**- EX 1 + 2**

# -\*- coding: utf-8 -\*-

#above code to confirm using Unicode UTF-8 in your scripts without a problem.

**print** "Hello World!"

**print** "I'd much rather you 'not'." #single quote inside double quotes str

**print** 'How many ## I "said" do not touch this.' # hash sign & double quotes inside quotes or even double

**- EX 3**

**print** "I will now count my chickens:"

**print** "Hens", **25** + **30** / **6** #comma puts to separates different print

**print** "Now I will count the eggs:"

**print** "Roosters", **100** - **25** \* **3** % **4** # % modulo mean X divided by Y with J remaining from X, ex: 10 divided by 3 with 1 remaining, The result of % is the J

**print** **3** + **2** + **1** - **5** + **4** % **2** - **1** / **4** + **6**

**print** "Is it true that 3 + 2 < 5 - 7?"

**print** **3** + **2** < **5** - **7**

**print** "What is 3 + 2?", **3** + **2**

**print** "What is 5 - 7?", **5** - **7**

**print** 'Ahmed'\***3**

**print** round(**1.7**), round(**1.4**)

**print** "Is it greater?", **5** > - **2**

**print** "Is it less or equal?", **5** <= - **2**

**- EX 4**

cars = **100**

space\_in\_a\_car = **4.0**

drivers = **30**

passengers = **90**

cars\_not\_driven = cars - drivers

cars\_driven = drivers

carpool\_capacity = cars\_driven \* space\_in\_a\_car

average\_passengers\_per\_car = passengers / cars\_driven

**print** "There are", cars, "cars available."

**print** "There are only", drivers, "drivers available."

**print** "There will be", cars\_not\_driven, "empty cars today."

**print** "We can transport", carpool\_capacity, "people today."

**print** "We have", passengers, "to carpool today."

**print** "We need to put about", average\_passengers\_per\_car, "in each car."

**- EX 5**

my\_name = 'Zed A. Shaw'

my\_age = **35**

my\_height = **74**

my\_weight = **180**

my\_eyes = 'Blue'

my\_teeth = 'White'

my\_hair = 'Brown'

# %s for str, %d for num, %r for the expression as it is

# will be replaced in order

**print** "Let's talk about %s." % my\_name

**print** "He's %d inches tall." % my\_height

**print** "He's %d pounds heavy." % my\_weight

**print** "He's got %s eyes and %s hair." % (my\_eyes, my\_hair) # in order

**print** "His teeth are usually %s depending on the coffee." % my\_teeth

# this line is tricky, try to get it exactly right

**print** "If I add %d, %d, and %d I get %d." % (

my\_age, my\_height, my\_weight, my\_age + my\_height + my\_weight) # could be mixed

**- EX 6 + 7**

x = "There are %d types of people." % **10**

binary = "binary"

do\_not = "don't"

y = "Those who know %s and those who %s." % (binary, do\_not)

**print** x, # comma at end of print order bring next print in same line

**print** y

# using str inside str

**print** "I said: %r." % x

**print** "I also said: '%s'." % y

hilarious = False

joke\_evaluation = "Isn't that joke so funny?! %r"

**print** joke\_evaluation % hilarious

w = "This is the left side of..."

e = "a string with a right side."

**print** w + e

**- EX 8**

formatter = "%r %r %r %r"

**print** formatter % (**1**, **2**, **3**, **4**)

**print** formatter % ("one", "two", "three", "four")

**print** formatter % (True, False, False, True)

**print** formatter % ("I had this thing."

,"That you could type up right.",

"But it didn't sing."

,"So I said goodnight."

) # make several lines

**- EX 9 + 10**

days = "Mon Tue Wed Thu Fri Sat Sun"

months = "\nJan\nFeb\nMar\nApr\nMay\nJun\nJul\nAug\tso on" # \n print in new line & \t print tap

**print** "Here are the days: ", days

**print** "Here are the months: ", months

**print** """

There's something going on here.

With the three double- quotes.

We'll be able to type as much as we like.

Even 4 lines if we want, or 5, or 6.

"""

**print** "I am 6'2\" tall." # put \ before double qoute to tel python just print

**print** 'I am 6\'2" tall.' # put \ before qoute to tel python just print

**print** "I am 6'2\\ \" tall." # double backslash to tell python print it i need it in my txt

tabby\_cat = "\tI'm tabbed in."

persian\_cat = "I'm split\non a line."

backslash\_cat = "I'm \\ a \\ cat."

fat\_cat = """

I'll do a list:

\t\* Cat food

\t\* Fishies

\t\* Catnip\n\t\* Grass

"""

**print** tabby\_cat

**print** persian\_cat

**print** backslash\_cat

**print** fat\_cat

**- EX 11 + 12**

**print** 'How old are you?', # comma has been added to continue print in same line

age=raw\_input() # diff raw\_input() type it raw inside str & input() type it num

**print** 'how tall are you?',

tall=int(raw\_input()) # int() convert str to integer incase str contain nums only

# float() convert str to floating no. in case str contain nums only

weight=input('How much do you weight?') # prompt msg to enter the data

**print** 'so your are %r years old, %r tall, %r weight.'%(age, tall, weight)

# python -m pydoc raw\_input <- typr this in normal terminal and read some comment from pydoc

**- EX 13**

**from** **sys** **import** argv

script, first, second, third = argv

**print** "The script is called:", script # on command line gives the file name

**print** "Your first variable is:", first

**print** "Your second variable is:", second

**print** "Your third variable is:", third

# at the command line type the file name.py followed by 3 variable name and

# output will will be the above printed str followed by your variable names.

# below is another way to add names to the str

**print** raw\_input("The script is called:")

**print** raw\_input("Your first variable is:")

**print** raw\_input("Your second variable is:")

**print** raw\_input("Your third variable is:")

**- EX 14**

# another ex on script unpacking and raw\_input

**from** **sys** **import** argv

script, user\_name = argv

prompt = '> '

**print** "Hi %s, I'm the %s script." % (user\_name, script)

**print** "I'd like to ask you a few questions."

**print** "Do you like me %s?" % user\_name

likes = raw\_input(prompt)

**print** "Where do you live %s?" % user\_name

lives = raw\_input(prompt)

**print** "What kind of computer do you have?",

computer = raw\_input(prompt)

**print** """

Alright, so you said %r about liking me.

You live in %r. Not sure where that is.

And you have a %r computer. Nice.

""" % (likes, lives, computer)

**- EX 15**

**from** **sys** **import** argv

script, filename = argv

**print** 'Here we are printing %r' % filename

text = open(filename) # assign class to file name

**print** text.read() # use text.read() function to print it

text.close()

# using raw.input()

text\_again=open(raw\_input('Tell me file name to read it for you:> '))

**print** text\_again.read()

text\_again.close()

**- EX 16**

**from** **sys** **import** argv

script, filename = argv

**print** "We're going to erase %r." % filename

**print** "If you don't want that, hit CTRL- C (^C)."

**print** "If you do want that, hit RETURN."

raw\_input("?")

**print** "Opening the file..."

target = open(filename, 'w') # 'w' create file if not exist and start writing from begging

# 'a' create file if not exist and stat writing from end

**print** "Truncating the file. Goodbye!"

raw\_input("?")

target.truncate()

**print** "Now I'm going to ask you for three lines."

line1 = raw\_input("line 1: ")

line2 = raw\_input("line 2: ")

line3 = raw\_input("line 3: ")

**print** "I'm going to write these to the file."

target.write(line1+'\n'+line2+'\n'+line3)

**print** "And finally, we close it."

target.close()

**- EX 17**

**from** **sys** **import** argv

**from** **os.path** **import** exists

script, from\_file, to\_file = argv

**print** 'copying from %s to %s' % (from\_file, to\_file)

file\_open=open(from\_file)

data=file\_open.read()

**print** 'the input file is %d bytes long' % len(data)

**print** 'does the output file exists? %r' % exists(to\_file) # true or false

raw\_input('if you ready press enter>')

output\_file=open(to\_file, 'w')

output\_file.write(data)

output\_file.close()

file\_open.close()

**- EX 18 + 19**

**def** **print\_args**(\*d): # like script in ex 15 - 17 - \* gives multiple arguments

a1, a2, a3, a4 = d # assign as u WANT but consider the arguments inside () when you print

**print** 'arg1: %r, arg2: %r, arg3: %r, arg4: %r' % (a1,a2, a3, a4)

ahmed='ahmed'

num=**60**

**def** **print\_again**(arg1, arg2):

**print** 'arg1: %r, arg2: %r' % (arg1,arg2)

**def** **print\_one**(arg1):

**print** 'arg1: %r' % arg1

**def** **print\_none**():

**print** 'nothing to print'

print\_args(ahmed\***5**, 'ola', 'bilal',(**50**+**10**)\*num) # variable and nums could be used

print\_again(ahmed\***3**,"shaaban")

print\_one("AHMED!!")

print\_none()

**- EX 20**

**from** **sys** **import** argv

script, filename = argv

**def** **print\_all**(file):

**print** file.read()

**def** **rewind**(file):

file.seek(**0**)

# VERY IMP: think abt it like a tape need to start from beginning after u read \*listen\*

**def** **print\_a\_line**(line\_count, file):

**print** line\_count, file.readline()

# read line after line and if u need from bigging u must use above .seek(0)

# .readline() each time Used it print \n after it read line to prevent it add ,(comma) after the print

current\_file=open(filename)

**print** 'lets print it all first: \n'

print\_all(current\_file)

**print** 'lets rewind: \n'

rewind(current\_file)

**print** 'print line by line: \n'

current\_line = **1**

print\_a\_line(current\_line, current\_file)

current\_line += **1**

print\_a\_line(current\_line, current\_file)

current\_line += **1**

print\_a\_line(current\_line, current\_file)

**- EX 21**

**def** **add**(a, b):

**print** "ADDING %d + %d" % (a, b)# incase i hashed# it, it stop the print msg only but the value still same

**return** a + b

**def** **subtract**(a, b):

**print** "SUBTRACTING %d - %d" % (a, b)

**return** a - b

**def** **multiply**(a, b):

**print** "MULTIPLYING %d \* %d" % (a, b)

**return** a \* b

**def** **divide**(a, b):

**print** "DIVIDING %d / %d" % (a, b)

**return** a / b

**print** "Let's do some math with just functions!"

age = add(**30**, **5**) # once func called it give the print if it contain print

height = subtract(**78**, **4**) # if u pressed caps lock + tab it will change first charc to uppercase

weight = multiply(**90**, **2**)

iq = divide(**100**, **2**)

**print** "Age: %d, Height: %d, Weight: %d, IQ: %d" % (age, height, weight, iq)

**print** "Here is a puzzle."

what = add(age, subtract(height, multiply(weight, divide(iq,add(**1**,**1**))))) # func starting as inside out (from deepest to first)

**print** "That becomes: ", what, "Can you do it by hand?"

**- EX 22 + 23 + 24**

**def** **secret\_formula**(started):

jelly\_beans = started \* **500**

jars = jelly\_beans / **1000**

crates = jars / **100**

**return** jelly\_beans, jars, crates # check the order despite off the var names

start\_point = **10000**

beans, jars, crates = secret\_formula(start\_point)

# here we made script with order and assigned to func

# var name could be changed but order is important

**print** "With a starting point of: %d" % start\_point

**print** "We'd have %d beans, %d jars, and %d crates." % (beans, jars, crates) # we calling the assigned script

**print** "We can also do that this way:"

**print** "We'd have %d beans, %d jars, and %d crates." % secret\_formula(start\_point)

# another way to calling it

**def** **family**(a,b,c):

dad=a

mom=b

son=c

**return** dad,mom,son

**print** family('ahmed', 'ola', 'bilal')

x,y,z=family('ahmed', 'ola', 'bilal')

**print** 'dad is %s, mom is %s, son is %s' % (x,y,z)

**print** 'dad is %s, mom is %s, son is %s' % family('ahmed', 'ola', 'bilal')

**- EX 25**

**def** **break\_words**(stuff):

"""This function will break up words for us."""

# ''' documentation you import .py this sentence appeares in help(filename)

words = stuff.split() # create list and defulat parameter is space

**return** words

**print** break\_words('ahmed ola bilal')

**def** **sort\_words**(words):

"""Sorts the LIST from NUMS, space, a, b, .... z."""

**return** sorted(words)

**print** sort\_words(['ahmed', 'ola', 'bilal', **19**,**30**,**44**])

**def** **print\_first\_word**(words):

"""Prints the first word after popping it off."""

word = words.pop(**0**) # take off the given index from a list if no index givien it consider last elemnet '-1'

**print** word

**print** words # the list will be cutted

print\_first\_word(['ahmed','ola','bilal'])

# list.remove('str') delete the given str from list (error will apply if no such str)

**def** **print\_last\_word**(words):

"""Prints the last word after popping it off."""

word = words.pop(-**3**) #[ 0, 1 , 2]

**print** word #[-3, -2,-1]

**print** words

print\_last\_word(['ahmed','ola','bilal'])

**def** **sort\_sentence**(sentence):

"""Takes in a full sentence and returns the sorted words."""

words = break\_words(sentence)

**return** sort\_words(words)

**print** sort\_sentence('ahmed ola bilal')

**def** **print\_first\_and\_last**(sentence):

"""Prints the first and last words of the sentence."""

words = break\_words(sentence)

print\_first\_word(words)

print\_last\_word(words)

**def** **print\_first\_and\_last\_sorted**(sentence):

"""Sorts the words then prints the first and last one."""

words = sort\_sentence(sentence)

print\_first\_word(words)

print\_last\_word(words)

**print** "Let's practice everything."

**print** 'You\'d need to know \'bout escapes with \\ that do \n newlines and \t tabs.'

poem = """

\t1The lovely world 1

with logic so 1 firmly planted

cannot discern \n the needs 1 of love

nor comprehend passion from 1 intuition

and requires 1 an explantion

where there 1 is none.

"""

""" # using quot or double quot without assigning to var used as documentaion

"""

**print** poem.find('1',poem.find('1', poem.find('1', poem.find('1')+**1**)+**1**)+**1**) # find(1, comes after find(1)+1)

# find('str', start index, end index) return -1 if doesnt find the require

**print** poem.index('1',poem.index('1', poem.index('1')+**1**)+**1**)

# same just like .find() but it raise an error if didnt find the required

# print poem.replace('1', '2' , 2 ) # print str and replace('str', with given 'str', 2 times only)

**print** poem.count('1') # check how many count of given str

**print** "--------------"

**print** poem

**print** "--------------"

five = **10** - **2** + **3** - **5**

**print** "This should be five: %s" % five

**def** **secret\_formula**(started):

jelly\_beans = started \* **500**

jars = jelly\_beans / **1000**

crates = jars / **100**

**return** jelly\_beans, jars, crates

start\_point = **10000**

beans, jars, crates = secret\_formula(start\_point)

**print** "With a starting point of: %d" % start\_point

**print** "We'd have %d jeans, %d jars, and %d crates." % (beans, jars, crates)

start\_point = start\_point / **10**

**print** "We can also do that this way:"

**print** "We'd have %d beans, %d jars, and %d crabapples." % secret\_formula(start\_point)

sentence = "All good\tthings come to those who wait."

words = break\_words(sentence)

sorted\_words = sort\_words(words)

print\_first\_word(words)

print\_last\_word(words)

print\_first\_word(sorted\_words)

print\_last\_word(sorted\_words)

sorted\_words = sort\_sentence(sentence)

**print** sorted\_words

print\_first\_and\_last(sentence)

print\_first\_and\_last\_sorted(sentence)

**- EX 26 + 27 + 28**

**print** True and True # T

**print** False and True # F

**print** **1** == **1** and **2** == **1** # F

**print** "test" == "test" # T

**print** '\n'

**print** **1** == **1** or **2** != **1** # T

**print** True and **1** == **1** # T

**print** False and **0** != **0** # F

**print** True or **1** == **1** # T

**print** '\n'

**print** "test" == "testing" # F

**print** **1** != **0** and **2** == **1** # F

**print** "test" != "testing" # T

**print** "test" == **1** # F

**print** '\n'

**print** not (True and False) # T

**print** not (**1** == **1** and **0** != **1**) # F

**print** not (**10** == **1** or **1000** == **1000**) # F

**print** not (**1** != **10** or **3** == **4**) # F

**print** '\n'

**print** not ("testing" == "testing" and "Zed" == "Cool Guy") # T

**print** **1** == **1** and not ("testing" == **1** or **1** == **0**) # T

**print** "chunky" == "bacon" and not (**3** == **4** or **3** == **3**) # F

**print** **3** == **3** and not ("testing" == "testing" or "Python" == "Fun") # F

**print** '\n'

**print** 'test' <= "test" # T # QUOTE OR DOUBLE Q DOESNT AFFECT

**print** 'test' <= 'testI' # T

**print** 'test' <= 'test-ING' # T # CHECK SAME ORDER AND THEN COMPARE

**print** 'test' <= 'ING-test' # F # order is important

**print** '\n'

**print** True and 'aaaaaaa' # 1st check right side if true check left side then return last one checked

**print** False and 'aaaaaaa' # 1st check right side found false then return false without checked left side

**print** 'aaaaaaa' and True ,'\t', 'aaaaaaa' and False # it hase to check the left side

**print** '\n'

**print** True or **1** # the oppist in 'or' 1st check right side if rtuen true and doesnt check left side

**print** False or **1** # 1st check right side if false check left side and return last one checked

**print** **1** or False,'\t',**1** or True # doesnt have to check the left side

**print** any and True , any and False

**- EX 29 + 30**

people = **30**

cars = **40**

buses = **15**

# in order 'elif','else' are connected to first 'if' and will apply after 'if' result...

**if** cars > people:

**print** "We should take the cars."

**elif** cars > people:

**print** "We should not take the cars."

**else**:

**print** "We should not decide."

# if elif changed to if connection lose with 1st if, and start new if code.

**if** cars > people:

**print** "We should take the cars."

**if** cars > people:

**print** "We should not take the cars."

**else**:

**print** "We should not decide."

**- EX 31 + 32 + 33**

the\_count = [**1**, **2**, **3**, **4**, **5**]

fruits = ['apples', 'oranges', 'pears', 'apricots']

change = [**1**, 'pennies', **2**, 'dimes', **3**, 'quarters']

**for** number in the\_count:

**print** "This is count %d" % number

# same as above

**for** fruit in fruits:

**print** "A fruit of type: %s" % fruit

# also we can go through mixed lists too

# notice we have to use %r since we don't know what's in it

**for** i in change:

**print** "I got %r" % i

# we can also build lists, first start with an empty one

elements = []

# then use the range function to do 0 to 5 counts

**for** i in range(**0**, **6**):

**print** "Adding %d to the list." % i

# append is a function that lists understand

elements.append(i)

**print** elements

# now we can print them out too

**for** i in elements:

**print** "Element was: %d" % i

**print** range(**1**,**11**)\***2** # creates fixed list that could be appended only

fam=['ahmed', 'ola', 'bilal']

fam[**0**]=['ahmed'] # to append a list inside list element must be changed from str to list

fam[**0**].append('mahmoud')

**print** fam

i = **0**

**while** True :

**if** i == **20**:

exit(**0**) # to exit loop or func

**print** i

i+=**1**

**- EX 36**

list1 = range(**0**,**5**)

**del** list1[**1**] # use del to delete element in a list

**print** list1

**for** x in list1:

**print** x

**print** x # can see that var x in for loop still has value if its in a fucn value will be lost

**del** x # del use to delete var

#print x # if printed error 'x' is not defined

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

mynum = **1** # its a global var

**def** **func1**():

**global** mynum # writing global means confirm to change in the global var

mynum = **2**

**def** **func2**():

mynum = **3** # global var changed inside func only but outisde still as it is

**print** mynum # first global var

func1()

**print** mynum # var has been changed inside func and value confirmed

func2()

**print** mynum # var has been changed inside func but value not confimred

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

**assert** **2** < **5**, 'problem' # assert or assert not expression , then error 'msg' i want

# assert not 2 < 5 , 'wow its working'

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

mylist = [x+**1** **for** x in range(**3**)] # creates list as equation 'x+1' for every x in range(3) will be added to 1

**print** mylist

new = [[x+**1**] **for** x in range(**3**)] # creates an individual list for each element

**print** new

T\_or\_F\_list = [ x >= **3** **for** x in range(**5**)] # print a result of true or false instead in the new list

**print** T\_or\_F\_list

generator = (x+**1** **for** x in range(**3**))

# generator is a list but for loop purpose only and it use () not []

**for** i in generator:

**print** i

**print** generator

# Return sends a specified value back to its caller whereas Yield can produce a sequence of values.

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

a=**5**

**print** eval('a + 50') # eval run and equation inside str

**exec**('a = 50 + 5') # assign var inside str

**print** a

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

#if 5 != 4:

#raise Exception ('mmm it looks like assert')

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

**for** letter in 'Python':

**if** letter == 'h':

**continue** # the oppiset of break where when the rule apply doent let the loop continue do-

# ing what its doing and force it to continue the next loop.

**print** 'Current Letter :', letter

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

# else statement could be used with while & for loop just like if 'used when loop become false'

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

''' How python think: mylist.append('str')

1st look for mylist ex:var, list, func, class

2nd find a func called append in it

and python read it like append(mylist , 'str')

mylist = str\_func's module

append() = func called append

'''

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

dict = {'Name':'Zara', 'Age':**7**, 'Class':'First'}

dict['Age'] = **8**; # update existing entry

dict['School'] = "DPS School"; # Add new entry

#del dict['Name']; # remove entry with key 'Name'

#dict.clear(); # remove all entries in dict

#del dict ; # delete entire dictionary

**print** "dict['Age']: ", dict['Age']

**print** "dict['School']: ", dict['School']

**print** dict

**- EX 37 +38 +39**

region = { # elements here called key

'Riyadh' : 'RYD' ,

'Makkah' : 'MK' ,

'Madina' : 'MD' ,

'Estren' : 'EST' ,

'Tabuk' : 'TB'

}

cities = {

'MK' : 'jeddah' ,

'MD' : 'yanbu' ,

'EST' : 'damam'

}

cities['RYD'] = 'in ry'

cities['TB'] = 'in tb'

**print** 'citi in Makkah is:', cities['MK']

**print** 'another way:' , cities[region['Makkah']]

**print** 'Riyadh abberivation is:' , region['Riyadh']

#using dict in loop, transfer dict to list by .items()

**for** reg , abber in region.items():

**print** 'region is %s and abber is %s and beggest city is %s' % (reg, abber, cities[abber])

**print** region.items()

dict\_to\_list = region.items()

**print** dict\_to\_list[**0**]

**print** region.get('Riyadh') # .get() here just return the value of named key

**print** region.get('Hail', None ) # .get(new key , value or None) return here a value or None for key that doesnt exist but dont add it to dict

**print** region.keys() # print all the keys name in a list

**print** region

**- EX 40**

# Class is a blueprint that could be used as a design to be clone

# remember using (python file name argument1 argument2 ...) as a script

# its look like it but using it inside python

# 1st make the initial definition of arguments and var that will be used

**class** **Song**(): # better to name the Class as 1st upper letter

**def** **\_\_init\_\_**(self, words): # self here represent class name

self.lyrics = words # as Song.lyrics

**def** **sing\_me\_a\_song**(self):

**for** line in self.lyrics:

**print** line

**def** **count\_lines**(self):

**print** len(self.lyrics)

# making a clone called happy\_bday

happy\_bday = Song(["Happy birthday to you",

"I don't want to get sued",

"So I'll stop right there"])

bulls\_on\_parade = Song(["They rally around the family",

"With pockets full of shells"])

happy\_bday.sing\_me\_a\_song() # calling func sing\_me..

**print** len(happy\_bday.lyrics) # using the self.lyrics outside func

happy\_bday.count\_lines()

bulls\_on\_parade.sing\_me\_a\_song()

**print** len(bulls\_on\_parade.lyrics)

bulls\_on\_parade.count\_lines()

**- EX 44**

**class** **Parent**(object): # can write object here to use super()

**def** **\_\_init\_\_**(self, family\_name, color):

self.last = family\_name

self.color = color

**def** **implicit**(self):

**print** "PARENT implicit()"

**class** **Child**(Parent): # whats written inside () is the parent class and could be multiple parents (parent1, parentr2, ...)

**def** **implicit**(self): # making same func like parent mean to override the parent

**print** 'Child override parent implicit()'

super(Child, self).implicit() # force current class to use parent func super

**print** 'parent return to override'

#pass # pass here means no block to be run and just pass it

dad = Parent('shaaban', 'semi white')

son = Child('shaaban', 'semi white') # to give more arguments u need to override parent.\_\_init\_\_ by creat its own init

dad.implicit()

son.implicit()

**print** son.last

# composition is dont use inhirtance and just use the other class func

# in above case in class Child when calling implicit() from parent we type

# self.parent.implicit() and self here represent child

**- os & open**

'''

open('filename', access mode , buffering )

access mode :

- 'r' (read only) the defualt access mode, pointer at the beginning of file

- 'r+' (reading & writing) pointer at the beginning of file but doesnt create new file if not exist

- 'w' (writing only) Overwrites if file exist or create new file not folder, always remove old date to start clean

- 'w+' (writing & reading) Overwrites if file exist or create new file

- 'a' (appending file) pointerat the end of file

- 'a+' (appending & reading)

'''

new\_text = open('test text.txt', 'w+') # w & w+ mode require write() as it remove all date in file if exist

# it better to use r+ as it doesn’t remove date but if data were given it will act just like w mode

**print** new\_text.name # return file name

**print** new\_text.mode # return the access mode of the file

**print** new\_text.closed # return F if file still open, T if file closed

new\_text.write('again this a test wrinting inside text file by write method through python\n')

# in this situation pointer at the end use .seek() to return back to first

new\_text.writelines(['line2\n', 'line2\n', 'line3\n']) # add severl line from list

**print** new\_text.tell() # return my current position place

new\_text.seek(**0**) # chagne pointer position to the given byt num

**print** new\_text.tell()

**print** new\_text.read() # .read() start reading from the current position to the given byt num. ex: read(10)

new\_text.seek(**0**)

**print** new\_text.next() # read only 1 line and add empty line after reading then if method repeated it reads next line 7 so on

**print** new\_text.next() # if lines finish it gives error (next()used in loops)

new\_text.seek(**0**)

**print** new\_text.readline() # read only 1 line and add empty line after reading then if method repeated it reads next line 7 so on

# when lines finish it gives empty

# new\_text.truncate() # earse the rest of data after the current position

new\_text.seek(**0**)

**print** new\_text.readlines() # return a list contain each line in 1 element

new\_text.close() # this close the file again

**print** new\_text.closed

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

**import** **os**, **sys**

os.remove('renamed file.txt') # delete file with (given name)

os.rename('test text.txt', 'renamed file.txt') # rename (given name , to new name)

#os.renames('old file', 'new folder/new file name') # move and chage file name (from, to )

os.mkdir('new folder') # creates new dirc (given folder name)

os.rmdir('new folder') # delete empty folder only

'''

with open('new folder/test.py' , 'w') as f :

f.write('#wow .. if u can read this, this is amazing')

f.close() '''

**print** os.getcwd() # return current working dirc address

# better to save the getcwd to var original\_path = os.getcwd()

# os.chdir(r'C:\Users\secretary-1\Udacity') # change working dirc to (given address)

# os.fchdir(r'open file path') # it change cwd to the dirc of the openned file

**print** os.access('test text.txt', os.F\_OK) # access('path', os.F\_OK) check if file exist

# check online for more option and other methods to change files & folder permissions

# chown() chflags() chmod() and a lot other methods

fd = os.open(r'renamed file.txt',os.O\_RDWR) # open file as read & write

os.write(fd, 'add line by os.write')

os.lseek(fd,**0**,**0**) # just like .seek() but give ('openned file', 0, 0)

**print** os.read(fd,**5555**) # read ('file openned', max num of byt)

os.close(fd)

**print** os.listdir(r'C:\Users\secretary-1\Udacity\my-exercise\my work') # return a list with all files & folder name in the given path

os.startfile('renamed file.txt')

**print** os.path.split(r'C:\Users\secretary-1\Udacity\my-exercise\my work\renamed file.txt')

# os.path.split(r'path') split the path to 2 arguments tree and file name (or last folder name)

**- time & calendar**

**import** **time**

**print** time.time() # return secs sinsce 12 am 1/1/1970 this called epoc

**print** time.localtime() # takes nothing for current time ot given secs and current 9 tupletime (wday# start from 0 monday)

**print** time.gmtime() # takes secs gives GM time by localtime() formate

**print** time.ctime() # same asctime() but takes secs then gives regulare formatted time

**print** time.mktime([**2009**, **1**, **1**, **0**, **0**, **0**, **0**, **0**, **0**]) # opposit if localtime() as takes 9 tupletime and return secs

**print** time.asctime([**2009**, **1**, **1**, **0**, **0**, **0**, **2**, **0**, **0**]) # takes nothing for current or 9 tupletime and

# return regualr formatted time (day name mm dd 24\_hr\_time yyyy)

time.sleep(**0**) # sys sleep for given secs then contiue printing after the secs

**print** time.strftime('%H',time.localtime()) # takes str only for current or str and given 9 tupletime return time formated as u given

# % a'A day b'B month y'Y yr(shrt'full) names

#%d/%m/%y day month year numbers = %x for date but mm/dd/yy %X represent time only

# %U wk# in year %j day of year %z time zone name

# %c date and time %H hr in 24

**print** time.strptime('21 november 2015 00:05:05', '%d %B %Y %X') # takes 2 str and return 9 tupletime

# strptime(str'must be euqal to given %formate' , str'given %formate')

t = [**2009**, **1**, **1**, **0**, **0**, **0**, **0**, **0**, **0**]

**print** time.strftime('%d/%m/%Y\t%X',t)

**from** **datetime** **import** date

**print** date.today().year

**print** date.today().month

**print** date.today().day

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

**import** **calendar**

calendar.calendar(**2000**) # print calendar of givien year

**print** calendar.month(**2018**, **10**) # print out month calendar

**print** calendar.isleap(**2016**) # check leap year return True or False

**print** calendar.leapdays(**2000**,**2018**) # print no. of leap days btw 2 given years

**print** calendar.weekday(**2018**,**3**,**14**) # return weekday code of the given date (0=monday)

**- webbrowser & urllib**

**import** **webbrowser**

**from** **webbrowser** **import** Chrome # used to call Chrome class by using .register('name', Class name)

#webbrowser.open(r'www.google.com',0) # open url

#webbrowser.open\_new(r'www.google.com')

**print** webbrowser.register('chrome', Chrome('chrome')) # here we confirm using chrome by useing Chrome class

#google = webbrowser

#google.open(r'www.google.com')

'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

**import** **urllib** # in urllib must use the full address name http://www......

**print** urllib.urlopen(r'http://www.google.com').read(**5**) # .read() & .readline() & .readlines() could be used

urllib.urlretrieve(r'http://www.google.com','text.html') # copy HTML of the given site to a given name file

**print** urllib.getproxies() # return a proxy if it was given

response = urllib.urlopen(r'http://www.google.com')

**print** response.geturl()

headers = response.info()

**print** 'DATE :', headers['date']

**print** urllib.urlopen(r'http://www.wdylike.appspot.com/?q=shot and shot').read()