413-513 Notes- Week 2

Review:

<https://docs.python.org/3/library/stdtypes.html>

print \

("hey man wat up")

x=2.2

print("x =",end="");

print ("%d"%x) #**trunc** data!!

input("\nEnter to quit");

alt.

print (int(x)) #**trunc** data!!

Division differences – IMPORTANT for processing, printing detail!

\*(Data types important for operator choice!)

**True Division**

True division is where the result is always the real floating-point quotient, regardless of **operand type**.

>>> 1 / 2 # returns real quotient

0.5

>>> 1.0 / 2.0 # returns real quotient

0.5

**Floor Division**

A new division operator ( // ) always truncates the fraction and rounds it to the next smallest whole number toward the left on the number line, regardless of the operands' numeric types.

>>> 1.0 // 2.0 # floors result, returns float

0.0

>>> -1 // 2 # negatives move left on number line

-1

7.1.3.1. Format Specification Mini-Language

[**https://docs.python.org/2/library/string.html#formatspec**](https://docs.python.org/2/library/string.html#formatspec)

7.1.6. Deprecated string functions (page bottom)

ex.

string.**zfill**(*s*, *width*)

Pad a numeric string *s* on the left with zero digits until the given *width* is reached. Strings starting with a sign are handled correctly.

>>> '99'.zfill(5)

'00099'

>>> '99'.rjust(5,'0')

'00099'

if you want the opposite:

>>> '99'.ljust(5,'0')

'99000'

width = 10

x = 5

print "%0\*d" % (width, x)

> 0000000005

**[ Source Code examples from Scripts link ]**

**-Bin packing algorithm**

**-Time converter algorithm**

New material:

[ **if statements** ]

if’s (multi branch, nested), Conditional Expressions Operator (CEO)

examples/probs.

- check for greater of 3 values a,b,c

- bonus issuance

- fica

- The CEO

**if condition:**

**x = true\_value**

**else:**

**x = false\_value**

**x = true\_value if condition else false\_value**

The **condition expression** in the **middle** is evaluated **first**,

the true\_value expression is evaluated only if the condition was true.

the false\_value expression is only evaluated when the condition is false (**else** part).

\*use in print statement for evals

Calculating gross pay follows:

**hours= 40.0 #make float**

**rate = 10**

**overtime = 0**

**'''without ceo'''**

**if hours <= 40:**

**gross= rate \* hours**

**else:**

**overtime = hours - 40**

**gross= rate \* 40 + (1.5 \* rate) \* overtime**

**print(gross)**

**'''ceo example'''**

**gross= rate \* hours if hours <= 40 else rate \* 40 + (1.5 \* rate \* (hours - 40))**

**print(gross)**

Question? Is there switch case in python?

[ **Working with numeric types / math functions** ]

All [numbers.Real](https://docs.python.org/2/library/numbers.html#numbers.Real) types ([int](https://docs.python.org/2/library/functions.html#int), [long](https://docs.python.org/2/library/functions.html#long), and [float](https://docs.python.org/2/library/functions.html#float)) also include the following operations:

| **Operation** | **Result** |
| --- | --- |
| math.trunc(x) | *x* truncated to Integral |
| round(x[, n]) | *x* rounded to n digits, rounding ties away from zero. If n is omitted, it defaults to 0. |
| math.floor(x) | the greatest integral float <= *x* |
| math.ceil(x) | the least integral float >= *x* |

**Loops** ~ logic

uses:

-salary (use of forever loop, use of **break** statement)

-prime num tester (use of loop)

* **The range function simplifies the process of writing a for loop**
  + range returns an iterable object
    - Iterable: contains a sequence of values that can be iterated over
* **range characteristics:**
  + One argument: used as ending limit
  + Two arguments: starting value up to but *not* including ending limit
  + Three arguments: third argument is step value

ex. **range (10, 0, -1)**

Fixed loop example --*For loop from 0 to 2, therefore running 3 times.*

for x in range(0, 3):

print "We're on time %d" % (x)

*While loop from 1 to infinity, therefore running infinity times.*

x = 1

while True:

print ("To infinity and beyond! We're getting close, \

on %d now!" % (x))

x += 1

#salary problem

Sammy begins a new job with an annual salary of $ 52,500 for his first year of employment, a guarantee of a 5 % salary increase each for his second, third and

fourth year and a guarantee of a 9.5 % increase for his fifth and sixth year. Write a code that will output Sammy’s salary schedule for his first six years of employment. Round all amounts to two decimal places.

Write your program such that your output will appear similar to the tabular format

shown below.

|  |  |
| --- | --- |
| **Year** | **Salary** |
| 1 | $ **52500.00** |
| 2 | $ |
| 3 | $ |
| 4 | $ |
| 5 | $ |
| 6 | $ |

'''salary problem'''

#housekeeping - declare & initialize vars

salary = 52500

#print iniital header

print ("\t", "Year".ljust(16), "Salary".ljust(16))

#print 1st year salary

print ("\t", "1".center(3).ljust(16), "${:,.2f}".format(salary).ljust(16))

#set loop range to years (2-6) of the salary increases

for x in range(2,7):

if x < 5:

salary \*= 1.05

print ("\t", str(x).center(3).ljust(16), "${:,.2f}".format(salary).ljust(16))

else:

salary \*= 1.095

print ("\t", str(x).center(3).ljust(16), "${:,.2f}".format(salary).ljust(16))

**[** **Lab 1 – pcode ]**

#Defining variables

totalCost = 0

totalAnnualCost = 0

#prompt input from user

app1 = input ('Please enter the first appliance name: ')

cost1 = int (input ('Please enter the cost per kW - hr in cents: '))

usage1 = int (input ('Please enter annual usage (kW - hr): '))

totalCost1 = totalCost + (cost1/100) \* usage1

app2 = input ('Please enter the second appliance name: ')

cost2 = int (input ('Please enter the cost per kW - hr in cents: '))

usage2 = int (input ('Please enter annual usage (kW - hr): '))

totalCost2 = totalCost + (cost2/100) \* usage2

totalAnnualCost = totalCost1 + totalCost2 + totalCost3 + totalCost4 + totalCost5 + totalCost6

#Output

print ('')

print ('Energy Cost Worksheet')

print ('')

print ('Appliance\t', 'Cost\t\t', 'Annual Usage')

print (' \t', '(per kW - hr)\t', '(kW - hr)')

print (app1,'\t', cost1,' cents','\t', usage1)

print (app2,'\t', cost2,' cents','\t', usage2)

print ('')

print ('The total cost of the annual usage is $ %4.2f' % totalAnnualCost)

Similar example of lab 1using loops

**MPG w. loop problem**

'''Gas mileage data compilation'''

#housekeeping - declare & initialize vars

applist = [] #create list

totalMPG = 0.0 #make float type

cont = "y" #this one is used to stop the loop

count = 1

#loop runs until you say 'n(o)'

while cont == "y": #while user hasn't said to stop

miles = float(input("Please enter vehicle #'s" + str(count) +

" miles traveled (fractional portions allowed) "))

gallons = float(input("Please enter vehicle #" + str(count) +

" gallons consumed (fractional portions allowed) "))

MPG = miles / gallons #calculate mpg - use true division

totalMPG += MPG #accumulate mpg

applist.append([miles, gallons, MPG]) #add it to the list

count = count + 1

cont = input("Would you like to include data for another vehicle? (Y/N): ").lower()

#format header for display

print ("\nGas Mileage Cost Worksheet\n")

print ("\t", "Miles".ljust(16), "Gallons".ljust(16), "MPG".ljust(16))

print ("\t", "Traveled".ljust(16), "Used".ljust(16))

#format detail for display

for i in applist:

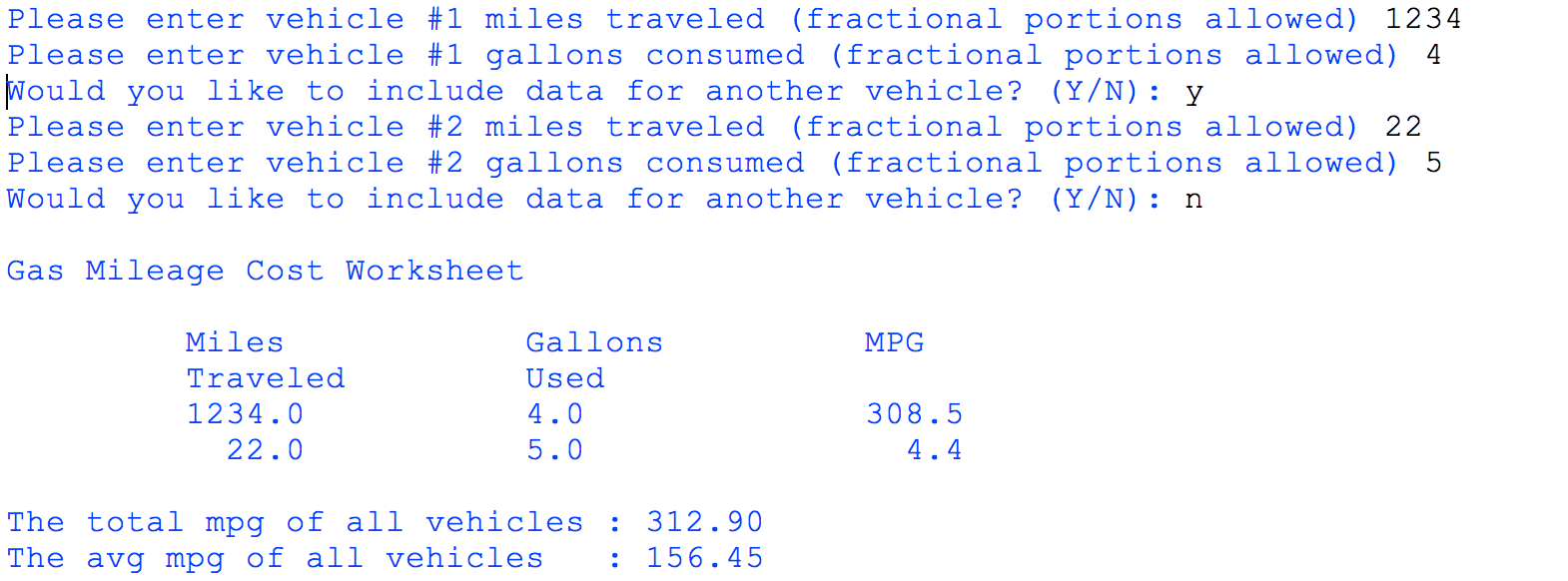
print ("\t", str(i[0]).rjust(6), " ".ljust(6), str(i[1]).rjust(6),

" ".ljust(11), "{:,}".format(i[2]).rjust(6))

#format summary info

print ("\nThe total mpg of all vehicles :", str(totalMPG).ljust(6,'0'))

print ("The avg mpg of all vehicles :", totalMPG/len(applist))



**MPG w. no loop problem**

'''Gas mileage data compilation'''

#housekeeping - declare & initialize vars

totalMPG = 0.0 #make float type

totalMiles = 0.0

totalGallons = 0.0

#handle car1

miles1 = float(input("Please enter vehicle 1 " \

" miles traveled (fractional portions allowed) "))

gallons1 = float(input("Please enter vehicle 1 " \

" gallons consumed (fractional portions allowed) "))

totalMPG += miles1/gallons1 #accumulate mpg

#handle car2

miles2 = float(input("Please enter vehicle 2 " \

" miles traveled (fractional portions allowed) "))

gallons2 = float(input("Please enter vehicle 2 " \

" gallons consumed (fractional portions allowed) "))

totalMPG += miles2/gallons2 #accumulate mpg

#format header for display

print ("\nGas Mileage Cost Worksheet\n")

print ("\t", "Miles".ljust(16), "Gallons".ljust(16), "MPG".ljust(16))

print ("\t", "Traveled".ljust(16), "Used".ljust(16))

#format detail for display

print ("\t", str(miles1).rjust(6), " ".ljust(6), str(gallons1).rjust(6),

" ".ljust(11), "{:,}".format(miles1/gallons1).rjust(6))

#print summary info

print ("\nThe total mpg of all vehicles :", str(totalMPG).ljust(6,'0'))

print ("The avg mpg of all vehicles :", totalMPG/2)