C"
print(words[0]) # prints "A"
empty tuple: ⇒ tpl=()
```

```
nums= [1, 2, 3, 4, 5]
print(nums[1:3]) # ⇒ [2,3)
nums[:3] → # [1,2,3]
nums[3:] → # [4,5]
```
✱ slicing can happen to tuples

```
nums[::2] ⇒ # [1,3,5]
nums[:4:2] ⇒ # [1,3]
nums[::-1] ⇒ # [5,4,3,2,1]
```

```
cubes=[i**3 for i in range(5)]
# [0,1,8,27,64]
evens=[i**2 for i in range(10) if i**2%2==0]
# [0,4,16,36,64]
```

```
nums=[4,5,6]
msg="Number:{0}{1}{0}".format(nums[0],nums[1])
a="{x},{y}".format(x=5,y=12)
```

```
print(",".join([1,2,3]))
# "1,2,3"
print("1,2,3".split(","))
# [1,2,3]
```

```
nums=[55,44,33,22,11]
if all([i>5 for i in nums])
  print('all larger 5')
if any([i%2==0 for i in nums])
  print(at least one even)
```

```
for v in enumerate(nums)
  print(v)
# (0,55) .....(4,11)
```

Text Analyzer

```
def count_char(text,char):
  count=0
  for c in text:
    if c==char:
      count +=1
  return count
filename=input("Enter a name:")
with open(filename) as f:
  text = f.read()
for char in "abcdefghij......z":
  perc = 100*count_char(text,char)/len(text)
  print("{0}-{1}%".format(char, round(perc,2))
```

pure func: return a value; only depends on their arguments

anonympus #lambda

```
print((lambda x: x**2)(-2))
square = lambda x: x**2
print(square(-2))
```

map

```
nums = [1,2,3]
res= list(map(lambda x:x+5, nums))
# [6,7,8]
```

```
filter | nums=[1,2,4,7]
res=list(filter(lambda x: x%2==0, nums))
or [x for x in nums if x%2==0]
# [2,4]
```

Generators (recomend)

```
def numbers(x):
  for i in range(x):
    if i%2==0:
      yield i
print(list(numbers(11)))
# [0,2,4,6,8,10]
```

```
def countdown():
  i=5
  while i>0
  yield i
  i-=1
for x in countdown
  print(x)
```

Decorators

```
def decor(func)
  def wrap():
    print("=====")
    func()
    print("=====")
  return wrap
def print_text()
  print("Hello world!")
decorated = decor(print_text)
decorated()
```
or
```
@decor
def print_text():
  print("Hello world!")
```

Sets: nums.add(-7) nums.remove(3)

```
num_set={1,2,3,4,5}
word_set=set(["A","B","C"])
print(3 in num_set)
print("spam" not in word_set)
```

first={1,2,3,4,5,6}
second={4,5,6,7,8,9}

first | second # {1...9}
first & second {4,5,6}
first - second {1,2,3}
first ^ second {1,2,3,7,8,9}
second-first {8,9,7}

✱ tuple might represent a dictionary key, because immutable

```
class Animal:
  def __init__(self,color,legs):
    self.color=color
    self.legs = legs
class Dog(Animal):
    def bark(self)
      print "wang!"
bob = Dog("Brown",4)
bob.bark()
```

Magic Methods are special methods which have __ at the beginning and end

__sub__	__floordiv__	__and__
__mul__	__mod__	__xor__
__truediv__	__pow__	__or__

__lt__	<
__le__	<=
__eq__	==
__ne__	!=
__gt__	>
__ge__	>=
__len__	len()
__getitem__	indexing
__setitem__	assign vals
__delitem__	del index val
__iter__	in, for loops
__contains__	in

```
class A:
  def spam(self)
    print(1)
class B:
  def spam(self)
    print(2)
    super.spam()
B().spam()
# 2
  1
```

```
class Vector2D:
  def __init__(self,x,y):
    self.x=x
    self.y=y
  def _add_(self,other)
    return Vector2D(self.x+other.x, self.y+other.y)
```

"we are all consenting adults here"
weakly private method ⇒ singule score in front
✱ from module_name import Not import with _

strong privat method __
access: _Spam__privatemethod

```
class Spam:
  __egg=7
  def print_egg(self):
    print(self.__egg)
s = Spam()
s.print_egg() #7
print(s._Spam__egg) #7
print(s.__egg) #error
```

Class Methods: cls, self can be changed @classmethod

```
class Rectangle:
  def __init__(self,width,height)
    self.width=width
    self.height=height
  def calculate_area(self)
    return self.width*self.height
  @classmethod
  def new_square(cls, side_length):
    return cls(side_length, side_length)
square= Rectangle.new_square(5)
```

```
class Pizza: #static Methods
  def __init__(self.toppings):
    self.toppings=toppings
  @staticmethod
  def validate_topping(topping):
    if topping=="pineapple":
      raise ValueError("No pineapples!")
    else:
      return True
ingredients =["A","B","C"]
if all(Pizza.validate_topping(i) for i in ingredients):
  pizza=Pizza(ingredients)
```

property can make read only

```
@property
def a_allowed(self)
  return False
```

Property has setter getter

@a-allowed.setter

property example, Regulation, check Pythonicness page 2