Coding in Python

Online Resource = Codecademy
emoser91@gmail.com
emoser
msoe4moser

Basic syntax -

- writing in file called script.py
- easy to print out
  print "welcome to python!"
- Variables
  my_variable = 10
  my_int = 7
  my_float = 1.23
  my_bool = True (capital "T" important)
- Reassigning
  my_int = 7
  my_int = 3
  print my_int (prints 3)
- white space means right space
  def spam();
  (4spaces) eggs = 12
  return eggs
  print spam() → (prints 12)
- single line comments
  # comment
- multi line comments
  """ comment """
- math
  addition = 72 + 23
  subtraction = 108 - 204
  multiplication = 108 * 0.5
  division = 108 / 9
- math
  exponentiation
  eggs = 10** 2 → 10^2
  modulo (remainder operator)
  spam = 3%2 → remainer of 1
  meal calculation example
  meal = 44.50
  tax = 0.0675
  tip = 0.15
  meal = meal + meal * tax
  total = meal + meal * tip
- strings
  name = "Eric"
  escaping characters
  'There\'s a snake in my boot'
  ↑ prevents errors
  Access by index
  fifth_letter = "Monty" [4]
  ↳ sets varrible to letter "Y"

- <u>String methods</u> strings and console output

length

```
parrot = "Norwegian Blue"
len(parrot)
print len(parrot)
```

↳ prints 14

lowercase

```
print parrot.lower()
```

upper case

```
print parrot.upper()
```

str() function turns non-strings into strings

```
pi = 3.14
print str(pi)
```

- <u>Printing methods</u>

print "Monty Python" → prints string in console

print variable → prints what is stored in variable

print "Spam" + "and" + "Eggs" → prints concatination of strings

print "Value of pi is " + str(3.14)

print " I like %s and %s " (string 1 , string-2)

- <u>String I/O with command console</u>

name = raw_input ("what is your name")

quest = raw_input ("what is your quest")

color = raw_input ("what is your favorite color")

↳ lets you set the value of the variables from console

- Getting current date and time

```
from datetime import datetime
now = datetime.now()
print now
print now.year
print now.month
print now.day
print '%s/%s/%s' % (now.month, now.day, now.year)
print '%s:%s:%s' % (now.hour, now.minute, now.second)
```

- Conditionals and Control flow

example:

```
def clinic():
    print "You've just entered the clinic!"
    print "Do you take the door on the left or right?"
    answer = raw_input ("type left or right and hit 'enter' ").lower()
    if answer == "left" or "l":
        print "        "
    elif answer == "right" or "r":
        print "        "
    else:
        print "        "
clinic()
```

- Comparators

equal to ==

not equal to !=

less than <

less than or equal to <=

greater than >

greater than or equal to >=

- Boolean operators (all lower case)

and → checks if both are true

or →

not →

True and True = True

True or False = False

not True = False

example function

Pig latin translator

```
pyg = 'ay'
original = raw_input ('Enter a word : ')
if len (original ) > 0 and original . isalpha ( ) :
    print original
    word = original . lower()
    first = word [0]
    new_word = original [ 1 : len (original)] + first + pyg
    print new_word
else :
    print 'empty'
```

checks to make sure only letters entered

Functions

```
def square (n):
    """ Returns the square of a number """
    squared = n**2
    print "%d squared is %d" % (n, squared)
# Call function
square (10)  -> will return 10 squared is 100
# you can add more values to function with commas such as def power (base, exponent):
# then call with power (37,4)
```

- functions calling functions

```
def cube (n):
    n = n**3
    return n
def by_three (n):
    if (n % 3 == 0):
        m = cube (n)
        return m
    else:
        return False
```

Imports

```
import math
print math.sqrt (25)
```

-> only import sqrt

```
from math import sqrt
```

-> want all varribles

```
from math import*
```

-> How to see whats in an import

```
import math
everything = dir(math)
print everything
```

Imports

- premade functions

```
def biggest_number (*args):
    print max(args)
    return max(args)
def smallest_number (*args):
    ⋮
def distance_from_zero(args):
# function calls
maximum = max(1,2,3)
minimum = min(1,2,3)
absolute = abs(-1)

print type (42)  → prints  type int
print type (4.2  →         type float
print type ("Hi") →        type str
```

Functions

```
def hotel_cost (nights):
    return 140*nights
def plane_ride_cost (city):
    if city == "Tampa":
        return 220
    ⋮
```

Functions

```
def rental_car_cost (days)
    cost = days * 40
    if days >= 7 :
        cost = cost - 50
        return cost
    elif days >= 3 :
        cost = cost - 20
        return cost
    else :
        cost = cost
        return cost
```

→ Worked on vacation trip expenses program
↳ found on OneNote under python tab
↳ great review of functions

Lists - lists and dictionaries

→ like arrays in C

```
zoo_animals = ["pangolin", "casowary", "sloth", "Tiger"];
if len (zoo_animals) >3 :
    print "The first animal is " + zoo_animal [0]
    print " The second animal is " + zoo_animal [1]
    ⋮
```

Indexing -

```
        0  1  2  3
numbers = [5, 6, 7, 8]
print numbers [0] + numbers [1]
```

Slicing -

```
suit case = ['a', 'b', 'c', 'd']
suitcase [0 : 2]
my_list [:2]   #grabs the first two items
my_list [3:]   # grabs the fourth through last items
```

→ example

```
animals = "cat dog frog"
cat = animals [:3]
dog = animals [3 : 6]
frog = animals [6 :]
```

Maintaining order

```
animals = ["badger", "duck"]
duck_index = animal.index ("duck")
animal.insert (duck_index, "cobra")
```

↳ inserts cobra at duck location and shifts the rest right

↳ doesn't delete duck or remove it

For loops

```
my_list = [1, 9, 3, 8, 5, 7]
for number in my_list:
    print 2* number
```

↳ will print out 2, 18, 6, 16, 10, 14

another example

```
start_list = [5, 3, 1, 2, 4]
square_list = [ ]
for number in start_list:
    square_list.append (number**2)
    square_list.sort()
print square_list
```

Dictionary

similar to a list, but you can access values by looking up a key instead of an index

d = { 'key1' = 1 , 'key2' = 2 , 'key 3' = 3 }

→ great for pairing things such as phone numbers to names

example

residents = { 'A' = 1 , 'B' = 2 , 'C' = 3 }

print residents ['A'] → will print 1

You can also add new entries into an empty dictionary

menue { } #empty

menue ['spam'] = 2.50

menue ['Cookie'] = 1.50

print menue

#Note str (len (menue)) → will print how many items in dictionary

You can delete and change entries

del dictionary_name ['key']

dictionary_name ['key'] = new_value

Lists

You can remove items off of a list

backpack = ['a', 'b', 'c']

backpack . remove ('a')

Final Review of Lists and Dictionarys

key concept

dictionary_name ['list_key'] . list_function ()

→ such as .sort()

OR .remove ('key')

example found in <u>One Note</u>

For loops

example printing out a list with for loop

```
names = ["Eric", "Bob", "Phil"]
for loop in names:
    print loop
```

example printing keys in a dict with a for loop

```
dictionary = {  }
for key in dictionary
    print dictionary[key]
```

example

```
a = [0, 1, 2, 3, 4]
for number in a
    if number % 2 = 0
        print number
```

example

```
def fizz_count (x):
    count = 0
    for n in x:
        if n == "fizz":
            count += 1
    return count
# calling function
fizz_count (["fizz", "cat", "fizz"])
```

Dictionary and for loops

- see OneNote example for owning a store

Dict, list and printing with for loop

you can have a list in a dictionary

example

```
lloyd = {
        "name": "Lloyd",
        "homework": [90.0, 97.0, 75.0, 92.0]
    }
students = ["lloyd", "alice", "tyler"]
for n in students
    print n ["name"]
    print n ["homework"]
```

Complex example using classroom grade calculations on OneNote

Lists and Functions

- list modification

```
n = [1, 3, 5]
n[1] = n[1]*5     # replaces n[1] with n[1]*5
print n
```

- append to list

```
n.append (4)
print n  -> results in [1, 3, 5, 4]
```

- removing elements n = [1,3,5]

```
n.pop(1)          OR    n.remove(1)      OR    del (n[1])
# prints [1, 5]         # prints [3,5]         # prints [1,5]
```

Lists and functions

- m=5

```
n= 13
def add_function (x,y):
    result = x+y
    return result
print add_function (m,n)
```

- n= "Hello"

```
def string_function (s):
    arg = s+ 'world'
    return arg
print string_function (n)
```

- passing a list to a function

```
def list_function (x)
    return x
n= [3,5,7]
print list_function (n)
```

- modify a list element in a function

```
def list_function (x):
    x[1] = x[1]+3
    return x
```

- append to a list with function

```
def list_extender (LST)
    Lst.append (9)
    return list_extender
print list_extender (n)
```

Ranges

range (6) → [0,1,2,3,4,5] ⇒ range (stop)

range (1,6) → [1,2,3,4,5] ⇒ range (start, stop)

range (1,6,3 → [1,4] → range (start, stop, step)

- printing out a list

```
def print_list (x):
    for i in range (0, len(x)):
        print x[i]
print_list (n)
```

- double list

```
def double_list (x):
    for i in range (0, len(x)):
        x[i] = x[i]*2
    return x
```

Lists and Functions

- adding strings

```
n = ["michael", "Lieberman"]
def join_strings (words):
    result = " "
    for i in range (0, len(words)):
        result = result + words [i]
    return result
print join_strings (n)
```

- appending lists together with a function

```
n = [[1, 2, 3], [4,5,6,7,8,9]]
def flatten (lists):
    result = [ ]
    for numbers in lists :
        for numbers in numbers
            result.append (numbers)
    return result
print flatten (n)
  # prints [1,2,3,4,5,6,7,8,9]
```

- Battleship Game in <u>One Note</u>

<u>while loops</u> - loops

```
count = 0
while count <10:
    print "count is", count
    count +=1
```

while loops

simple error checker

```
choice = raw_input ('Enjoying the course? (Y/N)')
while choice != 'y' and choice != 'n':
    choice = raw_input ("Sorry, I didn't catch that. Enter again = ")
```

while / else loops

```
random_number = randint (1,10)
guesses_left = 3
while guesses_left >0 :
    guess = int (raw_input ("Your Guess:"))
    if guess == random_number:
        print "You win!"
        break
    else:
        print "wrong number"
        guesses_left -= 1
else:
    print "Sorry you lose"
```

For loops

```
hobbies = [ ] #empty list
for i in range (3):
    hobby = (raw_input ("Your hobby: ")) # raw input of strings
    hobbies.append (hobby)
    i+= 1
```

Printing out strings

```
word = "Eggs!"
for n in word:
    print n    # prints each letter out
```

for loops, changing strings and extended line print

```
phrase = "A bird in the hand..."
for char in phrase:
    if char == 'A' or char == 'a':
        print 'x',
    else
        print char
```

Looping over a dictionary

```
d = {'a' = 'apple', 'b' = 'berry', 'c' = 'cherry'}
for key in d:
    print key,
    print d [key]
  # print key followed by item on each line
  # looks like  a  apple
                b  berry
                c  cherry
```

For / else statements

```
for f in fruits:
    if f == 'tomato':
        ⋮

else:
    ⋮
```

Practice Problems

Function that checks if a value is even

```
def is_even (x):
    if x%2 == 0:
        return True
    else:
        return False
```

Function that adds all integers together so 1234 = 10

```
def digit_sum (n):
    n = str(n)
    numbers = [ ]
    for x in n:
        numbers.append (int(x))
    return sum (numbers)
print digit-sum (1234)
```

Factorial Function

```
def factorial (x):
    for i in range (1, x)
        x = i * x
    return x
# Factorial (4) will return 4*3*2*1 = 24
```

Prime number function

```
def is_prime (n):
    if n <= 1:
        return False
    else
        for m in range ( 2, int(n** 0.5) + 1 ):
            if not n%m:
                return False
            return True
```

Reverse Function

```
def reverse (test):
    n = len(test)
    x = " "
    for i in range (n-1, -1, -1):
        x += test [i]
    return x
print reverse ("eric")
 # prints out  cire
```

Anti Vowel Function

```
def anti_vowel (text):
    result = text
    for char in text:
        if char in "aeiouAEIOU":
            result = result. replace (char, " ")
    return result
```

Scrabble Score function

```
# score dictionary with letters and scores given
def scrabble_score (word):
    point = []
    for c in word.lower():
        pts = score [c]
        point. append (pts)
        return sum(point)
```

Censor Function

```
def censor (text, word):
    return text. replace (word, ("**" len (word)))
print censor ("this is cool", "cool")
 # prints out  this is ****
```

Filter out odd numbers function

```
def purify (numbers)
    newNumber = []
    for i in numbers:
        if i%2 == 0:
            newNumber.append (i)
    return newNumber
print purify ([4,5, 5, 4])
#prints out [4, 4]
```

Function that multiplys list items together

```
def product (integers)
    productNumber = 1
    for i in integers:
        productNumber *= i
    return productNumber
print product ([4, 5, 5])
#prints out 100
```

Function that removes duplicates from a list

```
def remove_duplicates (elements):
    newList = []
    for i in elements:
        if i not in newList:
            newList.append (i)
    return newList
print remove_duplicates ([1, 1, 2, 2])
#returns [1, 2]
```

median Function

```
def median (lst):
    lst = sorted (lst) # sorts smallest to largest
    if len(lst) < 1 : # case if there is no entry / empty list
        return None
    if len (lst)%2 == 1 : # calculation for even length lists
        return lst [((len (lst)+1)/2)-1]
    else:
        return float (sum (lst [((len (lst)/2)-1 : (len (lst)/2)+1])) / 2.0
```

→ Completed Exam Statistics problem in OneNote

Advanced Python Topics

Printing a Dictionary

```
my_dist = {"name", "Eric",
           "age", 23,
           "bdfl", True
          }

print my_dist.items  #prints [('name', eric), ('age', 23), ('bdfl', True)
print my_dist.keys()  #prints [name, age, bdfl]
print my_dist.values()  #prints [eric, 23, True]
```

The "in" operator

```
for number in range (5):
    print numbers,
for key in my_dict [key]:
    print key, my_dict [key],
for letter in "eric":
    print letter
```

List Comprehension syntax

```
doubles_by_3 = [x*2 for x in range (1,6) if (x*2)%3 == 0]
even_squares = [i**2 for i in range (1,11) if (i**2)%2 == 0]
print even_squares
# prints 4, 16, 36, 64, 100
```

List slicing syntax

```
L = [i**2 for i in range (1,11)]
# should be [1, 2, 9, 16, 25, 36, 49, 64, 81, 100]
print L [2:9:2]      # prints 9, 25, 49, 81
# syntax [start : end : step]
```

Further syntax

```
to_five = ['A', 'B', 'C', 'D', 'E']
print to_five [3:]  # prints D, E
print to_five [:2]  # prints A, B
print to_five [::2] # prints A, C, E
```

Reverse a List

```
my_list = range (1,11)
backwards = my_list [::-1]
print backwards
# prints 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

Anonymous Functions

```
# you can pass functions around just as if they were varribles
lambda x: x%3 == 0
is the same as
def by_three (x):
    return x%3 == 0
```

Lambda syntax

```
languages = ["HTML", "JavaScript", "Python", "Ruby"]
print filter (lambda x: x == "python", languages)
#prints ['python']
```

List Comprehension Syntax

```
# list of numbers from 1-15 that are divisible by 3 or 5
three_and_five = [x for x in range (1,16) if x%3 == 0 or x%5 == 0]
print three_and_five
```

Bitwise Operators

```
print 5>>4   # Right Shift = 0
print 5<<1   # left shift = 10
print 8 & 5  # Bitwise and = 0
print 9 | 4  # Bitwise or = 13
print 12^42  # Bitwise Xor = 38
print ~88    # Bitwise not = -89
```

Binary representation in Python

```
0b1 = 1
0b10 = 2
0b11 = 3
0b100 = 4
# same as in C language
```

Base 10 to binary Conversion (int to binary)

```
print bin(1)  # 0b1
print bin(2)  # 0b10
```

Binary to int

```
print int("111", 2)  # = 7
print int("0b100, 2)  # = 4
print int( bin(5), 2)  # = 5
```

Shifting Binaries left and right

shift_right = 0b1100

shift_left = 0b1

shift_right = 0b1100 >>2

shift_left = 0b << 2

print bin(shif_right) # 0b11

print bin(shift_left) # 0b100

Bitwise and = &

0 & 0 = 0

1 & 0 = 0

0 & 1 = 0

1 & 1 = 1

Bitwise or = |

0 | 0 = 0

0 | 1 = 1

1 | 0 = 1

1 | 1 = 1

Bitwise Xor = ^

0 ^ 0 = 0

0 ^ 1 = 1

1 ^ 0 = 1

1 ^ 1 = 0

Bit Masking

```
num = 0b1100
mask = 0b0100
desired = num & mask
if desired >0 :
    print "Bit on"
# checking 4th place from right
def check_bit4 (num):
    mask = 1 << 3
    return "on" if (num & mask ) > 0 else "off"
print check_bit4 (0b1000)

# turn on third bit from the right
a = 0b1011011
a |= (1 << 2)
print bin(a)      # = 0b1011111

#using Xor to flip all bits
a = 0b11101110
mask = 0b11111111
desired = a^mask
print bin(desired)
```

Classes - Introduction to classes

```
Class Animal (object):
    """Makes cool animals"""
    is_alive = True
    def__init__(self, name, age):
        self.name = name
        self.age = age
zebra = Animal ("Jeffrey", 2)
giraffe = Animal ("Bruce", 1)
panda = Animal ("Chad", 7)

print zebra.name, zebra.age, zebra.is_alive
#prints Jeffrey 2 True
```

↳ add more class information

File I/O

```
# have script.py file and output.txt file
my_list = [i**2 for i in range (1,11)]
f = open ("output.txt", "w")
for item in my_list:
    f.write (str (item) + "\n")
f.close
#print output = 1,4,9,16,25,36,49,64,81,100
```

The open function

```
f = open ("output.txt", "w")
```

→ told python to open output.txt in "w" mode ("write mode")

note r = read only mode

r+ = read/write mode

```
my_file = open ("output.txt", "r+")
# allows you to read and write to it
```

Writing data into .txt file

syntax: my_file.write ("Data to be written")

```
my_list = [i**2 for i in range (1,11)]
my_file = open ("output.txt", "r+")
for x in my_list:
    my_file.write ("%s\n" % str (x))
my_file.close() #needs to close to work properly
```

Reading data from output.txt

syntax: print my_file.read()

```
my_file = open ("output.txt", "r")
print my_file.read() #prints file contents into console window
my_file.close()
```

Check if files are closed

```
f = open ("bg.txt")
f.closed #False
f.close()
f.closed #True
```

File I/o

Reading a file line by line

```
my_file = open ("text.txt", "r")
print my_file.readline()
print my_file.readline()
print my_file.readline()
# prints out lines in order
```

Read/write process

```
# open the file for reading
read_file = open("text.txt", "r")
# second file handler to open the file for writing
write_file = open("text.txt", "w")
#write to the file
write_file.close()
#reading from the file
print read_file.read()
read_file.close()
```

With and as key words

```
with open("text.txt", "w") as textfile:
    textfile.write ("sucess!")
# this method automatically closes the file
```

Test if files are closed or not

```
with open("text.txt", "w") as textfile:
    textfile.write ("sucess!")
if not myfile.close():
    myfile.close()
    print myfile.closed
```

Classes (continued)

example

```
Class fruit (object):
    """ A class that makes various tasty fruits """ ← multi-line comment
    def __init__ (self, name, color, flavor, posionous):
        self.name = name
        self.color = color
        self.flavor = flavor
        self.posionous = posionous
    def description (self):
        print "I'm a %s %s and I taste like %s" % (self.color, self.name, self.flavor)
    def is_edible (self):
        if not self.posionous:
            print "Yep, im edible"
        else:
            print "Do not eat me"

lemon = fruit ("lemon", "Yellow", "Sour", False) ← adding object to class
lemon.description () ← function call
lemon.is_edible () ← function call
```

Class name

Try and Except in Python

Error handling in python is done through the use of exceptions that are caught in try blocks and handled in exception blocks.

try and exception - If an error is encountered, a try code execution is stopped and transfered down to the exception block

In addition to using an exception block after the try block, you can also use the finally block

The code in the finally block will be executed regardless of whether an exception occurs

Raising an exception - you can raise an exception in your own program by using the raise exception [, value] statement.

Example -

```
try:
    print "Hello World"
except:
    print "This is an error message!"
```

Common exception errors:

IO Error - If the file can not be opened

Imput Error - If python can not find the module

Value Error - Raised when a built-in operation or function recieves an argument that has the right type but an inappropriate value

Keyboard Interrupt - Raised when the user hits the interrupt key (Normally Control - C or Delete)

EOF Error - Raised when one of the built-in functions (input() or raw_input()) hits an end-of-file condition without reading any data

Try and Exception in python

Example-

```
try:
    block-1 ...
except Exception 1 :
    handler-1 ...
except Exception 2 :
    handler-2 ...
else:
    else-block
finally :
    final-block
```

Explanation of Example:

The code in block-1 is executed. If the code raises an exception, the various exception blocks are tested: if the exception is of class Exception-1, handler-1 is executed; Otherwise if its of class Exception-2, handler-2 is executed and so forth. If no exception is raised, the else-block is executed. No matter what happened previously, the final-block is executed once the code block is complete and any raised exceptions are handled

↳ final-block will always be run

Use of raise -

```
print " Test Case A"
testFunctionCall ("TestValue") = TestFinalValue
if TestFinalValue is '':
    raise RuntimeError ('No data returned from testCallFunction')
```

Case/switch statement does not exist in python!

learn about enter = lambda :
exit = lambda :
and lambda in general

```
def drive_start_stop_context(self, jog=False):
    return CustomContext(
        enter=lambda: self.command_word(('JOG' if jog else 'START',)).write(),
        exit=lambda: self.command_word(() if jog else ('STOP',)).write())
```