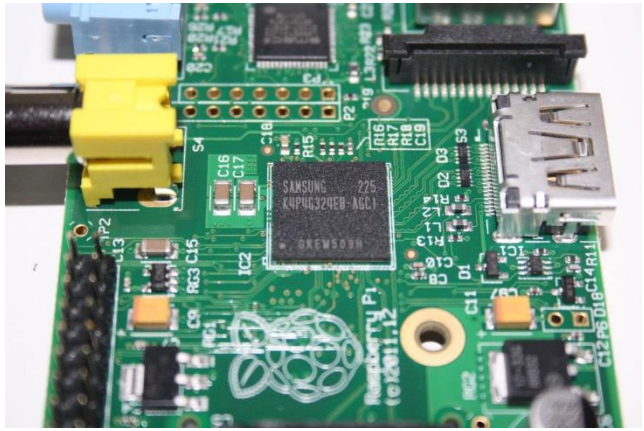
RaspberryPi

# Raspberry pi interfacing

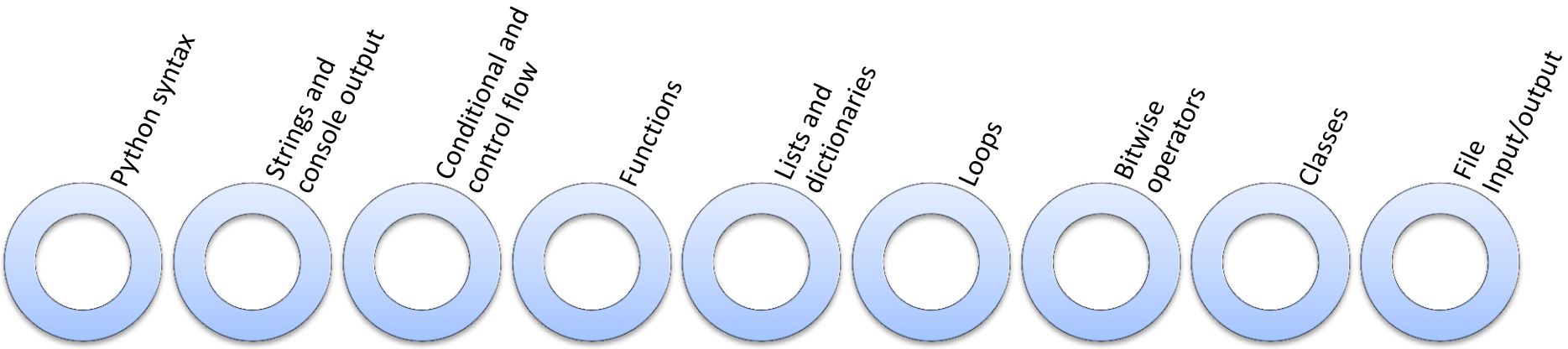
## Lecture 5: Python for raspberry pi



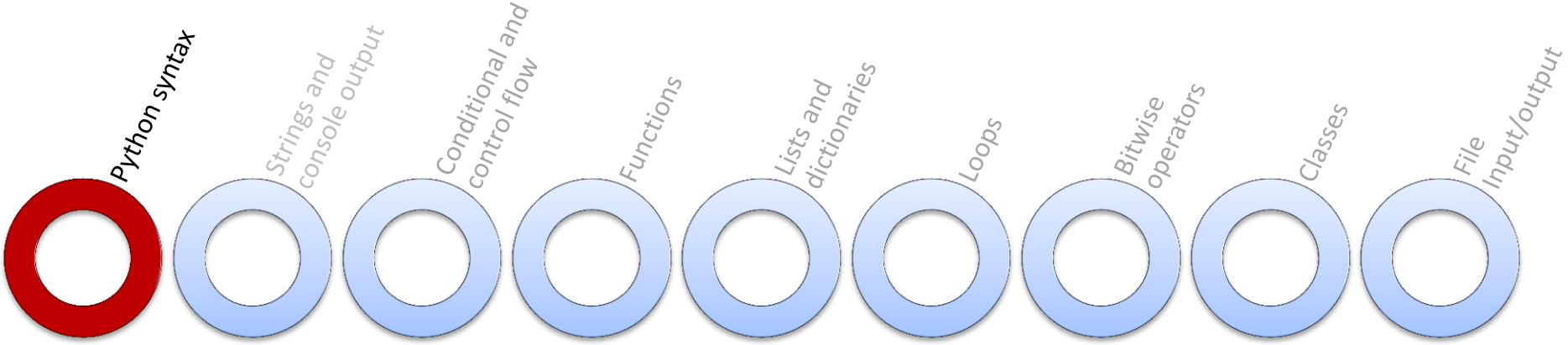
# Python Agenda

- Python syntax.
- Strings and console output.
- Conditional and control flow.
- Functions.
- Lists and dictionaries.
- Loops.
- Bitwise operators.
- Classes.
- File Input/output

# Python Agenda



# Python syntax



# Python syntax

- **#!/usr/bin/python**
  - It should be written at the start of any python script to tell the shell to execute the script using python, so the script should have execute permission.
- **print "Hello"**
  - Prints Hello to console
- **x = 3**
  - Define integer variable x with value 3
- **y = 4.5**
  - Define float variable y with value 4.5
- **z = True**
  - Define Boolean variable z with value True
- **x = 5/2**
  - Value of x will be 2 as operation is all on integers

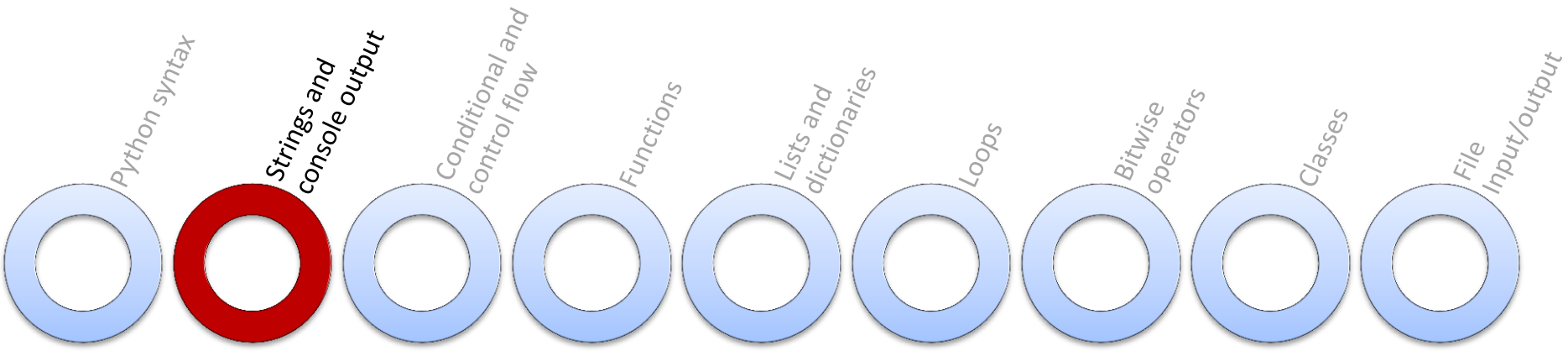
# Python syntax

- **x = 5.0/2**
  - Value of x will be 2.5
- **x = float(5)/2**
  - Value of x will be 2.5
- **# comment**
  - # is used to comment out any line
- **“””Comment multiple lines”””**
  - “”” any number of lines “”” is used to comment out any number of lines.

# Python syntax

- **x = 10 + 20**
  - X is sum of 10 and 20
- **x = 4 ** 2**
  - X is 4 to the power of 2
- **x = 101 % 2**
  - X is 101 modulo 2 which will equal 1

# Strings and console output



# Strings and console output

- **var = “Hello Python”**
  - Put string “Hello Python” in variable var
- **c = “Hello”[0]**
  - Put character at index zero of string “Hello” which is ‘H’ in variable c, so c will contain the value ‘H’
- **c = var[4]**
  - c will have the value at index 4 which is ‘o’
- **len(var)**
  - Returns the length of the given string
- **d = str(32)**
  - d will contain the string representation of the value 32, so d will contain “32” as a string.

# Strings and console output

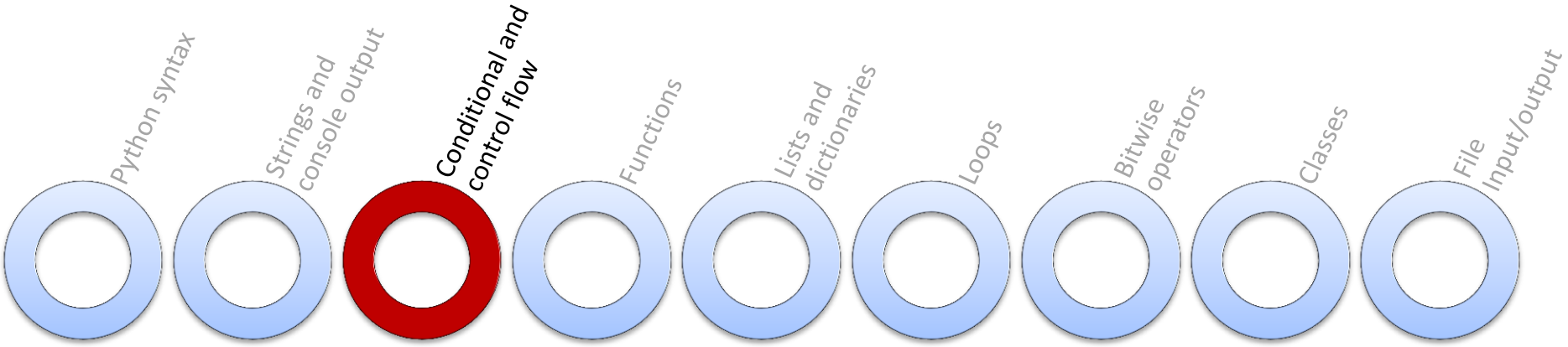
- **print var.lower()**
  - Print string in variable var with all letters lower case.
- **print var.upper()**
  - Print string in variable var with all letters upper case.
- **mystr = var + “Scripting”**
  - Concatenate strings, value of mystr will be “Hello Python Scripting”
- **print “Welcome” + “ to” + “ python”**
  - Prints Welcome to python
- **mystr = var[2:len(var)]**
  - Variable mystr will contain string inside var from index 2 (third character) till the end of the string inside var.
- **Print “value1 is %s, value2 is %s” %(var1, var2)**
  - Prints the values inside variables var1 and var2.

# Strings and console output

- **Date and time**

```
# import is used to include external library to be used inside our script
from datetime import datetime
now = datetime.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)
print ("%s/%s/%s %s:%s:%s") % (now.month, now.day, now.year, now.hour, now.minute, now.second)
```

# Conditional and control flow



# Conditional and control flow

- **Comparators > == <= <**
  - Compare between two values, returns True or False
- **and, or operators**
  - Combine between 2 or more conditions
- **not operator**
  - Returns the inverse of the condition
  - not is evaluated first, and is evaluated next, or is evaluated last. So use Parentheses better

# Conditional and control flow

- **If syntax**

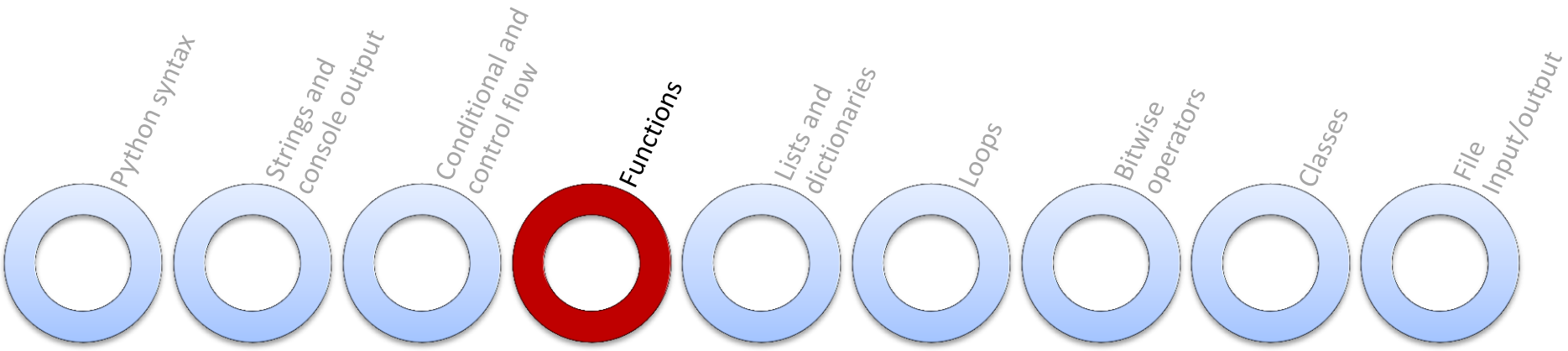
```
if x>10 :  #note the ':'
        print "Greater"     # note the indentation (4 white spaces)
elif x<10 :
        print "Less"
else :
        print "Equal"
```

# Conditional and control flow

- **Take input from user**

```
name = raw_input("What's your name?")  # returns string

integer = int(raw_input("Enter number: "))  # convert returned string to integer

integer = input("enter integer")  # returns integer

if name.isalpha()        # check that name contains alphabetical characters only
```

# Functions


Python syntax
Strings and console output
Conditional and control flow
Functions
Lists and dictionaries
Loops
Bitwise operators
Classes
File Input/output

# Functions

- **Python functions**

```
import math
print math.sqrt(25)     # print square root of 25

from math import sqrt           # to be able to use sqrt by its name
print sqrt(25)

import math          # Imports the math module
everything = dir(math) # Sets everything to a list of function from math
print everything      # Prints all available functions in math module
```

# Functions

- **Python functions**

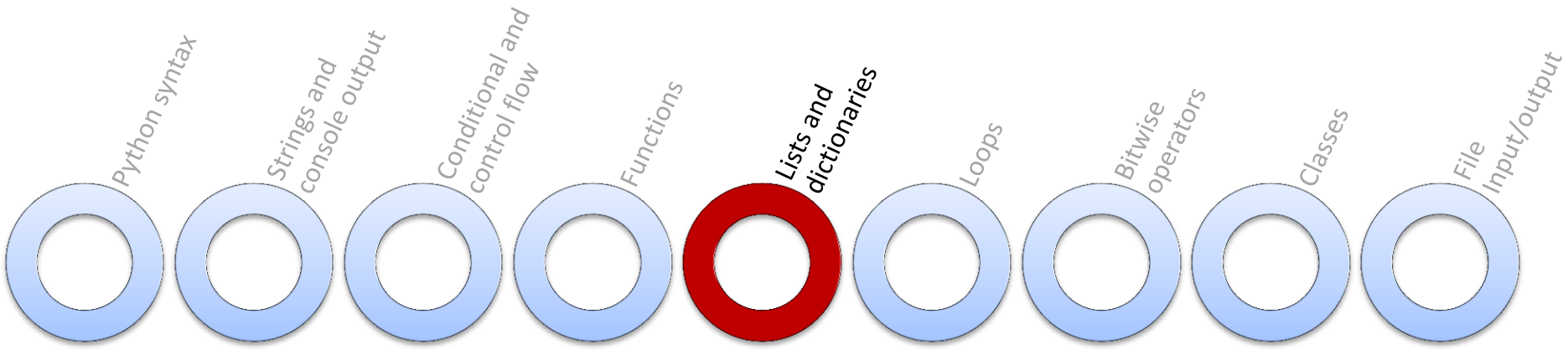
```
maximum = max(-2,3,1,15)      # return maximum of given values
minimum = min(3,7,2,9)        # return maximum of given values
absolute = abs(-42)     # return absolute value of given number

print type(3)             # prints  <type 'int'>
print type(2.4)           # prints <type 'float'>
print type("Hello")       # prints  <type 'str'>

#no need to import any module to be able to use previous functions as they are built in functions.
```

# Functions

- **User defined functions**

```
def print_positive(arg) :          # take one argument
    if agr>=0 :
        print arg
        return True          # return value to the caller
    else :
        return False
```

# Lists and dictionaries



# Lists and dictionaries

- **list = [item1, item2, item3]**
  - Create list of the given 3 items
- **list.append(newItem)**
  - Add new element at the end of the list
- **list[0:3]**
  - Values of the list from index 0 and stop before index 3
- **list[:2]**
  - Grabs the first two items
- **list[3:]**
  - Grabs the fourth through last items
- **string_index = list.index(“mystring")**
  - Find index of “mystring"

# Lists and dictionaries

- **list.insert(index, item)**
  - insert item at index and shift all next items down by 1
- **list.sort()**
  - sort list ascending
- **list.remove(item3)**
  - remove item3 from list
- **Loop on all list items**

```
my_list = [1,9,3,8,5,7]
for number in my_list:
    print 2 * number              # print 2 18 6 16 10 14
```

# Lists and dictionaries

- **students= {‘std1' : 90, ‘std2' : 85, ‘std2' : 93}**
  - Assigning a dictionary with three key-value pairs to students
- **print students[‘std1']**
  - Prints std1 grade which is 90
- **dict_name[new_key] = new_value**
  - add new value to dictionary
- **del dict_name[key_name]**
  - delete key-value pair from dictionary
- **dict_name[key] = new_value**
  - change value

# Lists and dictionaries

```
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
# print even numbers in list
for number in a :
    if number%2 == 0 :
        print number


# loop on string characters
for letter in “Mystring":
    print letter              # print all letters each one on separate line
```

# Lists and dictionaries

```
# function take list as argument
def count_small(numbers):       # returns number of values less than 10
    total = 0
    for n in numbers:
        if n < 10:
            total = total + 1
    return total

num = [4, 8, 15, 16, 23, 42]
small = count_small(num)
print small     # print number of values in list that are less than 10
```

# Lists and dictionaries

```
# shop example
shopping_list = ["banana", "orange", "apple"]
stock = {"banana": 6, "apple": 0, "orange": 32}
prices = {"banana": 4, "apple": 2, "orange": 1.5}

def compute_bill(food) :          # compute the total price of given food list
    total = 0
    for item in food :      # loop on all items in food list
        if stock[item]>0 :              # if item exists in stock
            stock[item]-=1              # reduce stock by 1
            total+=prices[item]         # add item price to total
    return total

Print compute_bill(shopping_list)
```

# Lists and dictionaries

```
n = [1, 3, 5]
n.pop(1)
# Returns 3 (the item at index 1) and remove it from list

n.remove(1)
# Removes 1 from the list,
# NOT the item at index 1

del(n[1])
# Doesn't return anything, but removes item at index 1
```

# Lists and dictionaries

```
def my_function(x):
    for i in range(0, len(x)):          # loop on all indices in list x
        x[i] = x[i] * 2        # change value at index i with the double of this value
    return x      # return the edited list
```

- Note

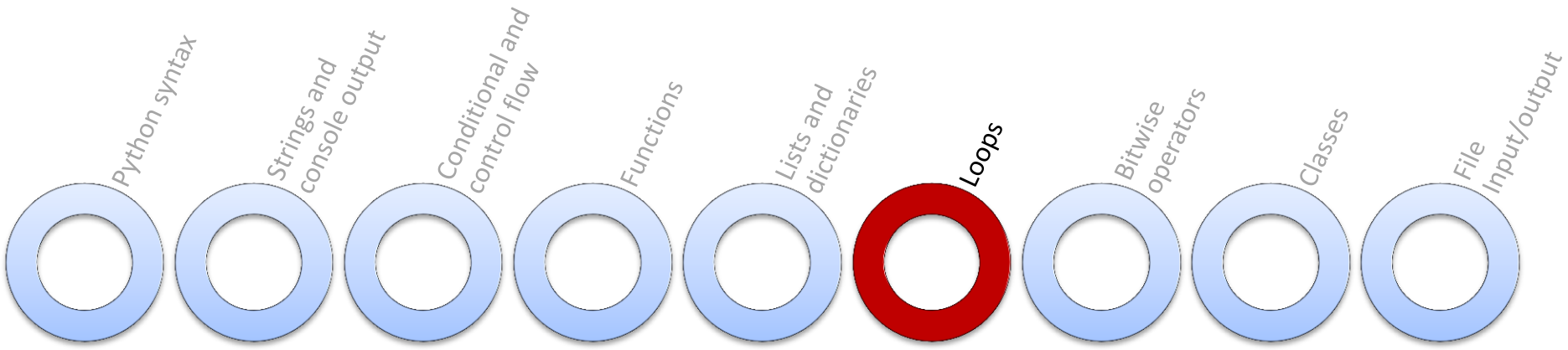
```
range(6)                    # [0,1,2,3,4,5]
range(1,6)                  # [1,2,3,4,5]
range(1,6,3)                 # [1,4] from 1 to 6 with step 3
```

```
letters = ['a', 'b', 'c', 'd']
print " ".join(letters)             # prints a b c d
print "---".join(letters)           # prints a---b---c---d
```

# Lists and dictionaries

```
# list comprehension
evens_to_50 = [i for i in range(51) if i % 2 == 0]    # list of even numbers till 50


my_list[::2]   # from start to end with stride of 2
my_list[::-1]   # from end to start with stride of 1
squares = [x**2 for x in range(5)]   #  contains squares of numbers from 0 to 4
```

# Loops


Python syntax
Strings and console output
Conditional and control flow
Functions
Lists and dictionaries
Loops
Bitwise operators
Classes
File Input/output

# Loops

- Note

```
# to use random numbers
from random import randint
coin = randint(0, 1)
dice = randint(1, 6)
```

# Loops

```
count = 0
while count < 10 : # Add a colon
    print count
    # Increment count
    count +=1

# or
while True :
    print count
    count +=1
    if count >= 10:
        break   #### using break
```

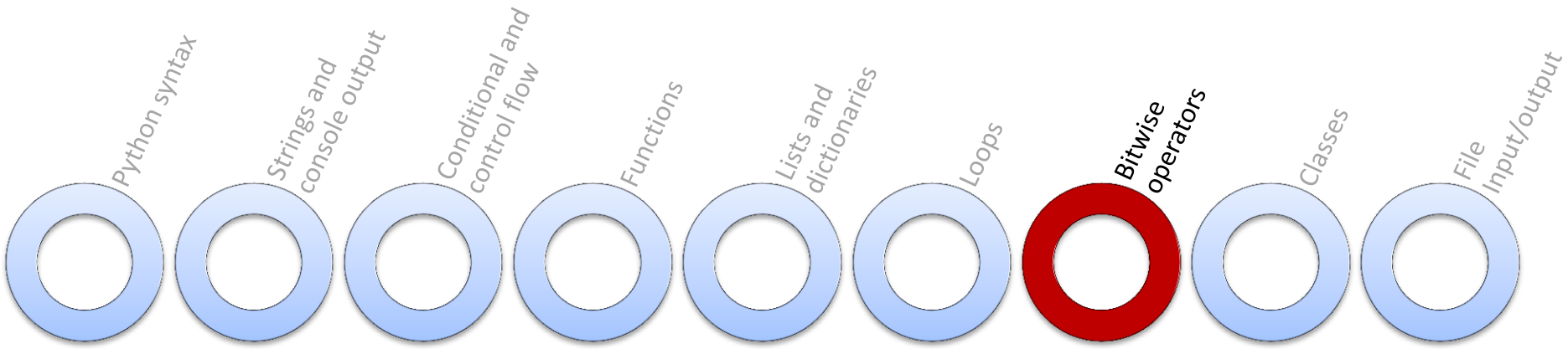
# Loops

```
count = 0
count = 0
while count < 3:
    num = random.randint(1, 6)
    print num
    if num == 5:
        print "Sorry, you lose!"
        break
    count += 1
else:
    print "You win!"    ## will be executed only if didn't enter loop, or loop terminated normally, but it will not be executed if loop terminated by "break" statement, the same like "for/else"
```

# Loops

```
choices = ['pizza', 'pasta', 'salad', 'nachos']
print 'Your choices are:'
for index, item in enumerate(choices):
    print index+1, item    # >> Your choices are: 1 pizza 2 pasta 3 salad 4 nachos
```

# Bitwise operators



# Bitwise operators

```
print 0b11 #  print 3


print  bin(2)   # print bin() returns binary representation of a number (similar oct() hex())
#note you can't use the return as number any more


print int("0b11001001", 2)  # print the integer base ten of the given binary number
```

# Bitwise operators

- shift_right

  0b1001 >> 1  # 0b0100

- shift_left

  0b1001 << 1  # 0b0010

0b1110 & 0b0101  # 0b0100

0b1110 | 0b0101  # 0b1111

0b1110 ^ 0b0101  # 0b1011

print ~1  # print -2

print ~2  # -3

print ~3  # -4

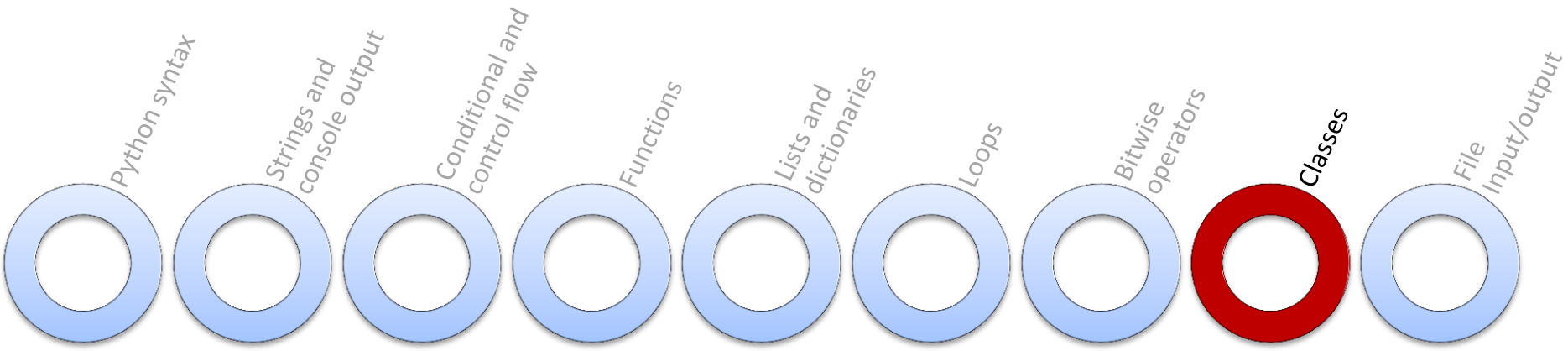
# Bitwise operators

```
# check value of bit 4
def check_bit4(input):
    mask = 0b1000
    result = input & mask  # using mask
    if result > 0:
        return "on"
    else:
        return "off"
```

# Bitwise operators

```
a = 0b10111011
print bin(a|0b100)  # set on bit3
print bin(a^0b11111111) # flip all bits in a
(0b1 << n-1) ^ number  # much simpler to flip bit number n
```

# Classes


Python syntax
Strings and console output
Conditional and control flow
Functions
Lists and dictionaries
Loops
Bitwise operators
Classes
File Input/output

# Classes

- Class: is an object oriented programing concept, class means a collection of some attributes (variables) and methods (functions) that are related to the same physical meaning.
- Defining a class: means defining a new collection of attributes and methods that are related to each other to be used later.
- Using class: is done by making a variable (instance) of that class, one can make multiple instances of the same class and they are all independent.

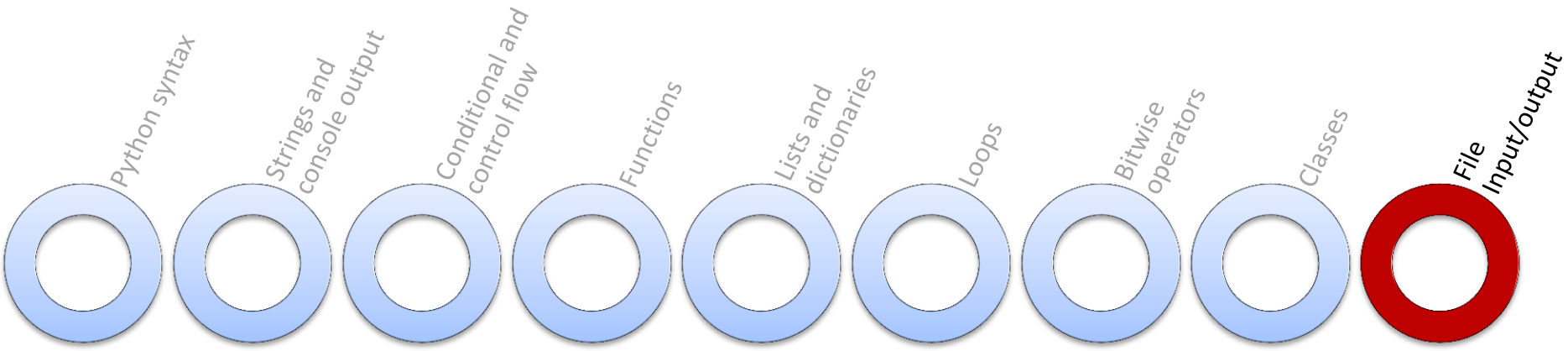
# Classes

```
# define new class
class Animal(object):
    # attributes section definition
  is_alive = True
  health = "good“
    # init method, called automatically each time an instance is made of the class
  def __init__(self, name, age):  #all methods must take (self) as first as this is the variable that points to the current instance
    self.name = name
    self.age = age
  # any methods are added here
  def description(self):
    print self.name
    print self.age
```

# Classes

```
# using class
hippo = Animal("any1", 3)        # make new instance
hippo.description()     # use the method for hippo instance
sloth = Animal("any2", 3)        # make new different instance
ocelot = Animal("any3", 4)       # make new different instance
# they all will print “good” as no one changed the initial value
print hippo.health
print sloth.health
print ocelot.health
```

# File Input/output



# File Input/output

```
# Read from file
my_file = open("output.txt", "r")       # open file with read only permission
print my_file.read()                    # read file content and print it
my_file.close()                         # close file

# Write to file
my_file = open("output.txt", “r+")      # open file with read / write permission
my_file.write(“Hello”)                  # write hello to file
my_file.close()                         # close file
```

# File Input/output

```
# make python close the file by itself after finishing
with open("text.txt", "w") as textfile:
        textfile.write("Success!")
if my_file.closed == False:     # check if file not closed through closed attribute
    my_file.close()
print my_file.closed                # print closed attribute current value, it should now be True as the file is closed
```

# File Input/output

- Different access modes:

r Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.

r+ Opens a file for both reading and writing. The file pointer placed at the beginning of the file.

w Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.

w+ Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.

# File Input/output

- Different access modes:

a Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.

a+ Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.