Python for Programmers

## Welcome

### Welcome

### Python 2 vs 3

### Mac Setup

### Windows setup

## Numbers

### Intro

### Variables and ints

Variable do not need to be declared. Like matlab juts introduce them

### Floats

### Math

### Advanced Math

import math  # need for pi

5\*\*3  # \*\* is exponential like in Fortran

5 // 3 # integer division

5 % 3   # remainder

round(math.pi,2)

int(math.pi)

# \_ gives most recent result

2\*\_

int(pi)

float(8)

### Conversion

### Outro

## Strings

# can use single or double quotes

'hello'

"hello"

# use 3 quotes of either type for multi-line strings

'''hello

world'''

"""hello

world"""

# can use strings as comments

# + is the concatination operator

print("hello"+" "+'world!')

# \* is used for repition of a string

print(9\*"9")

# printing variables

age = 30

print("My age is {}".format(age))

print("My age is {} and pi is {}".format(age, 3.1415))

"Hello".upper()

"Hello".lower()

"Hello world".title()

len("hello")

s="Hi my name is Bob".replace("Bob","Jim")

print(s)

s[0]

s[-1]

s[-17]   # gives same as s[0]

s[16]   # gives same as s[-1]

s[-18]    # gives an error

s[17]    # gives an error

# slices. s[n:m] goes from s[n] to s[m-1]  !

s[0:2]

s[6:10]

s[3:-4]

s[:10]  # same as 0:9

s[3:]   # same as 0:len

s[3:0]  # gives "" empty string

s[-1:]  # final character

## IF statements

# booleans

True

False

sunny = False

# use indentation instead of brackets!

if sunny:

    print("Sunny!")

    print("Yes Sunny!")

else:

    print("cloudy")

print("done")

# also have elif for convenience

x = -11

if x > 0:

    print("positive!")

elif x == 0:

    print("zero")

else:

    print("negative")

# python has ability to tell if between two nums!

y = 9

if 5 < y < 10:

    print("Tween")

#  and, or, not keywords

a=6.3

if a % 2 == 0 and 7 >= a >= 5:

    print("Even and between 7 and 5")

if not False:

    print("not")

# None is liek null in Java and Javascript

if not None:

    print("not")

## PROJECT 1 – GRADE CALC

pointspossible = 100

score = 84

percentage = score / pointspossible

studentname = "Bill"

print("Percentage is {}".format(percentage))

if 0 <= percentage < .60:

lettergrade="F"

elif .60 <= percentage < .70:

lettergrade="D"

elif .70 <= percentage < .80:

lettergrade="C"

elif .80 <= percentage < .90:

lettergrade="B"

else:

lettergrade="A"

print("{} {}".format(studentname, lettergrade))

## User Input

Capture a line of text

age=input("What is your age?: \n")

print("Your age is "+age+" years old!")

type of a variable

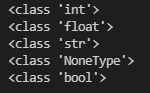
print( type(14) )

print( type(14.5) )

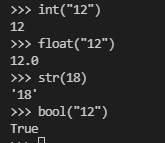
print( type(“14”) )

print( type(None) )

print( type(True) )



Type conversion is performed via int(), float(), str() etc



Exception handling

try:

    age = int(input("enter age:"))

    print("Valid age!")

except Exception:

    print("You need to provide a valid number")

## Functions

Simple function definition

def hello():

    print("Hello World")

hello()

function with paramters

def add(x1,x2):

    print(x1+x2)

add(3,4)

named parameters

def sub(x1,x2):

    print(x1-x2)

sub(4,3)

sub(x2=4,x1=3)

default parameters

def pname(name="nobody"):

    print(name)

pname("Ron")

pname()

**return value**

return causes end of function execution

def addtwo(num1, num2):

    return num1 + num2

print( addtwo(3,6) )

no return value means that the None object is returned

def hey():

    print("hey")

print( hey()   )

## Lists, Tuples, Sets

Type is **list**

games = ["Mario Bros 3", "Earthbound", "Pilotwings", "Mario Party"]

print(games)

print(type(games))

print(games[1])

games.insert(0, "Zelda")

print(games)

games[1] = 'Mario Bros 1'

print(games)

del(games[2])

print(games)

games = ["Zelda", "Mario Bros 3", "Earthbound", "Pilotwings", "Zelda", "Mario Party"]

# remove will remove the FIRST element that matches

games.remove("Zelda")

print(games)

print(len(games))

# **tuples** use parantheses.  both size and values are immutable!

tuple = ("Zelda", "Mario Bros 3", "Earthbound", "Pilotwings", "Zelda", "Mario Party")

print(tuple)

#to create an tuple of one element, you need to include a trailing comma

y = [ "hello" ]   # array

w = ( "hello" ) # value,, not a tuple

z = ( "hello", )  # tuple

shoes = ("Spizikes","Air Force 1","Curry 2","Melo 5")

# + operator does concatination for both arrays and tuples

# \* operator replicates

q = ["hello"] + 2\*[ "there" ]

def appendtotuple(thetuple, value):

    x = ( value, )

    t =  thetuple + x

    return t

print(appendtotuple(shoes,"loafers"))

**sets**: use curly braces {}. Empty set is set() not {}, which is an empty dictionary

myset= { "hello", "there "}

print(myset)

myset.discard("there")

myset.add("hi")

print(myset)

myset = set(games)

print(myset)

## Loops

# rang(10) -> class of type range. iterates over 0 to 9

for x in range(10):

    print(x)

numbers = [76, 83, 16, 69, 52, 78, 10, 77, 45, 52, 32, 17, 58, 54, 79, 72, 55, 50, 81, 74, 45, 33, 38, 10, 40, 44, 70, 81, 79, 28, 83, 41, 14, 16, 27, 38, 20, 84, 24, 50, 59, 71, 1, 13, 56, 91, 29, 54, 65, 23, 60, 57, 13, 39, 58, 94, 94, 42, 46, 58, 59, 29, 69, 60, 83, 9, 83, 5, 64, 70, 55, 89, 67, 89, 70, 8, 90, 17, 48, 17, 94, 18, 98, 72, 96, 26, 13, 7, 58, 67, 38, 48, 43, 98, 65, 8, 74, 44, 92]

odds = set([])

for n in numbers:

    if n % 2 == 1:

        odds.add(n)

print(odds)

# while loop

age = 27

while age <= 30:

    print(age)

    age += 1

n=0

y=1

while y <= 1000000000:

    n += 1

    y = 2\*\*n

print("n={}",n)

# break

for x in range(10):

    print(x)

    if (x == 5):

        break

# continue

for x in range(10):

    if (x == 5):

        continue

    print(x)

# else runs after loop finishes, but not if break eneded it

for x in range(10):

    if (x == 5):

        continue

    print(x)

else:

    print("All done")

# else won't run here

for x in range(10):

    if (x == 5):

        break

    print(x)

else:

    print("All done")

nums = [99, 20, 30, 35, 16, 49, 39, 11, 69, 48, 85, 32, 10, 47, 24, 80, 37, 21, 3, 99, 13, 11, 23, 12, 40, 50, 24, 14, 10, 62, 21, 24, 55, 57, 38, 55, 83, 63, 34, 31, 15, 26, 82, 47, 37, 14, 64, 72, 90, 39, 70, 50, 67, 61, 23, 28, 30, 13, 87, 58, 80, 62, 15, 49, 33, 7, 38, 2, 92, 76, 80, 18, 6, 25, 22, 25, 91, 9, 37, 83, 46, 98, 69, 3, 40, 6, 48, 1, 63, 51, 32, 19, 77, 74, 22, 75, 41, 19, 27, 82, 60, 6, 1, 55, 5, 71, 18, 84, 47, 16, 1, 8, 41, 6, 17, 100, 62, 36, 45, 32, 4, 33, 68, 15, 2, 92, 50, 54, 34, 12, 17, 16, 74, 95, 2, 61, 75, 12, 6, 39, 28, 18, 30, 39, 8, 34, 62, 31, 57, 8, 69, 19, 71, 70, 40, 79, 76, 96, 84, 76, 85, 4, 40, 64, 45, 11, 46, 100, 56, 9, 86, 5, 78, 81, 18, 70, 76, 46, 85, 69, 64, 88, 17, 91, 49, 93, 18, 29, 38, 42, 77, 63, 46, 32, 83, 88, 48, 68, 89, 80]

n = 0

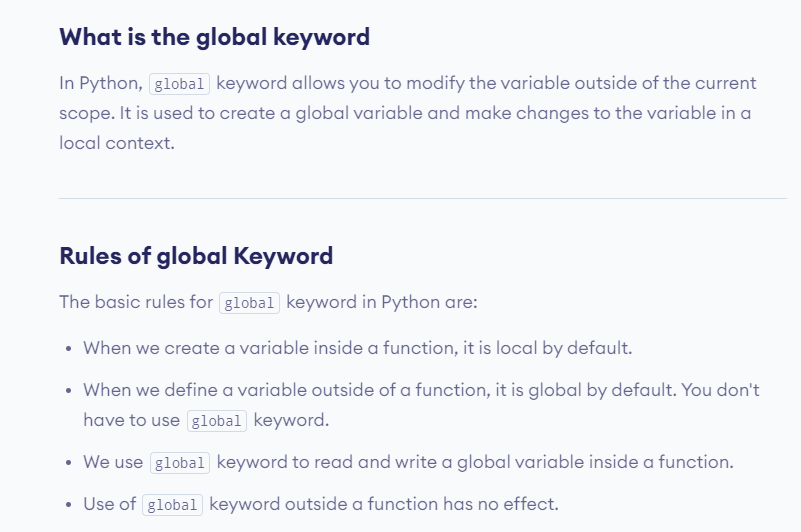
for x in nums:

    if (x == 68): break

    n += 1

print(n)

## PROJECT 2: Hangman



import time

lives = 10

stringword = "Apple"

word = list(stringword.lower())  # converts to a list of letters

winner = False

incorrectletters = []

guess = ["\_"] \* len(word)

def printscore():

    print("\n\nLives: ", str(lives))

    print("Incorrect Letters: ", end="")

    for i in range(len(incorrectletters)):

        if i == len(incorrectletters) - 1:

            print(incorrectletters[i], end="")

        else:

            print(incorrectletters[i], end=",")

    print("")

    print("Guess: ", end="")

    for i in range(len(guess)):

       print(guess[i], end="")

    print("")

def askforletter():

    letter = input("Please guess a letter: ").lower()

    correct = False

    for i in range(len(word)):

        if letter == word[i]:

            guess[i] = letter

            correct = True

    if correct:

        print("You got one!")

    else:

        print("Sorry there is no '"+letter+"'.")

        incorrectletters.append(letter)

        global lives

        lives -= 1

    time.sleep(1)

def checkwinner():

    if "\_" not in guess:

        global winner

        winner = True

while lives > 0 and not winner:

    printscore()

    askforletter()

    checkwinner()

if lives <= 0:

    print("You lost. The word was "+ stringword +".")

else:

    print("Congrats! The word is '"+stringword+"'. You won with " + str(lives) + " lives left!" )

bugs:

* Doesn’t check is entered string is a single letter (len(letter)==1 and letter in “a-z”
* Doesn’t check is already guessed

## Dictionaries

#give age sfor each dog

from itertools import count

dogs = { "Fido": 8, "Sean": 17, "One Woof Sam": 19 }

print(dogs)

print(dogs["Fido"])

# any object can be key and can mix object types in keys

cats = {

    "Felix": 8,

    "Boots": 8,

    13: 2

    }

print(cats)

words = ["PoGo","Spange","Lie-Fi"]

definitions = ["Slang for Pokemon Go","To collect spare change, either from couches, passerbys on the street or any numerous other ways and means","When your phone or tablet indicates that you are connected to a wireless network, however you are still unable to load webpages or use any internet services with your device"]

cooldictionary = {

    words[0]: definitions[0],

    words[1]: definitions[1],

    words[2]: definitions[2],

    }

print(cooldictionary.keys())

print(cooldictionary.values())

print(" ")

for key in dogs:

    print(key)

for key in dogs.keys():

    print(key)

for value in dogs.values():

    print(value)

for item in dogs.items():

    print(item)

dabools = [False, False, False, False, True, False, True, False, True, False, False, False, False, False, True, False, True, True, False, False, True, False, False, False, False, False, True, False, False, False, False, False, True, False, False, False, False, False, False, True, True, False, False, True, True, True, True, True, False, True, True, False, True, False, False, True, False, False, False, False, True, True, False, False, True, True, True, False, True, True, False, False, False, False, True, True, False, True, True, False, True, True, True, False, False, False, True, True, False, True, False, True, False, True, True, True, False, True, False, False, True, True, False, True, False, False, True, False, True, False, True, False, True, True, True, True, False, False, True, True, False, False, False, False, True, True, True, False, False, True, True, True, True, True, False, True, True, False, False, True, True, False, False, True, True, True, True, True, False, False, True, True, False, False, True, True, False, False, False, False, False, True, True, True, True, True, False, True, True, False, False, True, True, True, True, True, False, False, True, True, True, True, True, False, True, True, True, True, True, False, False, True, False, False, False, True, True, True, False, True, False, False, True, False, False, True, True, False, False, False, False, True, False, False, True, True, False, True, True, False, False, True, False, True, True, False, True, False, True, False, True, False, False, True, False, True, False, False, False, True, True, False, False, False, True, True, False, False, True, True, True, True, True, False, True, True, True, True, False, False, True, False, True, True, True, True, True, False, False, False, True, True, False, True, True, True, False, False, True, True, False, True, True, True, True, False, True, True, False, False, False, True, True, True, True, True, False, False, False, True, False, False, False, True, False, True, True, True, True, False, False, False, True, False, False, False, False, True, True, False, True, False, False, True, False, False, False, True, True, True, False, False, True, True, False, False, True, True, False, False, True, False, True, False, True, True, True, False, True, True, True, True, False, True, True, False, True, True, False, True, False, True, False, True, True, False, True, False, True, False, False, True, True, False, True, False, False, True, True, True, False, True, True, True, False, False, False, True, False, True, True, False, True, False, True, True, False, True, True, False, False, True, False, False, False, True, True, False, True, True, False, False, True, True, True, False, False, True, False, True, True, True, True, False, False, True, True, False, False, True, True, False, True, False, True, True, True, True, True, False, False, False, True, False, False, True, False, False, True, True, False, False, False, False, False, False, True, False, True, False, False, True, True, False, True, True, True, False, True, True, False, False, True, False, False, True, True, False, True, False, True, False, False, True, False, True, True, False, False, True, True, False, False, True, False, False, False, False, True, False]

counter = {

    True: 0,

    False: 0

}

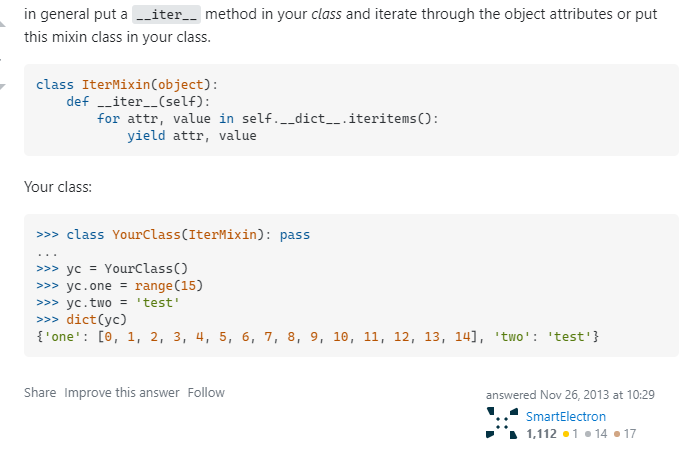
for bool in dabools:

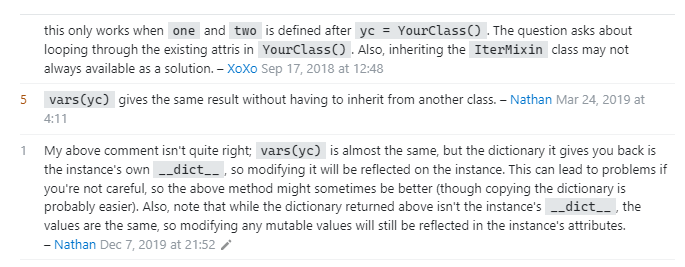
    counter[bool] += 1

print(counter)

## Classes







class Dog:

    info = "A Dog is a mammal" # class variable

    def barkhello():

        print("Woof Woof")

print(Dog.info)

dog1 = Dog()

dog1.name = "Fido" #instance variable

print(dog1.info)

print(dog1.name)

Dog.info = "N/A"

print(Dog.info)

print(dog1.info)

Dog.barkhello()

# dog1.barkhello()  # can;t call class method on instance

dog1.\_\_class\_\_.barkhello()

dog2 = Dog()

dog3 = Dog()

class Car:

    definition = "Something with 4 wheels and an engine"

class Dog:

    info = "A Dog is a mammal" # class variable

    def \_\_init\_\_(self, name, age, furcolor):   # can name 1st parameter anything you want. it will be the instance

        print("inside init")

        # instance vars

        self.name = name

        self.age = age

        self.furcolor = furcolor

    pass # end of class defs

    # now these are instance methods

    def barkhello(self):

        print("Woof Woof")

    def barkgoodbye(self):

        print("Woof!!")

dog1 = Dog("Fido", 3, "Black")

print(dir(dog1))

print(vars(dog1))

class BullDog(Dog):

    pass # end of class defs

    def growl(self):  # instance methods must always have self parameter

        print("GRRRWL")

class Hound(Dog): pass   # pass is used when you don;t want other propertieds

dog4 = BullDog("Fido", 3, "Black")

print(dog4.growl())

print(dog4.barkhello())

class IterMixin(object):

    def \_\_iter\_\_(self):

        for attr, value in self.\_\_dict\_\_.iteritems():

            yield attr, value

class Car:

    def \_\_init\_\_(this, year, make, model):

        this.year = year

        this.make = make

        this.model = model

    def age(self): return 2016 - self.year

mycar = Car(2000, "Honda", "Accord")

print(mycar.age())

class Car:

    def \_\_init\_\_(self,year, make, model):

        self.year = year

        self.make = make

        self.model = model

    def age(self):

        return 2016 - self.year

class Mustang(Car):

    def \_\_init\_\_(self, year):

        super().\_\_init\_\_(year, "Ford", "Mustang")

mine = Mustang(1960)

print(vars(mine))

# use     super().\_\_init\_\_('Mammal') to call class method on parent class

## Advanced Python

None object

print(None)

print(type(None))

def first(s):

    if len(s) > 0: return s[0]

    else: return None

print(first('hello'))

print(first(''))

def square(n):

    t = type(n)

    if t == int or t == float:

        return n\*\*2

    else: return None

importing files

**turnup.py**

def emojify(text):

    print("💩" + text + "💩")

import turnup

turnup.emojify("Nick is cool!")

file i/o

f = open("motto.txt","r")

print(f.read())

f = open("motto.txt","w")

f.write(“Hello there”)

## PORJECT 3: Word Counter

## Random Twitter Follower

## BONUS SECTION