Lecture Content: Wednesday (3 hours -- Morning)

Python background

1) Purpose: web development (server side); software development, statistical analysis

2) Why learn python --

Python has been popularly used like R. Julia is an emerging language.

a) Run on different platforms (Windows, Mac, Linux etc).

b) Easy to learn

c) Many developed libraries -- do not have to learn from scratch

d) Python runs on an interpreter system, doesn't need to compile --> write code quickly

3) Python versions: Python version 2 and 3

Each command will end if there is not more text after the end on the same line.

Python relies on indentation of four spaces or one tab to define the scope of loop, function, class, etc

4) Python code editor • simple text editor • IDE (Integrated Development Environment) is an editor with many more functions like running code, syntax highlighting, automatic code formatting, debugging

• download the anaconda package, which contains Spyder and Jupyter notebook.

Spyder -- free IDE

Jupyter notebook -- can contain code and results on the same page.

Anaconda setup --

How to use Jupyter notebook

-- basic Jupyter notebook operation can be found at
http://nbviewer.jupyter.org/github/jupyter/notebook/blob/master/docs/source/examples/Notebook/No
(http://nbviewer.jupyter.org/github/jupyter/notebook/blob/master/docs/source/examples/Notebook/No

-- Click Jupyter notebook icon on the Anaconda Menu

-- Click "new" on the right corner and select "Python" from the pull-down menu.

-- Name the notebook page: Click "Untitled".
In the "Rename Notebook" dialog box, type a proper name like "Python basics". Click "Rename".

-- cell actions apply to the currently selected cell

OR at the command prompt, type "jupyter notebook"

execute commands

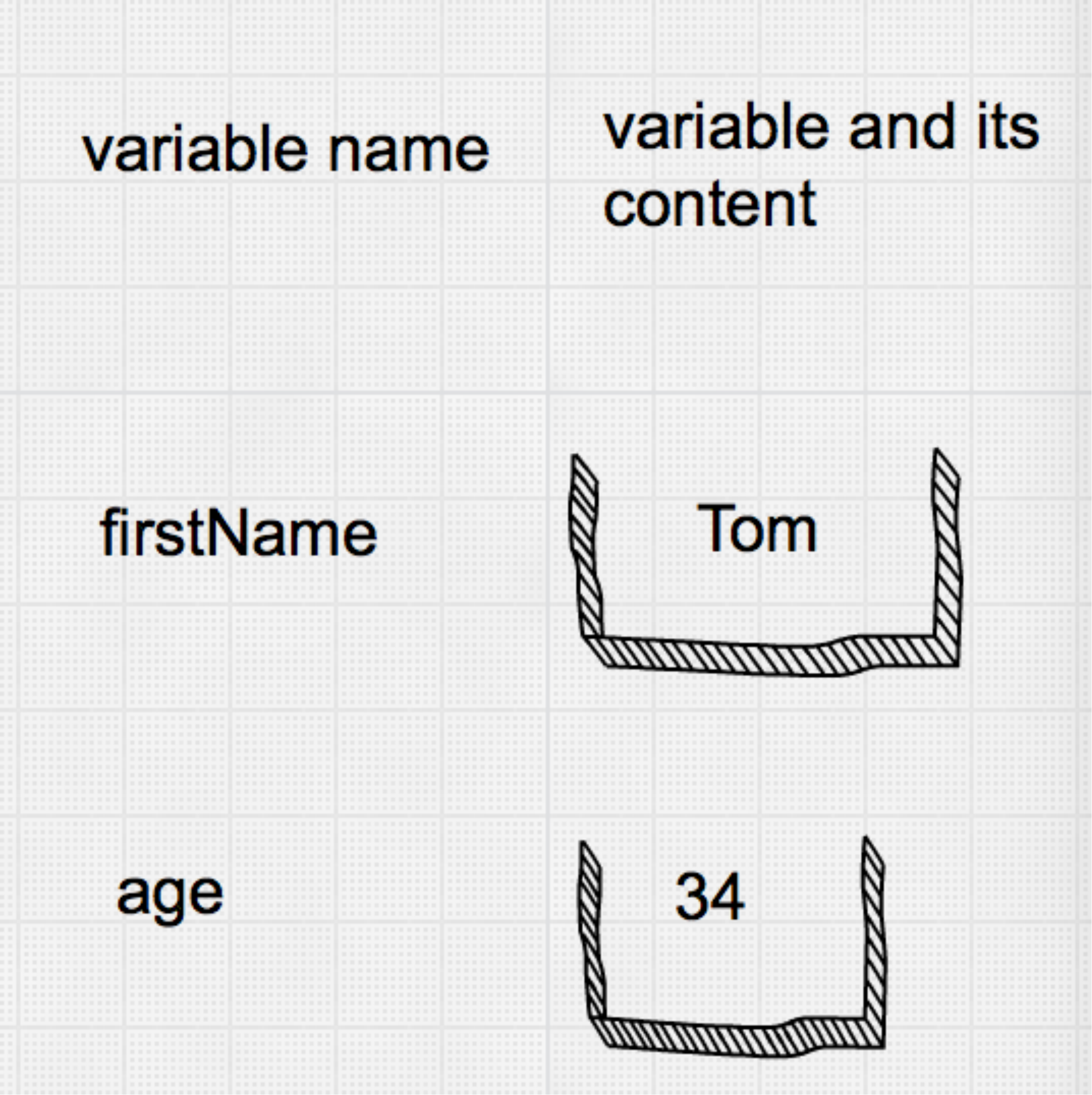
-- type print("Hello World!") and click the "Run" button

-- type print("I feel glad to take this workshop!") -- hold down the shift key and press the return key).

-- print("여권 대신 얼굴") and run it

-- how to add a cell below hold down alt/option key and press the Return key

variables and their names



print("Hello World!")

In [3]:

```
print("I am glad to take this workshop")
```

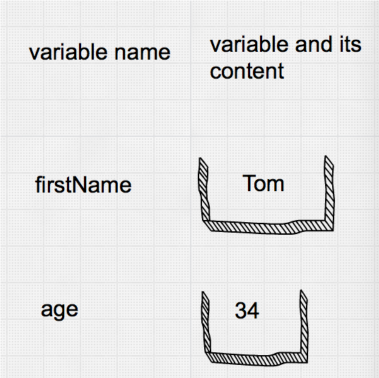
```
I am glad to take this workshop
```

In [3]:

```
print("여권 대신 얼굴")
```

```
여권 대신 얼굴
```

variables and their names



Rules for variable names

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

data types:

basic types: numbers, strings, boolean

complex types: list, tuple, dictionary, dates and time

In [21]:

```
#### Python numbers: int, float, complex
x = 2
y = 1.6
print(x)
print(type(x))

print(type(y))
```

```
2
<class 'int'>
<class 'float'>
```

In [190]:

```
#### data casting (specify the data type)
x = int(5)   # x will be 5
y = int(6.8) # y will be 6
z = int("4") # z will be a character "4"

print(x, y, z)
```

```
5 6 4
```

string literal is a sequence of characters

It is enclosed in single or double quotation marks. Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

In [20]:

```
sentence = "Hello World!"
print(sentence)
print(sentence[3])
print(sentence[4:7])

#slicing
```

```
Hello World!
l
o W
```

Complex data types like list, tuple, dictionary and date and time

In [25]:

```
##### list
items = ["one", "two", "three"] # the result of a list
print(items)
items[1] = "five"
print(items)
print(items[1])

```

```
['one', 'two', 'three']
['one', 'five', 'three']
five
```

In [48]:

```
# list constructor  -- create a list
animals = list(("tiger", "lion", "bird")) # create a list

# add items
animals.append("wolf") # add this to the end of a list

animals.insert(3, "ox") # Adds an element at a specified position

# delete items

animals.pop()
animals.pop(3) # delete one at a specfic position

animals.remove("bird") #

del animals[0] # delete an item from a specific position

```

In [53]:

```
animals.append("wolf")
```

In [61]:

```
print(animals)
```

```
['lion', 'wolf', 'wolf', 'one', 'two', 'three']
```

In [ ]:

```

```

In [55]:

```
# count()   how many elements with the specified value
animals.count("ox")
len(animals)  # the number of itmes in the list
```

Out[55]:

```
4
```

extend() Add the elements of a list (or any iterable), to the end of the current list ex) animals.extend(["one", "two", "three")

In [62]:

```
animals.extend(["one", "two", "three"])
```

In [64]:

```
animals.index("lion") # 1
```

Out[64]:

```
0
```

Accessing Values in Lists

https://www.tutorialspoint.com/python/python_lists.htm (https://www.tutorialspoint.com/python/python_lists.htm)

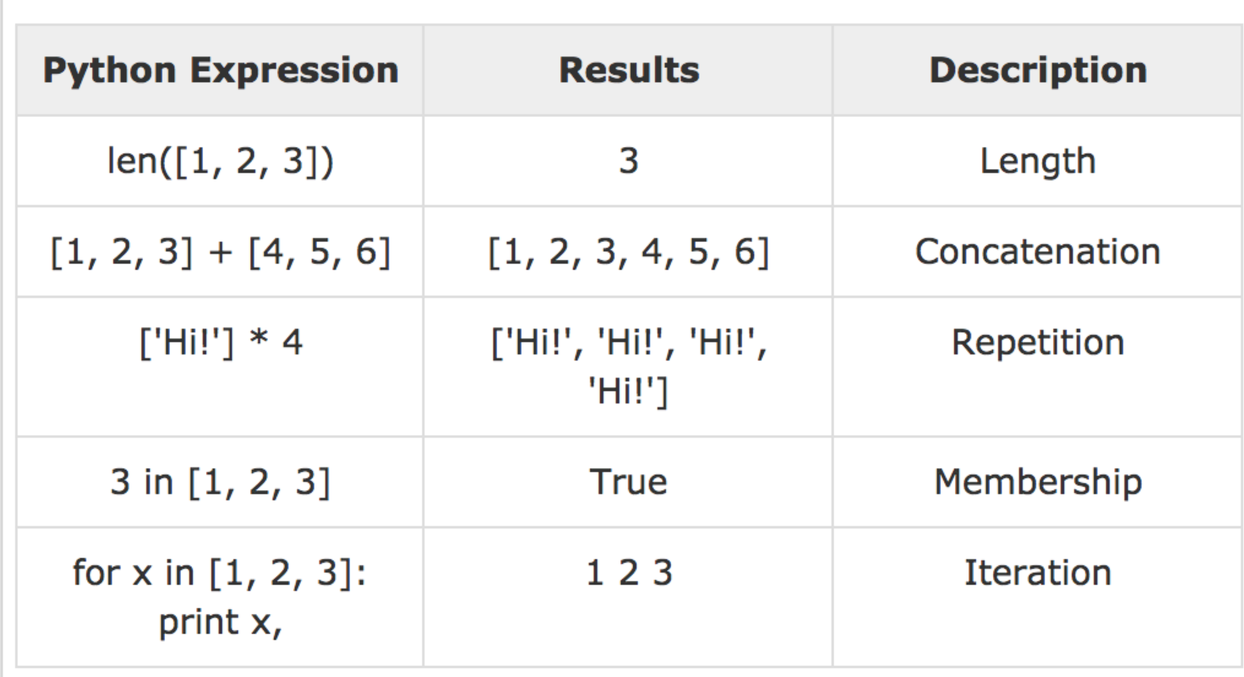
In [65]:

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];
print( "list1[0]: ", list1[0])
print ( "list2[1:5]: ", list2[1:5])


```

```
list1[0]:  physics
list2[1:5]:  [2, 3, 4, 5]
```

Basic List Operations

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.



Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

List comprehensions:

When programming, frequently we want to transform one type of data into another. As a simple example, consider the following code that computes square numbers:

In [69]:

```
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)   # Prints [0, 1, 4, 9, 16]
#You can make this code simpler using a list comprehension:

squares2  = [x**2 for x in nums]
print(squares2)
```

```
[0, 1, 4, 9, 16]
[0, 1, 4, 9, 16]
```

In [72]:

```
### List comprehensions can also contain conditions:
```

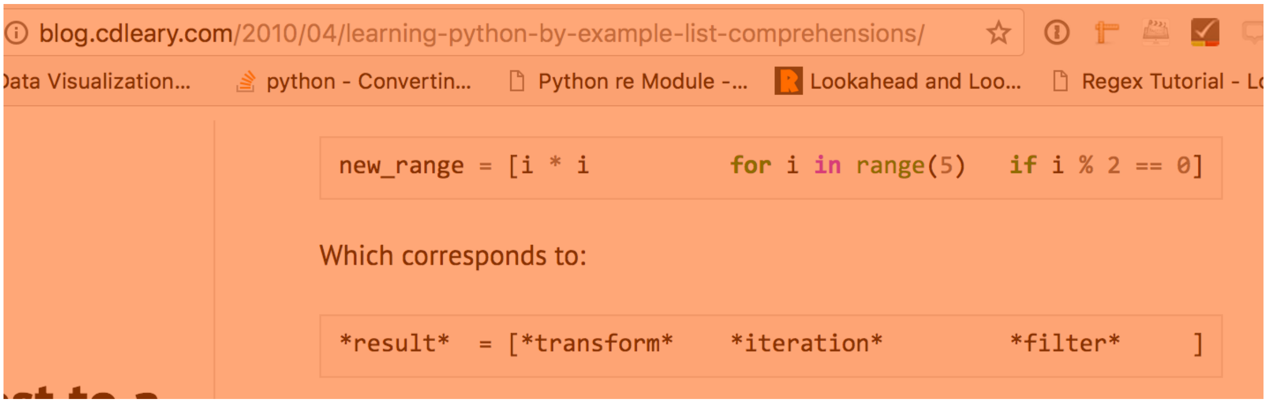
In [82]:

```
nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)  # Prints "[0, 4, 16]"

even_squares2 = []

for x in nums:
    if x%2 == 0:
        even_squares2.append(x**2)

print(even_squares2)
```

```
[0, 4, 16]
[0, 4, 16]
```

comprehension • 컴프리헨션

comprehension (http://www.pythonforbeginners.com/basics/list-comprehensions-in-python (http://www.pythonforbeginners.com/basics/list-comprehensions-in-python))



```
new_range = [i * i          for i in range(5)   if i % 2 == 0]
```

Which corresponds to:

```
*result*  = [*transform*    *iteration*         *filter*     ]
```

ORDER: 1) iteration; 2) filter; 3) transform 2) filter is optional

new_range = [] for i in range(5): if i%2 == 0: new_range.append(i*i)

Tuple

The tuple() Constructor It is also possible to use the tuple() constructor to make a tuple. The len() function returns the length of the tuple.

Example Using the tuple() method to make a tuple:

counting= tuple(("one", "two", "three"))
print(counting)

In [192]:

```
#
```

In [80]:

```
counting= tuple(("one", "two", "three"))
print(counting)

```

```
('one', 'two', 'three')
```

set

fruits = set(("apple", "banana", "cherry"))

fruits.remove("banana")

print(fruits)

print(len(fruits))

In [193]:

```
fruits = set(("apple", "banana", "cherry"))

fruits.remove("banana")

print(fruits)

print(len(fruits))
```

```
{'apple', 'cherry'}
2
```

In [194]:

```
### slicing
#Slicing: In addition to accessing list elements one at a time, Python provides concise syntax to access sublists; this is known as slicing:

nums = list(range(5))     # range is a built-in function that creates a list of integers
print(nums)               # Prints "[0, 1, 2, 3, 4]"
print(nums[2:4])          # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print(nums[2:])           # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print(nums[:2])           # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print(nums[:])            # Get a slice of the whole list; prints "[0, 1, 2, 3, 4]"
print(nums[:-1])          # Slice indices can be negative; prints "[0, 1, 2, 3]"
nums[2:4] = [8, 9]        # Assign a new sublist to a slice
print(nums)               # Prints "[0, 1, 8, 9, 4]"

```

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[0, 1, 8, 9, 4]
```

dictionary

myDic = { "one": 1, "two": 2, "three": 3 }

- add one item

myDic["four"] = 4

- add multiple items

myDic.update({"five": 5, "six": 6})

operators

"+" "-" "!= " "=" (assignment) "==" (equal sign)

conditions

Equals: a == b

Not Equals: a != b

Less than: a < b

Less than or equal to: a <= b

Greater than: a > b

Greater than or equal to: a >= b

In [88]:

```
# conditional statements
# if
#if else
#if elif else

lee = 41
tom = 53
if tom > lee:
    print("correct")

if tom > lee:
    print("correct")
else:
    print("incorrect")

price = 5

if price == 2:
    print("wrong answer")
elif price == 3:   #(one or multiple elif)
    pass
elif price == 4:
    pass
else:
    print("correct answer")
```

```
correct
correct
correct answer
```

In [196]:

```
#### while and break -- when a condition is met, break will stop the loop
i = 1
while i < 6:
  print(i)
  if i > 4:
      break
  i += 1
```

```
1
2
3
4
5
```

In [197]:

```
### continue -- skip once when an if conidtion is met and continue the loop
i = 1
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)

```

```
2
4
5
6
```

In [93]:

```
### for
fruits = ["pear", "banana", "persimmon"]
for x in fruits:
  print(x)

### for and break
  fruits = ["pear", "banana", "persimmon"]
for x in fruits:
  if x == "banana":
    break
  print(x)

```

```
pear
banana
persimmon
pear
```

In [202]:

```
### pass, range


for number in range(10,20):
    if number < 15:
        pass
    else:
        print(number)

```

```
15
16
17
18
19
```

In [203]:

```
for number in range(10,20, 2):
    if number < 15:
        pass
    else:
        print(number)
```

```
16
18
```

In [97]:

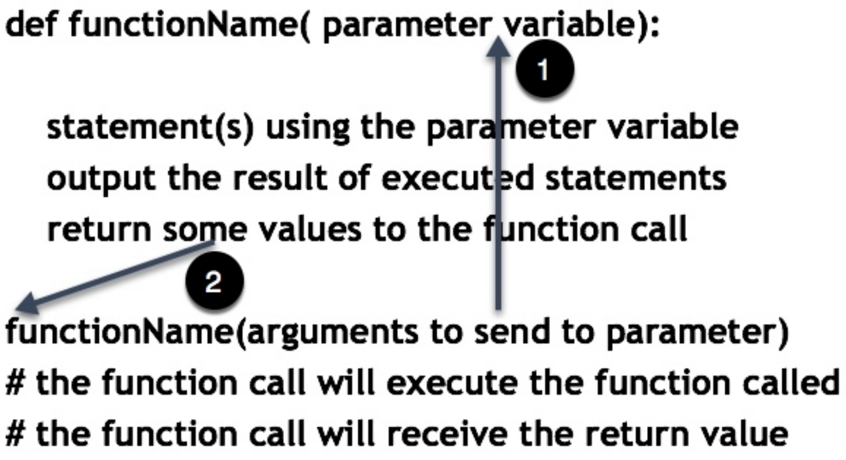
```

for number in range(20):
    if number < 15:
        pass
    else:
        print(number)

```

```
15
16
17
18
19
```

python functions

https://www.w3schools.com/python/python_functions.asp
(https://www.w3schools.com/python/python_functions.asp)

Use the keyword of "def". Need to consider input and output.



```
def functionName( parameter variable):

    statement(s) using the parameter variable
    output the result of executed statements
    return some values to the function call

functionName(arguments to send to parameter)
# the function call will execute the function called
# the function call will receive the return value
```

In [99]:

```python
def namePrint(myName):
    print("My name is " + myName)
    return "Your name is " + myName

namePrint("Judy")

recName = namePrint("Tom")

print(recName)

#-- provide a default value, which will be used if no argument is offered by the function call.

def namePrint(myName = "Lee"):
    print("My name is " + myName)
    return "Your name is " + myName
```

```
My name is Judy
My name is Tom
Your name is Tom
```

In [204]:

```python
### class

#https://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial/Classes
#class is a template or frame.
#You need to create objects out of each class.

class Shape:

    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.description = "This shape has not been described yet"
        self.author = "Nobody has claimed to make this shape yet"

    def area(self):
        return self.x * self.y

    def perimeter(self):
        return 2 * self.x + 2 * self.y

    def describe(self, text):
        self.description = text

    def authorName(self, text):
        self.author = text

    def scaleSize(self, scale):
        self.x = self.x * scale
        self.y = self.y * scale

#-- create an object

# create an object

rectangle = Shape(100, 45)
rectangle.area()
rectangle.describe("I would like to describe this figure")
print(rectangle.description)
rectangle.authorName("Lee")
print(rectangle.author)
rectangle.scaleSize(5)
print(rectangle.x)
print(rectangle.y)
```

```
I would like to describe this figure
Lee
500
225
```

module

functions and classes are saved as an independent file and can be imported as a module

In [205]:

```
### file/directory management

## find the current working directory -- where we are working right now
import os
os.getcwd()  # --> 'C:\\Users\\byunglee'
os.chdir("/tmp/") # change directory  -- go down
os.chdir("..") # Go up one directory from working directory

os.getcwd()
```

Out[205]: '/private'

In [119]:

```
# display all the contents of a directory
#os.listdir('C:\\Users\\byunglee') # pc# os.listdir()
os.listdir('/Users/byunglee/Documents/')  # mac
os.listdir()
```

```
 'AnacondaProjects',
 'Applications',
 'Applications (Parallels)',
 'AT.postflight.58179',
 'AT.postflight.61674',
 'awe.txt',
 'blog',
 'bokeh_plot.html',
 'Boostnote',
 'Calibre Library',
 'CCS.csv',
 'Ch7 TextClassification.ipynb',
 'check495.html',
 'cherrytree-master',
 'CmapToolsLogs',
 'collabshot_screens',
 'ColwizFiles',
 'com495_lr.html',
 'Conduit',
 'critter.py',
```

In [120]:

```
# make sure that you have a folder, workshop, and files under it
#os.listdir('C:\\Users\\byunglee\\Documents\\workshop')
# ['support.py', 'support2.py', '__pycache__']

### read a file
import os
os.getcwd()

```

Out[120]: '/Users/byunglee'

In [139]:

```
### upload read.txt to a folder under the current directory
pathTotestFile = "Documents/workshop/read.txt"
#### open -- bring the file to the memory
fileObj = open(pathTotestFile,"r")   ###” vs " (curly smart quotation marks not working)
readCon = fileObj.read()
print(readCon)

fileObj.close()


```

```
WASHINGTON — President Trump and two members of his cabinet mounted an ag
gressive defense on Monday of his policy of separating children from thei
r parents at the border in response to a growing outcry from members of b
oth parties.

“They could be murderers and thieves and so much else,” Mr. Trump said of
the people crossing the border. “We want a safe country, and it starts wi
th the borders, and that’s the way it is.”

Attorney General Jeff Sessions also defended the practice, while insistin
g that “we do not want to separate parents from their children,” and late
r, at a tumultuous White House news briefing, Kirstjen Nielsen, the secre
tary of homeland security, gave a forceful explanation of the administrat
ion’s actions, arguing that it had no choice, and insisting that the only
way the practice could end would be through congressional action.
```

In [ ]:

```

```

In [ ]:

```

```

In [206]:

```
pathTotestFile = "Documents/workshop/read.txt"
with open(pathTotestFile, 'r') as fileObj:
    print(fileObj.read())

## fileObj.read() could have been replaced as shown below
fileObj.read() # the entire file
fileObj.read(5) # five characters
fileObj.readline() # single line
fileObj.readline(3) # 3rd line

fileObj.readlines() # all lines are displayed as individual items in a list


fileObj = open(pathToFile, "r")
for line in fileObj:
    print(line)

#Python, to read a file, needs to connect to its folder to sys.path
#since Python only reads all files when their folders are linked to sys.path or
#the current working directory.


```

```
---------------------------------------------------------------------------
--
FileNotFoundError                         Traceback (most recent call las
t)
<ipython-input-206-9bce99e42322> in <module>()
      1 pathTotestFile = "Documents/workshop/read.txt"
----> 2 with open(pathTotestFile, 'r') as fileObj:
      3     print(fileObj.read())
      4
      5 ## fileObj.read() could have been replaced as shown below

FileNotFoundError: [Errno 2] No such file or directory: 'Documents/worksh
op/read.txt'
```

In [143]:

```python
fileObj = open(pathTotestFile,"r")
readCon = fileObj.read()
print(readCon)
fileObj.close()

with open(pathTotestFile, 'r') as fileObj:
    print(fileObj.read())

#fileObj.read() # the entire file
#fileObj.read(5) # five characters
#fileObj.readline() # single line
#fileObj.readline(3) # 3rd line

#fileObj.readlines() # all lines are displayed as individual items in a list


fileObj = open(pathTotestFile, "r")
for line in fileObj:
    print(line)


```

```
WASHINGTON — President Trump and two members of his cabinet mounted an ag
gressive defense on Monday of his policy of separating children from thei
r parents at the border in response to a growing outcry from members of b
oth parties.

“They could be murderers and thieves and so much else,” Mr. Trump said of
the people crossing the border. “We want a safe country, and it starts wi
th the borders, and that’s the way it is.”

Attorney General Jeff Sessions also defended the practice, while insistin
g that “we do not want to separate parents from their children,” and late
r, at a tumultuous White House news briefing, Kirstjen Nielsen, the secre
tary of homeland security, gave a forceful explanation of the administrat
ion’s actions, arguing that it had no choice, and insisting that the only
way the practice could end would be through congressional action.
WASHINGTON — President Trump and two members of his cabinet mounted an ag
gressive defense on Monday of his policy of separating children from thei
r parents at the border in response to a growing outcry from members of b
oth parties.

“They could be murderers and thieves and so much else,” Mr. Trump said of
the people crossing the border. “We want a safe country, and it starts wi
th the borders, and that’s the way it is.”

Attorney General Jeff Sessions also defended the practice, while insistin
g that “we do not want to separate parents from their children,” and late
r, at a tumultuous White House news briefing, Kirstjen Nielsen, the secre
tary of homeland security, gave a forceful explanation of the administrat
ion’s actions, arguing that it had no choice, and insisting that the only
way the practice could end would be through congressional action.
WASHINGTON — President Trump and two members of his cabinet mounted an ag
gressive defense on Monday of his policy of separating children from thei
r parents at the border in response to a growing outcry from members of b
oth parties.
```

```
“They could be murderers and thieves and so much else,” Mr. Trump said of
the people crossing the border. “We want a safe country, and it starts wi
th the borders, and that’s the way it is.”



Attorney General Jeff Sessions also defended the practice, while insistin
g that “we do not want to separate parents from their children,” and late
r, at a tumultuous White House news briefing, Kirstjen Nielsen, the secre
tary of homeland security, gave a forceful explanation of the administrat
ion’s actions, arguing that it had no choice, and insisting that the only
way the practice could end would be through congressional action.
```

In [207]:

```python
###read a file as a python, need to connect to its folder to sys.path
###since Python only reads all files when their folders are linked to sys.path or
###the current working directory.
# put a directory liked to sys.path
import sys
print(sys.path)
sys.path.insert(0,  "c:\\Users\\byunglee\\Documents\\workshop")
```

```
['/Users/byunglee/Documents/workshop', '</path/to/application/app/folder
>', '/path/to/application/app/folder', 'c:\\Users\\byunglee\\Documents\\w
orkshop', '', '/anaconda3/lib/python36.zip', '/anaconda3/lib/python3.6',
'/anaconda3/lib/python3.6/lib-dynload', '/anaconda3/lib/python3.6/site-pa
ckages', '/anaconda3/lib/python3.6/site-packages/aeosa', '/anaconda3/lib/
python3.6/site-packages/factor_analyzer-0.2.2-py3.6.egg', '/anaconda3/li
b/python3.6/site-packages/IPython/extensions', '/Users/byunglee/.ipytho
n']
```

In [209]:

```python
import sys
print(sys.path)
sys.path.insert(0,  "/Users/byunglee/Documents/workshop")
```

```
['c:\\Users\\byunglee\\Documents\\workshop', 'c:\\Users\\byunglee\\Docume
nts\\workshop', '/Users/byunglee/Documents/workshop', '</path/to/applicat
ion/app/folder>', '/path/to/application/app/folder', 'c:\\Users\\byunglee
\\Documents\\workshop', '', '/anaconda3/lib/python36.zip', '/anaconda3/li
b/python3.6', '/anaconda3/lib/python3.6/lib-dynload', '/anaconda3/lib/pyt
hon3.6/site-packages', '/anaconda3/lib/python3.6/site-packages/aeosa', '/
anaconda3/lib/python3.6/site-packages/factor_analyzer-0.2.2-py3.6.egg',
'/anaconda3/lib/python3.6/site-packages/IPython/extensions', '/Users/byun
glee/.ipython']
```

In [146]:

```
### Module (an external file containing definitions)

#-- group related definitions into one file
#-- bring it into the program with an import statement rather than copy/pasting the entire file content

#https://www.tutorialspoint.com/python/python_modules.htm

# 1) put a file in the current working directory: Python only searches the current directory**, the directory that the entry-point script is running from,
# 2) link your document linked to  sys path -- and ** **sys.path, which includes locations**, such as the package installation directory
```

In [161]:

```
 # some_file.py
import sys
sys.path.insert(0, '/Users/byunglee/Documents/workshop')

import os
os.getcwd()
```

Out[161]:

```
'/Users/byunglee'
```

In [154]:

```
# other important commands
import os
os.getcwd()


os.chdir("/tmp/")
os.getcwd()

```

Out[154]:

```
'/private/tmp'
```

In [157]:

```
### • commenting 주석
# use # for a single line comment

#multiline comments

#print("xxx")
#print("xxx")
#print("xxx")

```

In [210]:

```
### module import
os.chdir('/Users/byunglee')
os.getcwd()
```

Out[210]:

```
'/Users/byunglee'
```

In [211]:

```python
# module.py has the following content

def printName(name):
    print("Welcome to this class!")
    return "Your name is " + name
```

In [165]:

```python
import module
newV  = module.printName("Tom")
print(newV)
```

```
Welcome to this class!
Your name is Tom
```

In [167]:

```python
#import built-in modules -- library
import math
print(math.sqrt(9))

# refer to https://docs.python.org/3/library/math.html
```

```
3.0
```

numpy

Numpy is the core library for scientific computing in Python. Create a multidimensional array and manipulate it

random

numpy.random.randn generates samples from the normal distribution, while numpy.random.rand from unifrom (in range [0,1)).

In [169]:

```python
import numpy as np
np.random.rand(3, 2)
```

```
Out[169]: array([[ 0.38602861,  0.94487934],
                 [ 0.34040798,  0.27409932],
                 [ 0.91182075,  0.97156734]])
```

In [170]:

```python
np.random.randn(3, 4)
```

```
Out[170]: array([[ 0.60234611, -0.51654992,  0.69997698, -0.2665834 ],
                 [-0.45858632,  0.25249858,  1.14932614, -0.61969466],
                 [-0.33704434,  0.06649314, -0.60529178,  0.5869642 ]])
```

In [171]:

```python
np.random.randint(2, 5, 9)
```

```
Out[171]: array([2, 2, 3, 4, 2, 4, 4, 4, 4])
```

In [172]:

```python
np.random.randint(5, 9)
```

```
Out[172]: 7
```

In [173]:

```
np.random.randint(1, 7, size=(4, 6))
```

Out[173]:

```
array([[4, 5, 4, 2, 3, 2],
       [4, 4, 4, 2, 6, 3],
       [2, 6, 1, 5, 6, 3],
       [6, 3, 3, 6, 6, 2]])
```

In [175]:

```
#rank: the number of dimensions is the rank of the array;
#shape:  the size of the array along each dimension

import numpy as np
a = np.array([1, 2, 3])   # Create a rank 1 array
print(a.shape)
b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array
print(b.shape)                     # Prints "(2, 3)"
```

```
(3,)
(2, 3)
```

In [180]:

```
# numpy slicing
#Slicing: Similar to Python lists, numpy arrays can be sliced. Since arrays may be multidimensional, you must specify a slice for each dimension of the array:

import numpy as np


a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(a)
# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]
print(b)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[2 3]
 [6 7]]
```

In [177]:

```
import os
os.getcwd()
```

Out[177]:

```
'/Users/byunglee'
```

Datatypes

Every numpy array is a grid of elements of the same type.

Numpy provides a large set of numeric datatypes that you can use to construct arrays.

Numpy tries to guess a datatype when you create an array, but functions that construct arrays usually also include an optional argument to explicitly specify the datatype. Here is an example:

In [184]:

```
import numpy as np

x = np.array([1, 2])   # Let numpy choose the datatype
print(x.dtype)         # Prints "int64"

x = np.array([1.0, 2.0])   # Let numpy choose the datatype
print(x.dtype)             # Prints "float64"

y = np.array([1, 2], dtype = np.float64)
print(y.dtype)
print(type(y))
```

```
int64
float64
float64
<class 'numpy.ndarray'>
```

Array math

Basic mathematical functions operate elementwise on arrays, and are available both as operator overloads and as functions in the numpy module:

In [187]:

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print(x + y)
print(np.add(x, y))
np.subtract(x, y)
np.multiply(x, y)
np.divide(x, y)
print( np.sqrt(x) )

### matrix multiplication
v = np.array([9,10])
w = np.array([11, 12])

# Inner product of vectors; both produce 219
print(v.dot(w))
print(np.dot(v, w))

### transpose
import numpy as np

x = np.array([[1,2], [3,4]])
print(x)    # Prints "[[1 2]
            #          [3 4]]"
print(x.T)  # Prints "[[1 3]
            #          [2 4]]


```

```
[[  6.   8.]
 [ 10.  12.]]
[[  6.   8.]
 [ 10.  12.]]
[[ 1.          1.41421356]
 [ 1.73205081  2.        ]]
219
219
[[1 2]
 [3 4]]
[[1 3]
 [2 4]]
```

In [ ]:

```

```

In [212]:

```
#====================== extra ========
```

In [ ]:

```

```

In [ ]:

```

```

In [164]:

```
import pandas as pd
filepath = 'C:\\Users\\byunglee\\Documents\\workshop\\kyungpook.csv'
df = pd.read_csv(filepath)
```

```
In [165]: df
```

Out[165]:

	id	season	major	startingTime	endingTime	s1	s2	s3	s4	s5	...	s37	s38	s39	s40	s
0	1	group1	o	1504181048	1504181784	0	0	1	1	0	...	1	1	0	0	
1	2	group1	o	1504181068	1504181837	1	0	1	0	0	...	0	0	0	0	
2	3	group1	c	1504181069	1504181840	0	0	1	1	0	...	1	0	0	0	
3	4	group1	o	1504181062	1504181856	1	1	1	1	0	...	1	1	0	0	
4	5	group1	p	1504181056	1504181910	1	1	1	1	0	...	1	1	0	0	
5	6	group1	b	1504181030	1504181942	1	1	1	0	0	...	0	1	1	0	
6	7	group1	p	1504181057	1504181943	1	1	1	0	0	...	0	1	1	0	
7	8	group1	d	1504181088	1504181946	1	1	1	0	0	...	0	1	0	0	
8	9	group1	o	1504181089	1504181962	0	0	1	0	0	...	0	1	1	0	
9	10	group1	p	1504181078	1504181965	0	1	1	0	0	...	0	1	0	0	
10	11	group1	d	1504181072	1504181980	0	1	1	1	0	...	1	0	1	0	
11	12	group1	p	1504181107	1504181989	0	1	0	1	1	...	1	0	1	0	
12	13	group1	o	1504181078	1504182005	1	1	0	0	1	...	0	1	0	0	
13	14	group1	p	1504181070	1504182021	1	0	1	0	0	...	0	0	1	0	
14	15	group1	p	1504181054	1504182024	0	1	1	1	0	...	1	0	1	1	
15	16	group1	p	1504181047	1504182055	0	0	1	1	0	...	0	1	1	0	
16	17	group1	p	1504181097	1504182064	1	1	1	0	0	...	0	1	1	0	
17	18	group1	p	1504181068	1504182098	1	0	1	0	0	...	0	1	0	0	
18	19	group1	c	1504181067	1504182104	0	0	1	1	0	...	0	0	0	0	
19	20	group1	o	1504181053	1504182116	0	1	1	1	0	...	0	1	0	0	
20	21	group1	o	1504181044	1504182140	1	0	0	1	1	...	0	1	0	0	
21	22	group1	o	1504181032	1504182149	0	0	1	1	0	...	0	1	0	1	
22	23	group1	e	1504181087	1504182188	0	0	1	1	1	...	1	1	0	0	
23	24	group1	p	1504181069	1504182198	0	1	0	0	0	...	0	0	0	0	
24	25	group1	p	1504181071	1504182229	0	0	0	0	1	...	0	1	1	0	
25	26	group1	o	1504181069	1504182261	1	0	1	0	0	...	0	0	1	0	
26	27	group1	j	1504181108	1504182271	1	0	0	0	0	...	0	1	1	0	
27	28	group1	p	1504181072	1504182302	1	0	0	1	1	...	0	0	1	1	
28	29	group1	c	1504181078	1504182386	1	1	1	0	0	...	0	1	1	0	
29	30	group1	o	1504181069	1504182391	1	0	0	0	1	...	1	1	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
456	457	group2	p	1526299777	1526301134	1	1	1	0	1	...	1	1	1	1	
457	458	group2	p	1526299954	1526301340	1	1	1	1	1	...	1	1	1	1	

	id	season	major	startingTime	endingTime	s1	s2	s3	s4	s5	...	s37	s38	s39	s40	s
458	459	group2	c	1526300888	1526301900	0	1	1	1	1	...	1	1	0	0	
459	460	group2	c	1526301138	1526301972	1	1	0	0	0	...	1	1	1	0	
460	461	group2	p	1526303159	1526303968	0	0	0	0	0	...	1	1	1	0	
461	462	group2	p	1526304369	1526305086	1	0	1	1	1	...	1	0	1	0	
462	463	group2	p	1526304537	1526305309	1	0	1	0	0	...	0	1	1	0	
463	464	group2	p	1526304982	1526305580	1	0	1	0	1	...	1	1	1	1	
464	465	group2	d	1526305414	1526306390	1	1	0	1	1	...	1	0	1	0	
465	466	group2	p	1526305801	1526306451	1	1	1	1	1	...	1	1	1	0	
466	467	group2	p	1526306059	1526306979	1	1	1	1	1	...	1	1	1	0	
467	468	group2	d	1526306513	1526307276	1	1	1	1	0	...	1	1	1	0	
468	469	group2	p	1526306362	1526307294	1	1	1	1	1	...	1	1	1	1	
469	470	group2	p	1526306529	1526307844	0	1	1	1	0	...	0	1	1	1	
470	471	group2	c	1526306922	1526307856	1	1	1	1	1	...	0	1	1	0	
471	472	group2	d	1526306889	1526308963	0	1	1	0	0	...	1	1	1	1	
472	473	group2	p	1526311327	1526311883	1	1	1	1	1	...	1	1	1	0	
473	474	group2	p	1526311298	1526311995	1	0	1	1	1	...	1	0	1	0	
474	475	group2	p	1526311543	1526312229	0	0	1	0	0	...	1	1	1	0	
475	476	group2	j	1526311805	1526312288	1	1	1	1	1	...	1	1	1	1	
476	477	group2	p	1526311567	1526312321	1	0	1	0	0	...	1	1	1	0	
477	478	group2	j	1526311903	1526312525	1	1	1	1	1	...	0	1	1	1	
478	479	group2	j	1526312072	1526312598	1	1	1	1	1	...	0	1	1	0	
479	480	group2	p	1526311936	1526312696	0	1	1	1	0	...	1	0	1	0	
480	481	group2	c	1526312102	1526312735	1	1	1	1	0	...	0	1	1	1	
481	482	group2	p	1526311920	1526312962	0	1	1	1	0	...	1	1	1	0	
482	483	group2	p	1526311541	1526313003	1	1	1	1	1	...	1	1	1	1	
483	484	group2	c	1526312413	1526313045	1	0	0	0	1	...	1	1	1	1	
484	485	group2	d	1526312253	1526313753	1	1	1	1	1	...	1	1	1	0	
485	486	group2	c	1526311837	1526313856	1	0	1	1	0	...	1	0	1	1	

486 rows × 51 columns

```
In [166]: df.head()
```

Out[166]:

	id	season	major	startingTime	endingTime	s1	s2	s3	s4	s5	...	s37	s38	s39	s40	s41
0	1	group1	o	1504181048	1504181784	0	0	1	1	0	...	1	1	0	0	0
1	2	group1	o	1504181068	1504181837	1	0	1	0	0	...	0	0	0	0	0
2	3	group1	c	1504181069	1504181840	0	0	1	1	0	...	1	0	0	0	0
3	4	group1	o	1504181062	1504181856	1	1	1	1	0	...	1	1	0	0	1
4	5	group1	p	1504181056	1504181910	1	1	1	1	0	...	1	1	0	0	1

5 rows × 51 columns

```
In [167]: df.tail(10)
```

Out[167]:

	id	season	major	startingTime	endingTime	s1	s2	s3	s4	s5	...	s37	s38	s39	s40	s
476	477	group2	p	1526311567	1526312321	1	0	1	0	0	...	1	1	1	0	
477	478	group2	j	1526311903	1526312525	1	1	1	1	1	...	0	1	1	1	
478	479	group2	j	1526312072	1526312598	1	1	1	1	1	...	0	1	1	0	
479	480	group2	p	1526311936	1526312696	0	1	1	1	0	...	1	0	1	0	
480	481	group2	c	1526312102	1526312735	1	1	1	1	0	...	0	1	1	1	
481	482	group2	p	1526311920	1526312962	0	1	1	1	0	...	1	1	1	0	
482	483	group2	p	1526311541	1526313003	1	1	1	1	1	...	1	1	1	1	
483	484	group2	c	1526312413	1526313045	1	0	0	0	1	...	1	1	1	1	
484	485	group2	d	1526312253	1526313753	1	1	1	1	1	...	1	1	1	0	
485	486	group2	c	1526311837	1526313856	1	0	1	1	0	...	1	0	1	1	

10 rows × 51 columns

```
In [169]: df.columns

```

```
Out[169]: Index(['id', 'season', 'major', 'startingTime', 'endingTime', 's1', 's2',
's3',
       's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's
14',
       's15', 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's2
4',
       's25', 's26', 's27', 's28', 's29', 's30', 's31', 's32', 's33', 's3
4',
       's35', 's36', 's37', 's38', 's39', 's40', 's41', 's42', 's43', 's4
4',
       's45', 'totalScore'],
      dtype='object')
```

In [170]:

```
df.describe()
```

Out[170]:

	id	startingTime	endingTime	s1	s2	s3	s4	
count	486.000000	4.860000e+02	4.860000e+02	486.000000	486.000000	486.000000	486.000000	48
mean	243.500000	1.511581e+09	1.511583e+09	0.569959	0.584362	0.648148	0.518519	
std	140.440379	1.030964e+07	1.030922e+07	0.495592	0.493339	0.478040	0.500172	
min	1.000000	1.504181e+09	1.504182e+09	0.000000	0.000000	0.000000	0.000000	
25%	122.250000	1.504197e+09	1.504198e+09	0.000000	0.000000	0.000000	0.000000	
50%	243.500000	1.504283e+09	1.504284e+09	1.000000	1.000000	1.000000	1.000000	
75%	364.750000	1.525967e+09	1.525968e+09	1.000000	1.000000	1.000000	1.000000	
max	486.000000	1.526312e+09	1.526314e+09	1.000000	1.000000	1.000000	1.000000	

8 rows × 49 columns

In [186]:

```
#selection by label
dataset0 = df["s1"]
```

In [187]:

```
dataset0
```

Out[187]:

```
0      0
1      1
2      0
3      1
4      1
5      1
6      1
7      1
8      0
9      0
10     0
11     0
12     1
13     1
14     0
15     0
16     1
17     1
18     0
19     0
20     1
21     0
22     0
23     0
24     0
25     1
26     1
27     1
28     1
29     1
      ..
456    1
457    1
458    0
459    1
460    0
461    1
462    1
463    1
464    1
465    1
466    1
467    1
468    1
469    0
470    1
471    0
472    1
473    1
474    0
475    1
476    1
477    1
478    1
479    0
```

```
480    1
481    0
482    1
483    1
484    1
485    1
Name: s1, Length: 486, dtype: int64
```

```
In [172]:


dataset1 = df.loc[:, 's1']
```

```
In [173]: type(dataset1)

Out[173]: pandas.core.series.Series
```

```
In [174]: dataset2 = df.loc[:, ['s1', 's3']]
```

```
In [175]: type(dataset2)

Out[175]: pandas.core.frame.DataFrame
```

```
In [176]: dataset3 = df.loc[:, "s1":"s5"]
```

```
In [177]: type(dataset3)

Out[177]: pandas.core.frame.DataFrame
```

```
In [178]:  dataset3
```

```
Out[178]:
        s1  s2  s3  s4  s5
     0   0   0   1   1   0
     1   1   0   1   0   0
     2   0   0   1   1   0
     3   1   1   1   1   0
     4   1   1   1   1   0
     5   1   1   1   0   0
     6   1   1   1   0   0
     7   1   1   1   0   0
     8   0   0   1   0   0
     9   0   1   1   0   0
    10   0   1   1   1   0
    11   0   1   0   1   1
    12   1   1   0   0   1
    13   1   0   1   0   0
    14   0   1   1   1   0
    15   0   0   1   1   0
    16   1   1   1   0   0
    17   1   0   1   0   0
    18   0   0   1   1   0
    19   0   1   1   1   0
    20   1   0   0   1   1
    21   0   0   1   1   0
    22   0   0   1   1   1
    23   0   1   0   0   0
    24   0   0   0   0   1
    25   1   0   1   0   0
    26   1   0   0   0   0
    27   1   0   0   1   1
    28   1   1   1   0   0
    29   1   0   0   0   1
   ...  ...  ...  ...  ...  ...
   456   1   1   1   0   1
   457   1   1   1   1   1
```

	s1	s2	s3	s4	s5
458	0	1	1	1	1
459	1	1	0	0	0
460	0	0	0	0	0
461	1	0	1	1	1
462	1	0	1	0	0
463	1	0	1	0	1
464	1	1	0	1	1
465	1	1	1	1	1
466	1	1	1	1	1
467	1	1	1	1	0
468	1	1	1	1	1
469	0	1	1	1	0
470	1	1	1	1	1
471	0	1	1	0	0
472	1	1	1	1	1
473	1	0	1	1	1
474	0	0	1	0	0
475	1	1	1	1	1
476	1	0	1	0	0
477	1	1	1	1	1
478	1	1	1	1	1
479	0	1	1	1	0
480	1	1	1	1	0
481	0	1	1	1	0
482	1	1	1	1	1
483	1	0	0	0	1
484	1	1	1	1	1
485	1	0	1	1	0

486 rows × 5 columns

In [180]:

```
#select by integer position
dataset4 = df.iloc[:, 3:6]
```

In [181]:

```
type(dataset4)
```

```
Out[181]: pandas.core.frame.DataFrame
```

In [183]:

```
dataset4["timeSpent"] = dataset4["endingTime"] -
dataset4["startingTime"]
```

```
In [184]: dataset4
```

```
Out[184]:
            startingTime  endingTime  s1  timeSpent
        0     1504181048  1504181784   0        736
        1     1504181068  1504181837   1        769
        2     1504181069  1504181840   0        771
        3     1504181062  1504181856   1        794
        4     1504181056  1504181910   1        854
        5     1504181030  1504181942   1        912
        6     1504181057  1504181943   1        886
        7     1504181088  1504181946   1        858
        8     1504181089  1504181962   0        873
        9     1504181078  1504181965   0        887
       10     1504181072  1504181980   0        908
       11     1504181107  1504181989   0        882
       12     1504181078  1504182005   1        927
       13     1504181070  1504182021   1        951
       14     1504181054  1504182024   0        970
       15     1504181047  1504182055   0       1008
       16     1504181097  1504182064   1        967
       17     1504181068  1504182098   1       1030
       18     1504181067  1504182104   0       1037
       19     1504181053  1504182116   0       1063
       20     1504181044  1504182140   1       1096
       21     1504181032  1504182149   0       1117
       22     1504181087  1504182188   0       1101
       23     1504181069  1504182198   0       1129
       24     1504181071  1504182229   0       1158
       25     1504181069  1504182261   1       1192
       26     1504181108  1504182271   1       1163
       27     1504181072  1504182302   1       1230
       28     1504181078  1504182386   1       1308
       29     1504181069  1504182391   1       1322
      ...            ...         ...  ...       ...
      456     1526299777  1526301134   1       1357
      457     1526299954  1526301340   1       1386
```

	startingTime	endingTime	s1	timeSpent
458	1526300888	1526301900	0	1012
459	1526301138	1526301972	1	834
460	1526303159	1526303968	0	809
461	1526304369	1526305086	1	717
462	1526304537	1526305309	1	772
463	1526304982	1526305580	1	598
464	1526305414	1526306390	1	976
465	1526305801	1526306451	1	650
466	1526306059	1526306979	1	920
467	1526306513	1526307276	1	763
468	1526306362	1526307294	1	932
469	1526306529	1526307844	0	1315
470	1526306922	1526307856	1	934
471	1526306889	1526308963	0	2074
472	1526311327	1526311883	1	556
473	1526311298	1526311995	1	697
474	1526311543	1526312229	0	686
475	1526311805	1526312288	1	483
476	1526311567	1526312321	1	754
477	1526311903	1526312525	1	622
478	1526312072	1526312598	1	526
479	1526311936	1526312696	0	760
480	1526312102	1526312735	1	633
481	1526311920	1526312962	0	1042
482	1526311541	1526313003	1	1462
483	1526312413	1526313045	1	632
484	1526312253	1526313753	1	1500
485	1526311837	1526313856	1	2019

486 rows × 4 columns

```
In [191]:  dataset4.mean()
```

```
Out[191]: startingTime    1.511581e+09
          endingTime      1.511583e+09
          s1              5.699588e-01
          timeSpent       1.562549e+03
          dtype: float64
```

In [192]:

```
dataset5 = dataset4[dataset4['timeSpent'] > 1563]
#boolean selection
dataset5
```

Out[192]:

	startingTime	endingTime	s1	timeSpent
122	1504196837	1504198800	1	1963
190	1504268869	1504270598	1	1729
222	1504279562	1504281179	1	1617
248	1504283139	1504284825	0	1686
249	1504283143	1504285260	0	2117
253	1504283134	1504285607	0	2473
254	1504283206	1504285685	0	2479
259	1504283077	1504285738	0	2661
260	1504283169	1504285751	0	2582
265	1504283094	1504285826	1	2732
286	1504284891	1504286680	0	1789
287	1504268970	1504379789	1	110819
288	1504283099	1504456725	0	173626
351	1525966369	1525968003	0	1634
397	1526044087	1526045665	1	1578
438	1526065213	1526068113	1	2900
440	1526066818	1526069801	1	2983
471	1526306889	1526308963	0	2074
485	1526311837	1526313856	1	2019

In [194]:

```
dataset5["timeSpent"].mean()
```

Out[194]:

```
16919.0
```

In [196]:

```
import numpy as np
data1 = np.random.randint(0, 9, size=50)
```

In [197]:

```
data1
```

Out[197]:

```
array([2, 1, 5, 0, 3, 0, 5, 7, 1, 1, 7, 5, 1, 6, 3, 2, 0, 1, 5, 5, 2, 8,
       5, 1, 2, 1, 2, 2, 1, 0, 6, 5, 5, 6, 8, 1, 7, 6, 1, 5, 1, 5, 3, 8,
       7, 2, 5, 2, 6, 6])
```

In [198]:

```
dataset1 = pd.Series(data1)
```

In [199]:

```
dataset1
```

Out[199]:

```
0     2
1     1
2     5
3     0
4     3
5     0
6     5
7     7
8     1
9     1
10    7
11    5
12    1
13    6
14    3
15    2
16    0
17    1
18    5
19    5
20    2
21    8
22    5
23    1
24    2
25    1
26    2
27    2
28    1
29    0
30    6
31    5
32    5
33    6
34    8
35    1
36    7
37    6
38    1
39    5
40    1
41    5
42    3
43    8
44    7
45    2
46    5
47    2
48    6
49    6
dtype: int32
```

In [202]:

```
hist1 = dataset1.value_counts()
```

In [203]:

```
hist1
```

Out[203]:

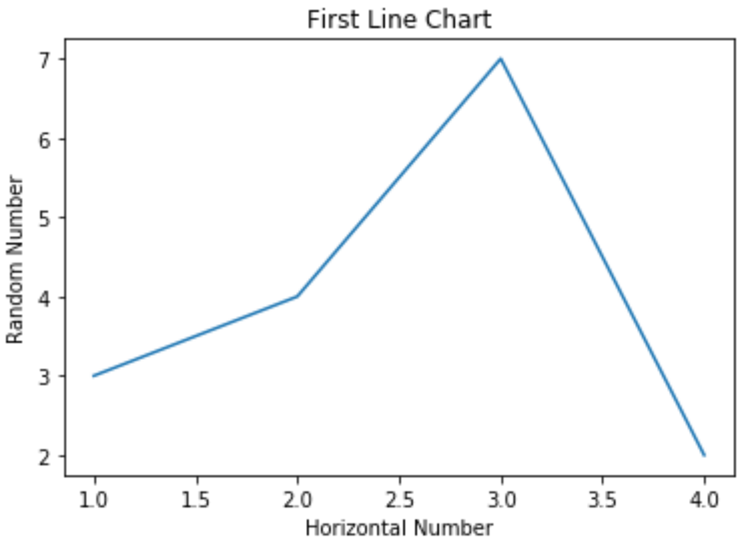
```
5    11
1    11
2     8
6     6
7     4
0     4
8     3
3     3
dtype: int64
```

In [ ]:

```

```

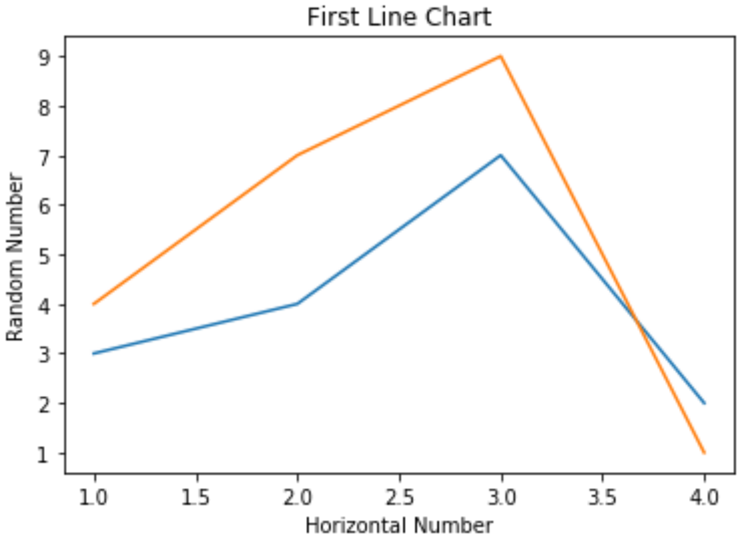
In [207]:

```
#visualization -- line chart
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot([1,2, 3, 4], [3, 4, 7, 2])
plt.xlabel("Horizontal Number")
plt.ylabel("Random Number")
plt.title("First Line Chart")
plt.show()

```



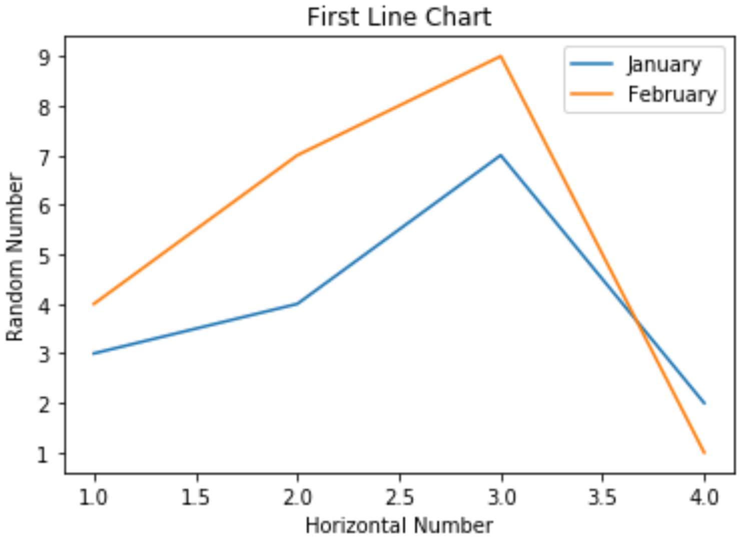
In [208]:

```
#visualization -- line chart
%matplotlib inline
import matplotlib.pyplot as plt
x = [1,2, 3, 4]
y =  [3, 4, 7, 2]
y2 = [4, 7, 9, 1]
plt.plot(x,y)
plt.plot(x, y2)
plt.xlabel("Horizontal Number")
plt.ylabel("Random Number")
plt.title("First Line Chart")
plt.show()

```



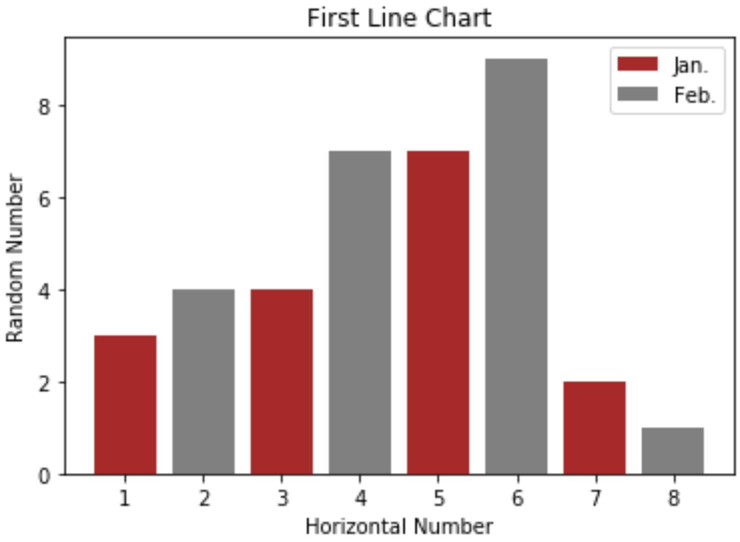
In [210]:

```
#visualization -- line chart
%matplotlib inline
import matplotlib.pyplot as plt
x = [1,2, 3, 4]
y =  [3, 4, 7, 2]
y2 = [4, 7, 9, 1]
plt.plot(x,y, label = 'January')
plt.plot(x, y2, label = "February")
plt.xlabel("Horizontal Number")
plt.ylabel("Random Number")
plt.title("First Line Chart")
plt.legend()
plt.show()
```



In [228]:

```
#visualization -- bar chart
%matplotlib inline
import matplotlib.pyplot as plt
x = [1,3, 5, 7]
y =  [3, 4, 7, 2]
x2 = [2, 4, 6, 8]
y2 = [4, 7, 9, 1]

plt.bar(x, y, label = "Jan.", color = "brown" )
plt.bar(x2, y2, label = "Feb.", color = "grey")
#plt.bar(x2, y2, label = "Feb.")


plt.xlabel("Horizontal Number")
plt.ylabel("Random Number")
plt.title("First Line Chart")
plt.legend()

plt.show()
```

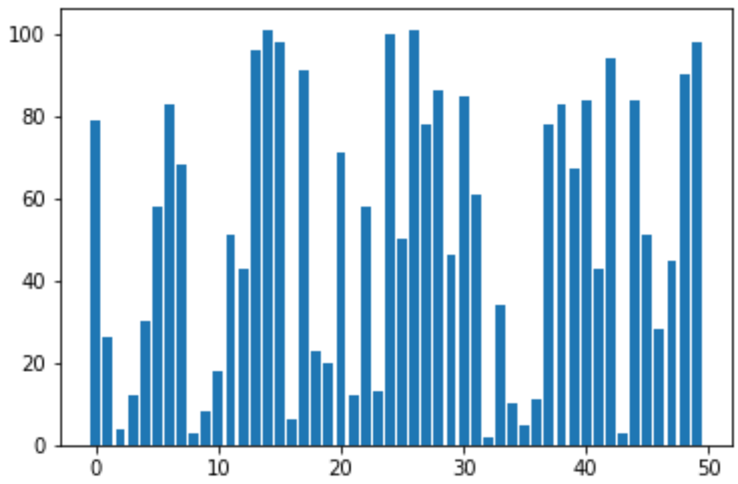


In [231]:

```
# histograms

import matplotlib.pyplot as plt

#age = [12, 34, 22, 91, 23, 45]
age = []
import random
for x in range(50):
    age.append( random.randint(1, 101))

print(age)

x = [x for x in range(len(age))]
plt.bar(x, age)
plt.show()
```

```
[79, 26, 4, 12, 30, 58, 83, 68, 3, 8, 18, 51, 43, 96, 101, 98, 6, 91, 23,
20, 71, 12, 58, 13, 100, 50, 101, 78, 86, 46, 85, 61, 2, 34, 10, 5, 11, 7
8, 83, 67, 84, 43, 94, 3, 84, 51, 28, 45, 90, 98]
```

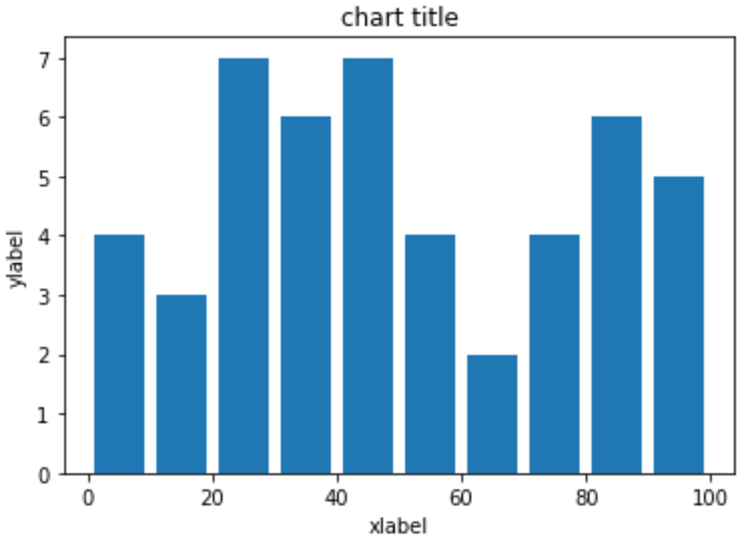
Out[231]: <BarContainer object of 50 artists>



In [238]:

```
# histograms

import matplotlib.pyplot as plt

#age = [12, 34, 22, 91, 23, 45]
age = []
import random
for x in range(50):
    age.append( random.randint(1, 101))

print(age)

#x = [x for x in range(len(age))]
#plt.bar(x, age)

bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.hist(age, bins, histtype = "bar", rwidth = 0.8)

plt.xlabel("xlabel")
plt.ylabel("ylabel")
plt.title("chart title")

plt.show()
```

```
[48, 30, 101, 40, 1, 85, 23, 13, 82, 14, 13, 28, 9, 85, 62, 25, 38, 4, 4
2, 35, 101, 25, 74, 94, 61, 73, 28, 74, 48, 33, 38, 85, 45, 98, 45, 57,
2, 58, 98, 90, 25, 96, 72, 59, 83, 39, 89, 57, 20, 49]
```

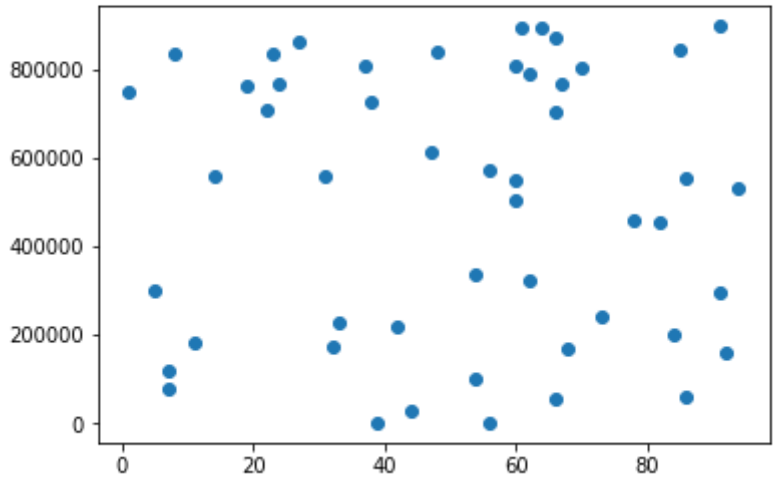


In [243]:

```
#scatter plot
# histograms

import matplotlib.pyplot as plt

#age = [12, 34, 22, 91, 23, 45]
age = []
import random
for x in range(50):
    age.append( random.randint(1, 101))

print(age)

income = [ random.randint(1000, 1000000) for x in range(50) ]
print("INCOME")
print(income)
print(len(age), len(income))
plt.scatter(age, income)
```

```
[56, 91, 78, 66, 7, 39, 62, 66, 24, 68, 38, 60, 61, 91, 19, 5, 66, 60, 9
4, 11, 23, 27, 1, 73, 48, 44, 92, 64, 86, 32, 47, 62, 8, 14, 33, 82, 60,
31, 54, 37, 67, 7, 86, 56, 84, 85, 70, 22, 54, 42]
INCOME
[2114, 293887, 457026, 704867, 119952, 1767, 789665, 54003, 767318, 16925
9, 724795, 549187, 893029, 896634, 760677, 302025, 871366, 503596, 52966
8, 183680, 834056, 863487, 746453, 240141, 840167, 30760, 161147, 895332,
551547, 175365, 612583, 323616, 832405, 558167, 226552, 455592, 807734, 5
57839, 100898, 808972, 766208, 80266, 61002, 570301, 200953, 842745, 8043
46, 707465, 337141, 218220]
50 50
```

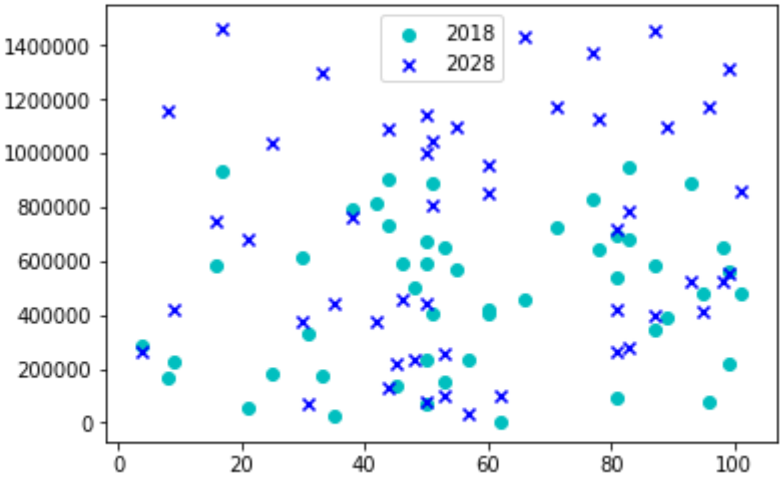
Out[243]: <matplotlib.collections.PathCollection at 0x27e8ed8e588>



In [246]:

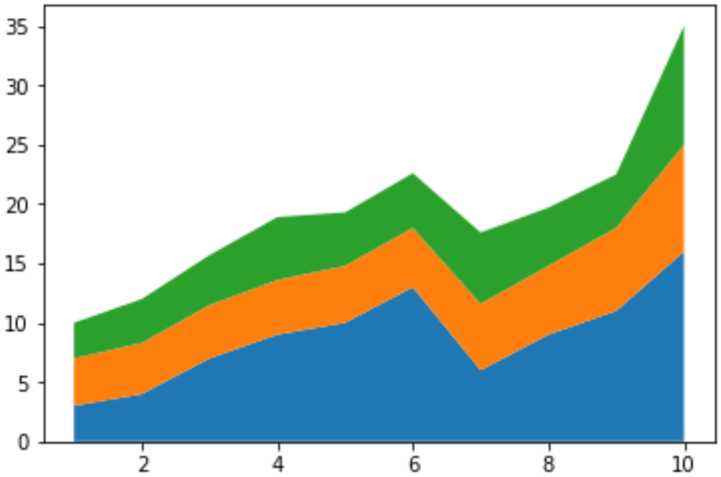
```python
#scatter plot
# histograms

import matplotlib.pyplot as plt

#age = [12, 34, 22, 91, 23, 45]
age = []
import random
for x in range(50):
    age.append( random.randint(1, 101))

print(age)

income = [ random.randint(1000, 1000000) for x in range(50) ]
income2 = [ random.randint(1500, 1500000) for x in range(50) ]


plt.scatter(age, income, marker = 'o', color = "c", label = "2018")
plt.scatter(age, income2, marker = 'x', color = "b", label = "2028")

plt.legend()
plt.show()
```

```
[87, 48, 99, 50, 44, 17, 99, 55, 25, 83, 35, 30, 50, 93, 81, 57, 89, 101,
42, 95, 60, 33, 21, 77, 81, 45, 50, 98, 96, 51, 60, 44, 71, 53, 4, 8, 46,
9, 16, 31, 51, 87, 83, 81, 66, 50, 78, 62, 53, 38]
```



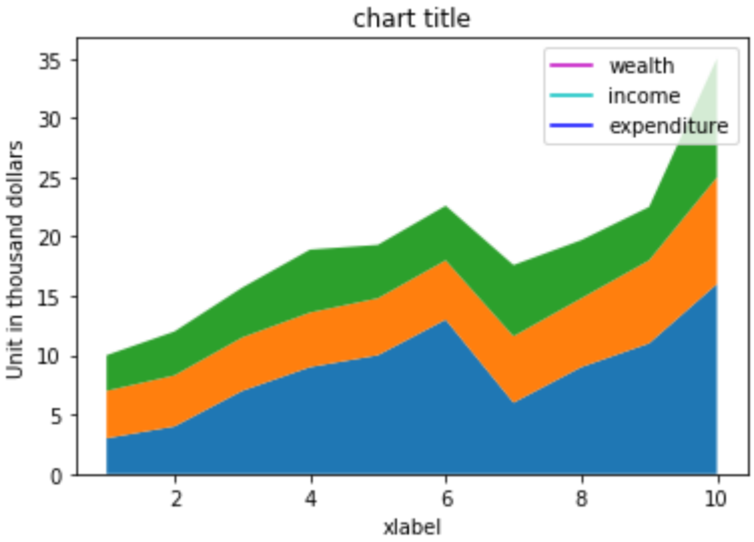
In [249]:

```
# stack plot
import matplotlib.pyplot as plt

year = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

wealth = [3, 4, 7, 9, 10, 13, 6, 9, 11, 16]
income =      [4, 4.3, 4.5, 4.6, 4.8, 5, 5.6, 5.8, 7, 9]
expenditure = [3, 3.7, 4.2, 5.3, 4.5, 4.6, 6, 4.9, 4.5, 10 ]

plt.stackplot (year, wealth, income, expenditure)
plt.show()
```



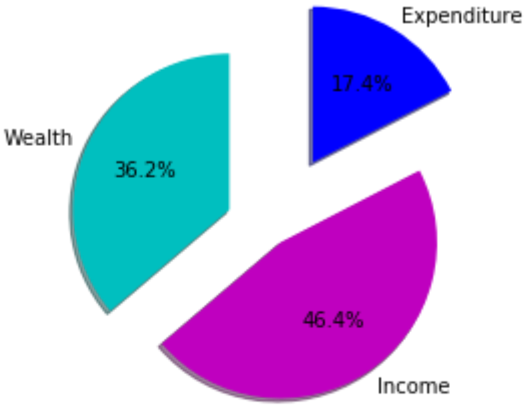
In [252]:

```
# stack plot
import matplotlib.pyplot as plt

year = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# unit in thousands
wealth = [3, 4, 7, 9, 10, 13, 6, 9, 11, 16]
income =      [4, 4.3, 4.5, 4.6, 4.8, 5, 5.6, 5.8, 7, 9]
expenditure = [3, 3.7, 4.2, 5.3, 4.5, 4.6, 6, 4.9, 4.5, 10 ]

# legend
plt.plot([], [], color = 'm', label = "wealth")
plt.plot([], [], color = 'c', label = "income")
plt.plot([], [], color = 'b', label = "expenditure")

plt.stackplot (year, wealth, income, expenditure)
plt.legend()
plt.xlabel("xlabel")
plt.ylabel("Unit in thousand dollars")
plt.title("chart title")
plt.show()
```



In [262]:

```
# pie chart
import matplotlib.pyplot as plt
labels3 = 'Wealth', "Income", "Expenditure"
sizes =  [25, 32, 12]
colors3 = ["c", "m", "b"]
plt.pie(sizes, labels = labels3, colors = colors3, startangle= 90, shadow = True, explode = (0.3, 0.1, 0.5), autopct = "%1.1f%%") # startangle???
plt.axis('equal')  #equal makes the circle
plt.show()
```



In [263]:

```
#statistics

national = pd.DataFrame(["white"]*100000 + ["hispanic"]*60000 +\
                        ["black"]*50000 + ["asian"]*15000 + ["other"]*35000)


minnesota = pd.DataFrame(["white"]*600 + ["hispanic"]*300 + \
                         ["black"]*250 +["asian"]*75 + ["other"]*150)

national_table = pd.crosstab(index=national[0], columns="count")
minnesota_table = pd.crosstab(index=minnesota[0], columns="count")

print( "National")
print(national_table)
print(" ")
print( "Minnesota")
print(minnesota_table)
```

```
National
col_0      count
0
asian      15000
black      50000
hispanic   60000
other      35000
white     100000

Minnesota
col_0     count
0
asian        75
black       250
hispanic    300
other       150
white       600
```

In [266]:

```
national = pd.Series( ["white"]*10 + ["black"]*5)
```

```
In [272]: type(national)
          national
```

```
Out[272]: 0     white
          1     white
          2     white
          3     white
          4     white
          5     white
          6     white
          7     white
          8     white
          9     white
          10    black
          11    black
          12    black
          13    black
          14    black
          dtype: object
```

```
In [276]: local = pd.DataFrame( ["white"]*10 + ["black"]*5)
          pd.crosstab(local[0], columns = "count")
```

Out[276]:

col_0	count
0	
black	5
white	10

In [273]:

```
print(type(local))
local
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[273]:

```
     0
 0   white
 1   white
 2   white
 3   white
 4   white
 5   white
 6   white
 7   white
 8   white
 9   white
10   black
11   black
12   black
13   black
14   black
```

In [274]:

```
pd.DataFrame(["white", "black", "white", "black","white",
"black","white" ])
```

Out[274]:

```
    0
0   white
1   black
2   white
3   black
4   white
5   black
6   white
```

In [291]:

```
#statistics

national = pd.DataFrame(["white"]*100000 + ["hispanic"]*60000 +\
                        ["black"]*50000 + ["asian"]*15000 +
["other"]*35000)


minnesota = pd.DataFrame(["white"]*600 + ["hispanic"]*300 + \
                         ["black"]*250 +["asian"]*75 + ["other"]*150)

#national_table = pd.crosstab(  columns="count")
national_table = pd.crosstab(index=national[0], columns="count")
minnesota_table = pd.crosstab(index=minnesota[0], columns="count")

print(national_table.shape)
print(national_table)

print(minnesota_table)
```

```
(5, 1)
col_0      count
0
asian      15000
black      50000
hispanic   60000
other      35000
white     100000
col_0     count
0
asian        75
black       250
hispanic    300
other       150
white       600
```

In [ ]:

```

```

In [ ]:

```

```