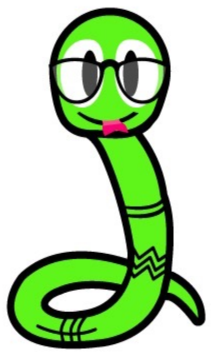
python™



PYTHON 3
Programming Language

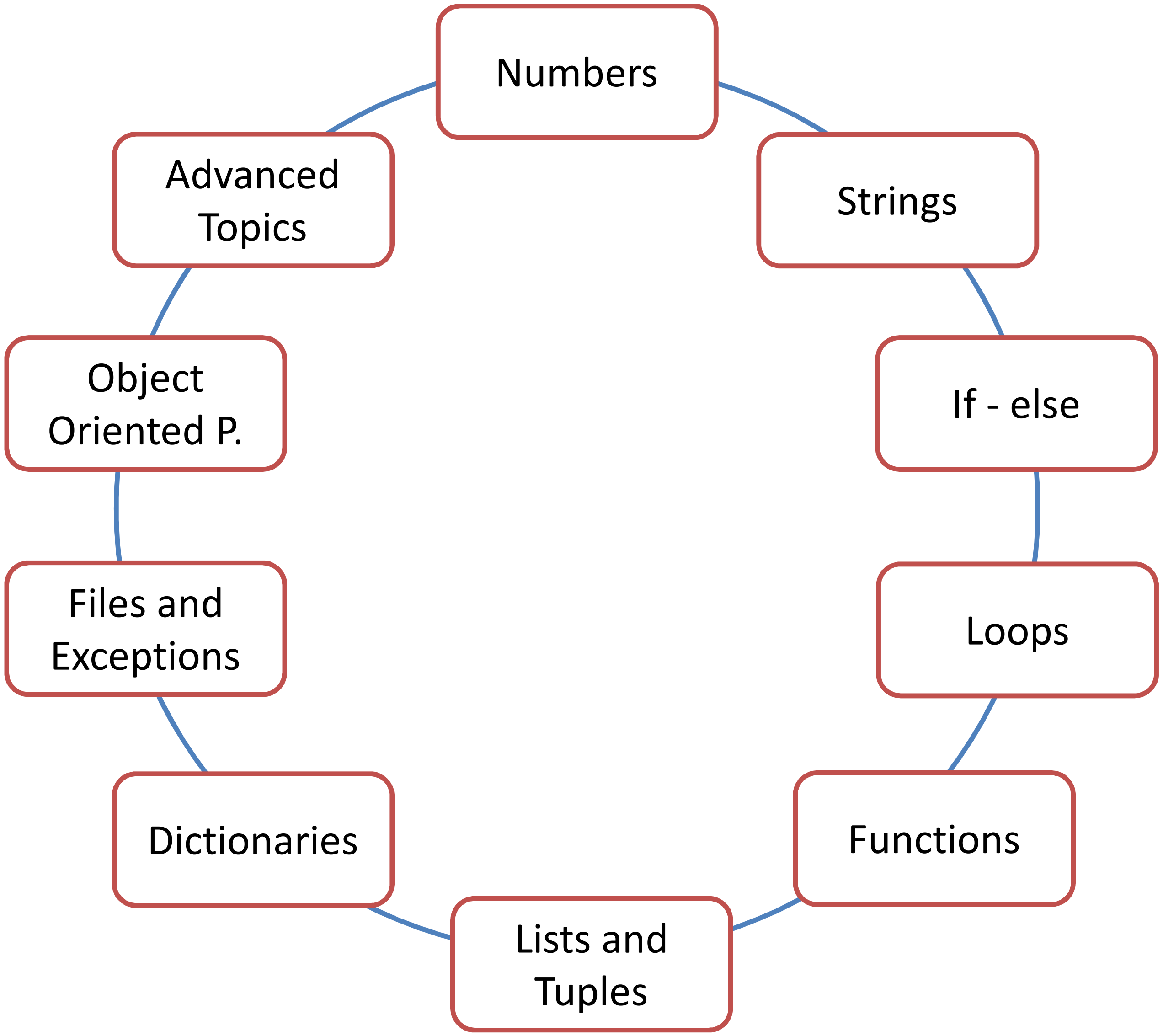
Knowledge Club

KC

Course Content

- Introduction
- Numeric Data Types
- Strings
- Assignments and if
- Loops
- Functions
- Lists and Tuples
- Dictionaries
- Files and Exceptions
- Object Oriented Programming
- Regular Expressions
- Advanced Topics

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

Introduction

- What is Python?
  - General-purpose Programming Language commonly known as *object-oriented scripting language* *Supports both procedural and object-oriented programing*
- Who is the creator?
  - Guido van Rossum – Dutch,1956
    - ✓Python - 1989 GPL Licenced
    - ✓Inspired by Monty Python a comedy show on BBC 1970s

Philosophy

- What is the philosophy? Python Enhancement Proposals (The Zen of Python – PEP20)
  - Beautiful is better than ugly
  - Explicit is better than implicit
  - Simple is better than complex
  - Complex is better than complicated
  - Readability counts
- Python vs. Perl
  - The short story is this: you can do everything in Python that you can in Perl, but you can read your code after you do it

Who Uses Python?

- *Google, Facebook, Instagram, Spotify, Quora, Netflix, Dropbox, Reddit, Pixar*
- *Machine Learning and Cloud Technologies extensively uses Python for projects*
- The popular *YouTube* video sharing service is largely written in Python.
- The *Dropbox* storage service software primarily in Python.
- The *Raspberry Pi* computer promotes Python as its educational language
- The widespread *BitTorrent* peer-to-peer file sharing system starts as Python
- *Industrial Light & Magic*, *Pixar* use Python in the production of animated movies.
- *Intel, Cisco*, *HP*, *Seagate*, *Qualcomm*, and *IBM* use Python for hardware testing
- *JPMorgan Chase*, *UBS*, *Getco*, and *Citadel* apply Python to financial market forecasting.
- *NASA*, *Los Alamos*, *Fermilab*, *JPL* use Python for scientific programming
- The *NSA* uses Python for cryptography and intelligence analysis
- *iRobot* uses Python to develop commercial and military robotic devices

IDE

- Which IDE should I use?
  - IDLE (Default)
  - Pycharm, PyDev, Komodo, Wing
  - Anaconda distribution including Spyder IDE, jupyter-notebook, and other tools and libraries
- Installation of Python on Windows & Linux
  - Download for Windows, already installed for Linux
  - Check version on Linux by $ python -V
  - Install version 3 by yum or zipped source files via: https://www.python.org/downloads/source/

Python Version 2 vs. 3

- Python 3.0 was released in 2008. The final 2.x version 2.7 release came out in mid-2010 and retired 2020.
- Python 3.4 in 2014, 3.5 in 2015, and 3.6 in 2016 December, 3.7 2018 June. Python 3.7.4 July 8, 2019 and 3.6.9 July 2, 2019. Python 3.8.1 Dec. 18, 2019.
- Check versions: https://www.python.org/downloads
- Stable Documentation: https://docs.python.org/3
- What's new? https://docs.python.org/3/whatsnew
- PEPs: https://www.python.org/dev/peps/
- Python Package Index: https://pypi.org/
- Python Relases: https://python-release-cycle.glitch.me/

Install Python and IDE on LINUX

- LAB: Install Python 3 and Pycharm and write Hello World Program and test interactive shell
- Solution:
  - Download Python 3 for Linux (Python 2 already installed on Linux as /usr/bin/python)
    - ✓ https://www.python.org/downloads
    - ✓ Copy to Linux, unzip and follow README to install
  - Download Pycharm Community Edition
    - ✓ https://www.jetbrains.com/pycharm/download
  - Write Hello World with vi and run, also check shell

Install Python and Jupyter on Linux

- First check RHEL already have an older Python
  - # python -V
- Copy new Python zip file under / and extract
- Follow README to install Python
  - Before: yum install zlib-devel and openssl-devel

  ```
  # ./configure;   make;   make test;   make install
  ```
- Install ipython and jupyter using pip

  ```
  # pip list; pip install --upgrade pip
  # pip install ipython
  # pip install jupyter
  ```

Configure Python

- Which shell parameter is responsible to access python?
- Add /usr/local/bin to PATH
- Practice: Configure your machine so that when you run python it calls 2.7 and when you run python3 it calls 3.6

  (Create a symbolic link named python3)
  - # env
  - # echo $PATH
  - # PATH=$PATH:/usr/local/bin
  - Edit system-wide or user-specific initialization file to make it persistent

First Python Program

- Edit hello.py

  ```
  #!/usr/local/bin/python3
  print('Hello World')
  ```

  - $ chmod +x hello.py
  - $ ./hello.py (One way to do it)
  - $ python hello.py (Another way, what is the difference?)
  - $ python (Interactive Shell, use exit() to quit)
  - If you installed another version of python change first line e.g. #!/usr/local/bin/python3

Types of Quotes

- How do you add comments?
  - Just use # as usual anywhere
- How about quotes? (Remember escape char is \)
  - Single and Double Quotes have same effect
    - ✓'This is a string'
    - ✓“This is a string”
    - ✓Raw input: r'C:\some\name'
  - Triple Quotes: Strings containing more than one line
    - ✓“””This is a
      string”””
- LAB - Edit hello2.py that reads username from keyboard and say 'Hello <NAME>. Wellcome to Python Class!'
  - Use function input
  - name = input('Enter your name ')
  - print('Hello',name,'Welcome to Python Class!')

How Python Works?

- You only need to worry about your source code .py
- This text code is compiled into a byte-code which is machine-independent and runs on PVM (Python Virtual Machine) which is always present after installation, default CPython
- If you really want to see this byte-code use this
  - $ pyhton3 -m py_compile hello2.py
  - This will create __pycache__ folder and .pyc file
- This resembles Java codes and JVM. Actually you can even create Python bytecode to be run on JVM using Jython. Default is CPython.
- If you want to make Python even faster use PyPy which is a JIT(Just in time compiler) compiles bytecode to machine code realtime. Another alternative is Cython to speed up.

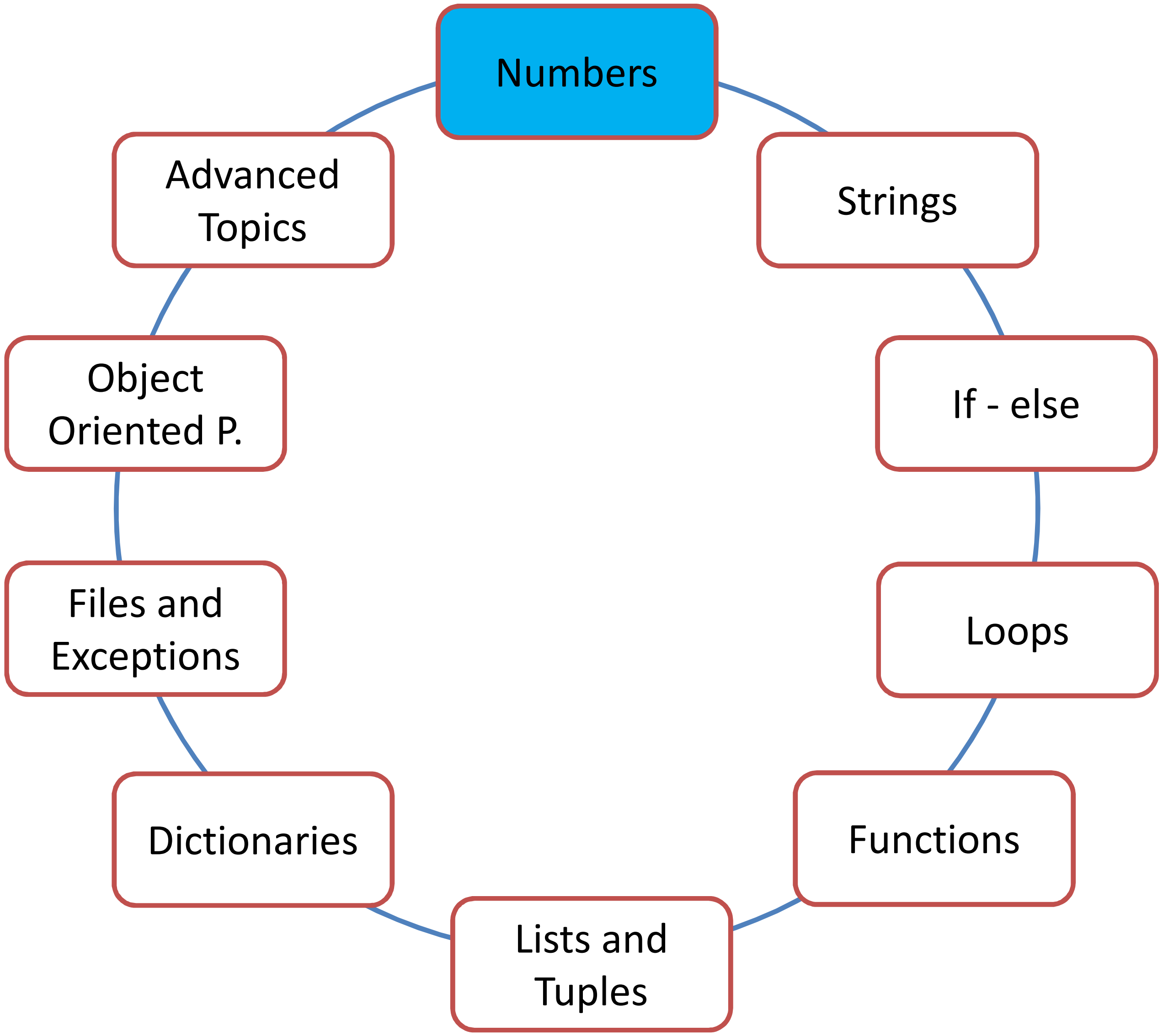
Naming Conventions

- A Python identifier is a name used to identify a variable, function, class, module, or other object
- An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits
- Naming convention for Python
  - Class names start with an uppercase letter and all other identifiers with a lowercase letter.
  - Starting an identifier with a single leading underscore indicates by convention that the identifier is meant to be private.
  - Starting an identifier with two leading underscores indicates a strongly private identifier.
  - If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.
- Google Python Style Guide has the following convention
  - package_name, module_name, ClassName, method_name, ExceptionName
  - function_name, GLOBAL_CONSTANT_NAME, global_var_name, instance_var_name, function_parameter_name, local_var_name

Data Representation - OOP

- Everything is object and no explicit type or variable declaration needed, all dynamic
- Following steps reflect the operation of all assignments (a=42)
  - Create an object to represent the value 42.
  - Create the variable a. Link the variable a to the new object 42.
  - Variables are entries in a system table, with spaces for links to objects.
  - Objects are pieces of allocated memory, with enough space to represent the values for which they stand.
  - References are automatically followed pointers from variables to objects
- Automatic Memory Management (Garbage Collector)
  - del <object> (Explicit Deletion)
- Here are the links for built-in functions, data types and modules
  - https://docs.python.org/3/library/functions.html
  - https://docs.python.org/3/library/stdtypes.html#
  - https://docs.python.org/3/py-modindex.html

Work Flow

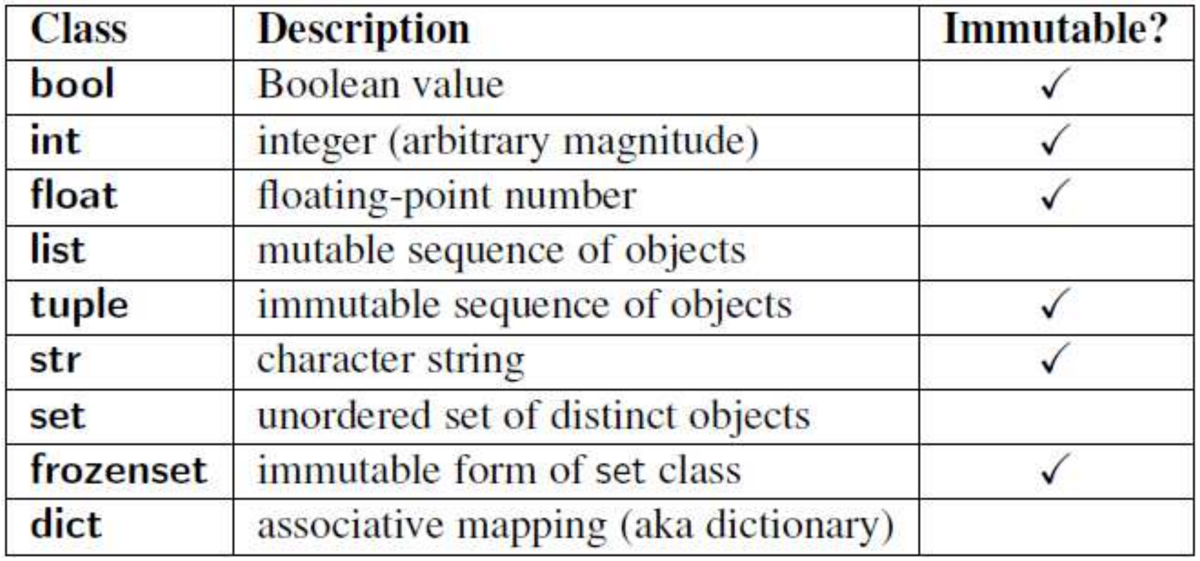


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

2. Numbers

- The principal built-in types are numeric, sequences, mappings, classes, instances and exceptions
- Python has five standard data types
  - Numbers
  - String
  - List
  - Tuple
  - Dictionary
- Built-in Constants
  - True, False, **None**
- Type function

```
>>> type(3)
<class 'int'>
>>> type('hello')
<class 'str'>
```



Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

Type Casting

Action	Converting what to what
int()	string, floating point → integer
float()	string, integer → floating point number
str()	integer, float, list, tuple, dictionary → string
list()	string, tuple, dictionary → list
tuple()	string, list → tuple

Basic Operators

- Basic Operators
  - +,-,/,*,%   and   +=,*=,/=
  - Exponentiation: **
  - Floor division: // (Floor the value 3/2->1)
    - ✓Note that standard division always result float! 4/2=2.0
- Bitwise Operators
  - >>,<<,|(or),&(and),^(xor),~(not), bin(), oct(), hex()
- Logical Operators
  - and, or, not
- Comparison
  - >,<,>=,<=,==,!= (Like C)
- Others: in and not in → Sequence, is and is not → Object

Numeric Data Types

- Numeric Data Types
  - int (Unlimited! Try >>>2**1000)
  - float (Default Precision: 17)
  - bool → True or False
  - None type
  - complex → 2+3j
  - If you want to get high precision with floating point operation, consider using mpmath module.
    - ✓Practice: Calculate (1 + 1/n) ^ n with high precision

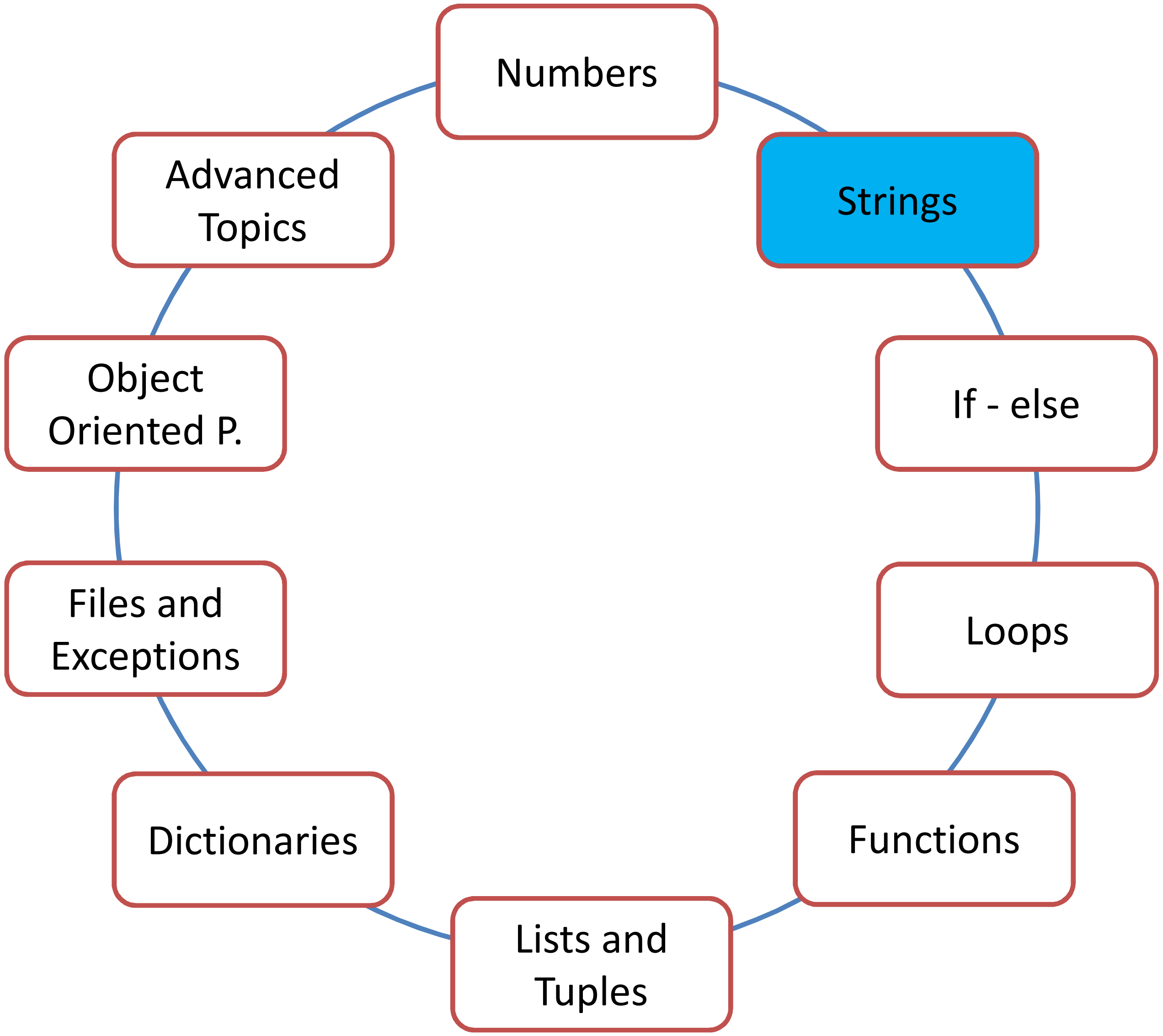
Numeric Functions

- Standard Functions
  - int(x) : Converts to integer (Truncates)
  - str(), float(), list(), tuple(), set(), dict()
  - round(x, 2) : Converts to float with precision 2
  - min(), max(), sum(), len(), del(), id(), eval()
- import math
  - math.e, math.pi
  - math.exp(x), math.log(x), math.sqrt(x)
- import time
  - time.sleep(0.5), time.ctime()
- import random
  - random.random()
  - random.randint(1, 6) -> Dice

LABS for Numbers

- LAB1: Write a Python Program that converts Fahrenheit to Celcius. (Ask Fahrenheit value from keyboard)
  - Use function -> input('Please enter Fahrenheit value: ')
  - Casting needed? => int(),float(),str()
    - ✓What is the return type of input() function?
- LAB2: Pisagor Triangle: Ask for sides of right triangle a and b and calculate Hypotenuse c. ($a^2 + b^2 = c^2$)
  - How can you print less numbers? Lower precision.
    - ✓Print functionality (See Appendix)
    - ✓Round function

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

3. Strings

- Strictly speaking, strings are sequences of one-character strings. Other sequence types include *lists* and *tuples*
- Practice

```
>>> S='Spam'
>>> len(S) -> 4 (Built-in)
>>> S[0] (First element)
>>> S[-1] or S[len(S)-1] (Last element)
>>> S[1:3] -> pa (Slice start at position 1 until 3)
>>> S[1:] -> pam (Start at 1st until end)
>>> S[-3:] -> pam (Last 3 chars)
>>> S+'xyz' -> Spamxyz (Concat)
>>> S * 3 -> SpamSpamSpam (Repetition)
>>> S=r'\temp\spam' (raw input no escape chars)
```

Strings are Immutable

- Strings are Immutable Objects

  ```
  >>> S -> Spam
  >>> S[0]='z' (Error! You can not change 1st element of S)
  >>> S = 'z' + S[1:] (This is ok since new object created!)
  >>> L = list(S) (Convert string to list, now you can change)
  >>> L[0]='z' (['z', 'p', 'a', 'm'])
  >>> S = ''.join(L)
  ```

- Getting a list about string methods

  ```
  >>> dir(S) (Assume S is a string. Ignore __ ones)
  ```

Strings

- Practice
  - Printing string chars

```
>>> myjob = 'hacker'
>>> for c in myjob: print(c, end='  ') (No newline, space instead)
h a c k e r
```

  - Extended Slicing

```
>>> S = 'abcdefghijklmnopqrstuvwxyz'
>>> S[1:10:2]   (Start at position 1 go till 10 skip 2)
bdfhj
>>> S[::2] (Start beginning skip 2 till end)
acegikmoqsuwy
>>> S[::-1] (Reverse!)
zyxwvutsrqponmlkjihgfedcba
```

String Methods

- Find and Replace Methods

  ```
  >>> S = 'I like tea. I drink at least five cups of tea daily'
  >>> S.find('cups') -> 34 (Searches pa and finds at position 1)
  >>> S.replace('tea','coffee')
  >>> S (Immutable! Use S=S.replace('tea','coffee')
  ```
- upper(),lower(),isalpha(),isdigit(),endswith(),startswith()

  ```
  >>> S.upper() -> SPAM
  >>> S.lower() -> spam
  >>> S.isalpha() ->  True
  >>> S.isdigit() -> False
  >>> S.startswith('SP') -> True
  >>> S.endswith('M') -> True
  ```

String Methods

- Split and Strip

```
>>> line = 'apple,orange,banana  \n'
>>> line.rstrip()
'apple,orange,banana'  # Removes right whitespaces
>>> new = line.rstrip().split(',')
['apple','orange','banana'] # First remove then split with ,
```

- ord and chr functions

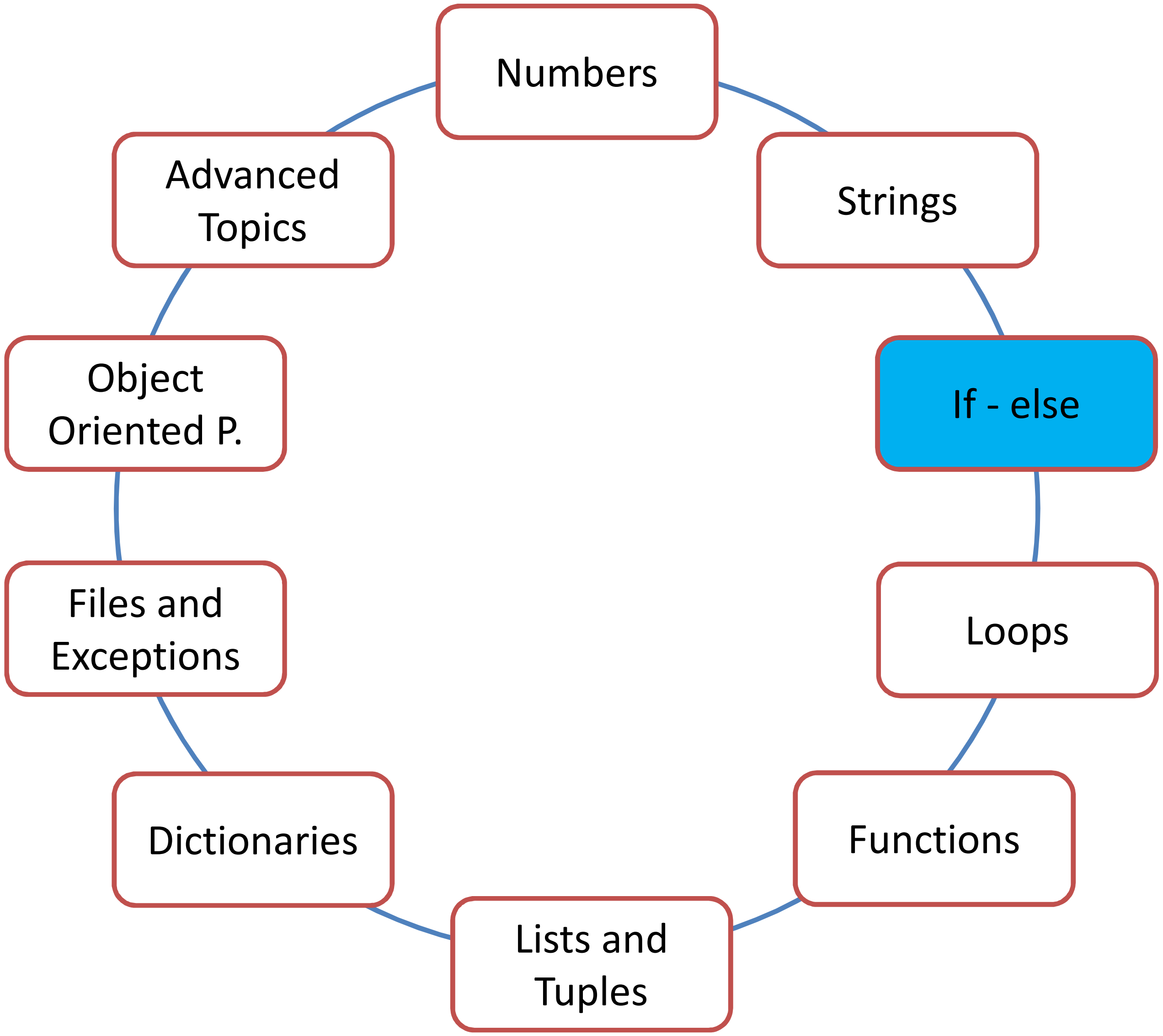
```
>>> ord('\n') -> 10
>>> chr(65) -> 'A'
```

  - Q: Print ASCII Table using for & chr

LAB – Generate Login Name

- Read users first_name, last_name, and number_id and generate a Login Name with the following rule: Take first 3 chars from first_name + take first 3 chars from last_name + take first number from id. Finally print a line like 'Hello <first_name>. Your account is created with <login_name> username and password. Please login and change your password immediately.
- Bonus question: How can you check first & last name contains only chars and id contains numbers?
- Bonus Lab: Take first 3 chars from first_name + take last 3 chars from last_name + take one random number from id

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

4. if

- Conditions: Booelan: True/False
- Logical Operators: And/Or/Not. Comparisons: <,>,>=,<=,==,!=
- Syntax: if, if/else, nested if

```
if condition:
    print('Ok')
 else:
    print('Not OK')
```

- What about more than one condition?

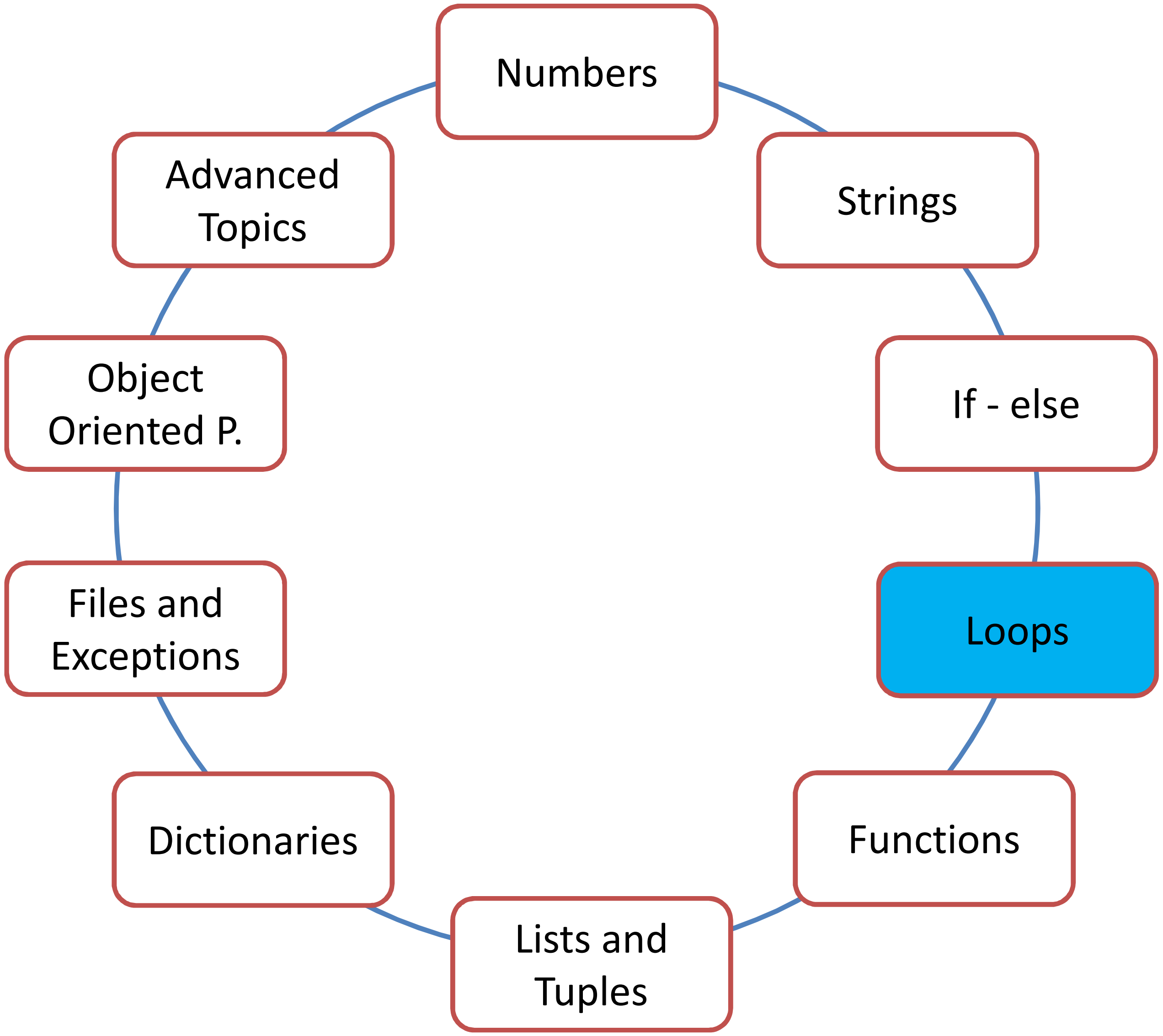
```
if condition1:
    print('first')
elif condition2:
    print('second')
else:
    print('last')
```

LAB – Grades

- Write a Python Program that read users exam result 0-100 and prints it in terms of A,B,C,D, or F
  - Grade A: 90-100, Grade B: 80-89
  - Grade C: 70-79, Grade D: 60-69
  - Grade F: 0-59   (Fail)
- Write a Python Program that reads two numbers a, b. If a/b < 0.5 or b = 0 print Values Not Acceptable.
- Bonus Lab: Read a year from keyboard and print if that year February has 28 or 29 days
- Python Info: Is there a case/switch statement in Python?

  https://www.python.org/dev/peps/pep-3103/

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

5. Loops

- While Loop

  ```
  while condition:
      statement1
      statement2
  else:  (Not needed)
  ```

- LAB - Counter
  - Read an end number from the keyboard and print numbers from 1 to end (1,2,3,4, … , end)
  - Bonus: Add half a second intervals (Performance issue)
  - Bonus Lab: Create a timer that counts down to zero on the same spot. Add bell sound for the last 3 seconds!

For Loop

- For Syntax: for var in <list>:
  >>> for fruit in ['apple','orange','banana']: print(fruit)
  >>> for var in [1,2,3,4,5]: print(var)
  >>> S = 'Lumberjack' ; T=('Once','upon','a','time')
  >>> for x in S: print(x,end=' ') ---> L u m b e r j a c k
  >>> for x in T: print(x,end=' ') ---> Once upon a time
- **range()** function (actually immutable sequence) is very useful to create a number list
  - range(5) ---> 0,1,2,3,4
  - range(2,6) ---> 2,3,4,5
  - range(1,9,2) ---> 1,3,5,7
  - range(10,2,-2) ---> 10,8,6,4
- For both loops, you can use break and continue statements
  - Practice: print 1..10 first. Then break at 5, or skip on 5.

5. Loop Labs

- LAB: Homogenous String. Read a string and check all chars are the same or not
- LAB: Palindrome. Read a string and check if is it is a palindrome
- LAB: Speed Test. Calculate the time to sum one millions numbers in Python
- LAB: 3n+1 Problem. Play the game including the iteration counter. Example: Initial Number is 5. (Game ends when you hit 1). Try 97!

  Example: 3 => 10 => 5 => 16 => 8 => 4 => 2 => 1 (Game Over!)
- LAB: Read a password and check if it contains at least 1 upper, 1 lower, 1 digit and length is at least 8 to accept

5. Loop Labs

- BONUS LAB: Which number has the highest steps in 3n+1 game between 1..100?
- BONUS LAB: Calculate and print transcendental pi and e numbers with online view. Show the real values with math.pi, math.e

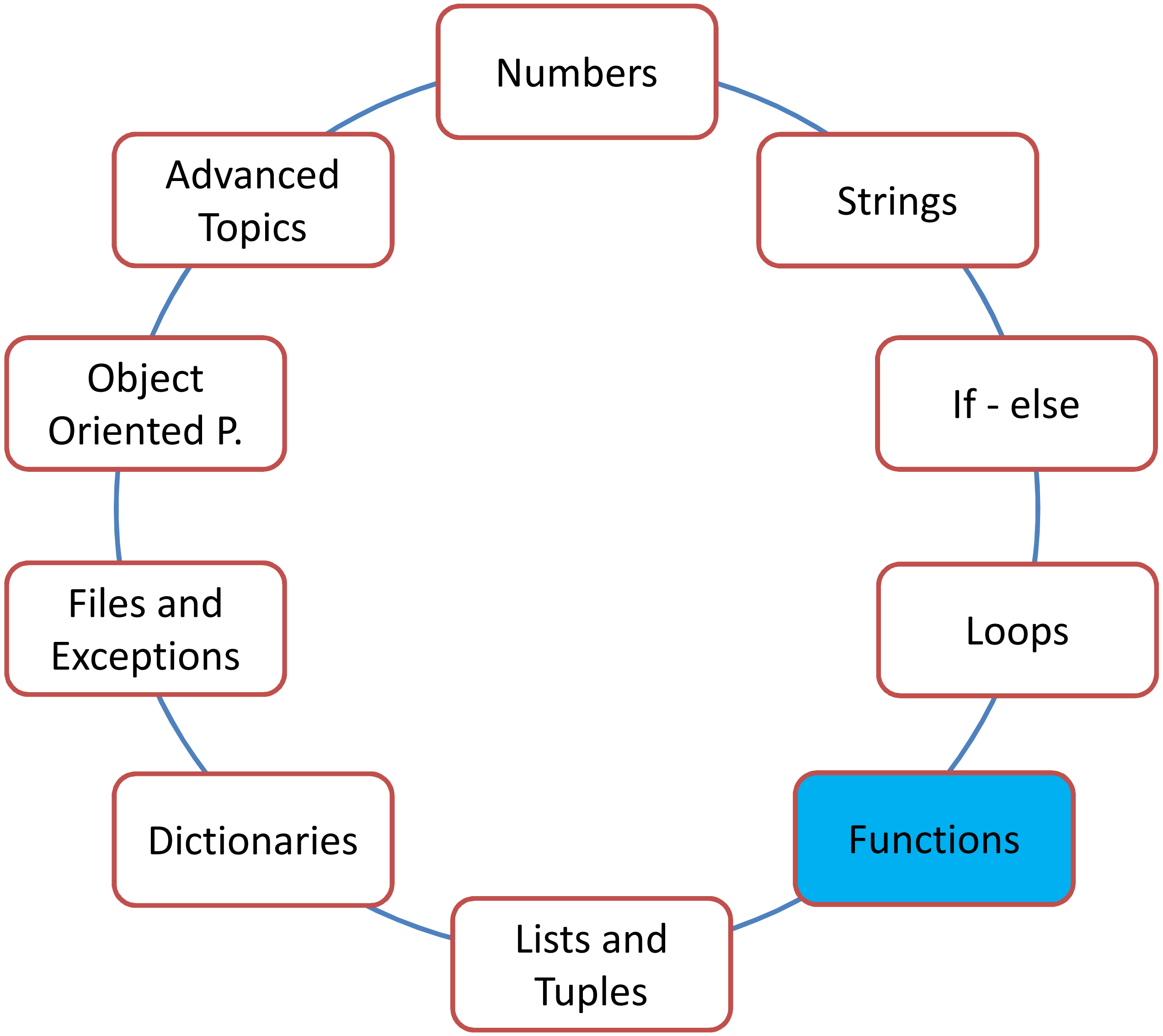
  Use the following formulas:

  e = 1 + 1/1 + 1/(1x2) + 1/(1x2x3) + 1/(1x2x3x4) + …

  π = 4 x (1/1 – 1/3 + 1/5 – 1/7 + 1/9 – 1/11 +- …) => Gregory-Leibniz Series (Slow)

  π = 3 + 4/(2*3*4) - 4/(4*5*6) + 4/(6*7*8) - 4/(8*9*10) + 4/(10*11*12) - 4/(12*13*14) +- …) => Nilakantha (Faster Convergence!)

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

6. Functions

- Defining a function

  def my_function(n): # n is passing parameter

  statements

  return result

- Calling a function => **my_function(n)**
- Scope: Local vs. Global parameters
  - All local within function unless global keyword used

  var = 88 # Global var since in the main

  def func():

  global var

  var = 99 # Global var: in effect outside def

  func()

  print(var) # Prints 99. Without global it should be 88

Scope

- Pass by value or Pass by reference?

  ```
  def change(n):  n=n+1
  val = 1
  print('Initial value is',val)
  change(val)
  print('New value is',val)      # You Still get 1
  ```

  (New reference created when function called to point same object. However when you change n, another object created, because of immutability)
- How can you change passing parameter? (Use return value)
- Avoid using global parameters as much as possible, create a main function to navigate your program
- If you want to create a global constant parameter to share among functions, just create a parameter outside functions

Function Return Values

- How to change passing values in functions?

```
def increment(n):
    n = n + 1
number = 42
increment(number)
print(number)
```

- How can you get back multiple values from function?

```
def calc(a, b):
    c = a + b;  d = a * b
    return c, d
total,mul = calc(7, 5)
```

- LAB: Write a factorial function

LAB - Fibonacci

- LAB: Write a Fibonacci Function and call from main script

  Read the parameter n from user and pass it to function Fibonacci(n). Try Fibonacci(1000)

  Bonus: Print the ratio to show phi approximation and Calculate and show phi mathematically: (1+sqrt(5))/2
- BONUS LAB: How to write a function that accept variable number of inputs? Write a function called sigma that adds any number of input float values

Recursion

- Recursive function is a function that calls itself
  - Very useful to solve some problems
    - ✓Binary Search, Quick Sort, Towers of Hanoi
  - Should be careful about termination rule
    - ✓Infinite recursion -> Stack Overflow!
- LAB: Write a recursive function to calculate n!
  - n! = n x (n-1)!
  - Termination rule: return 1 when you reach number 1
- BONUS LAB: Write Fibonacci Recursively

Random Number Functions

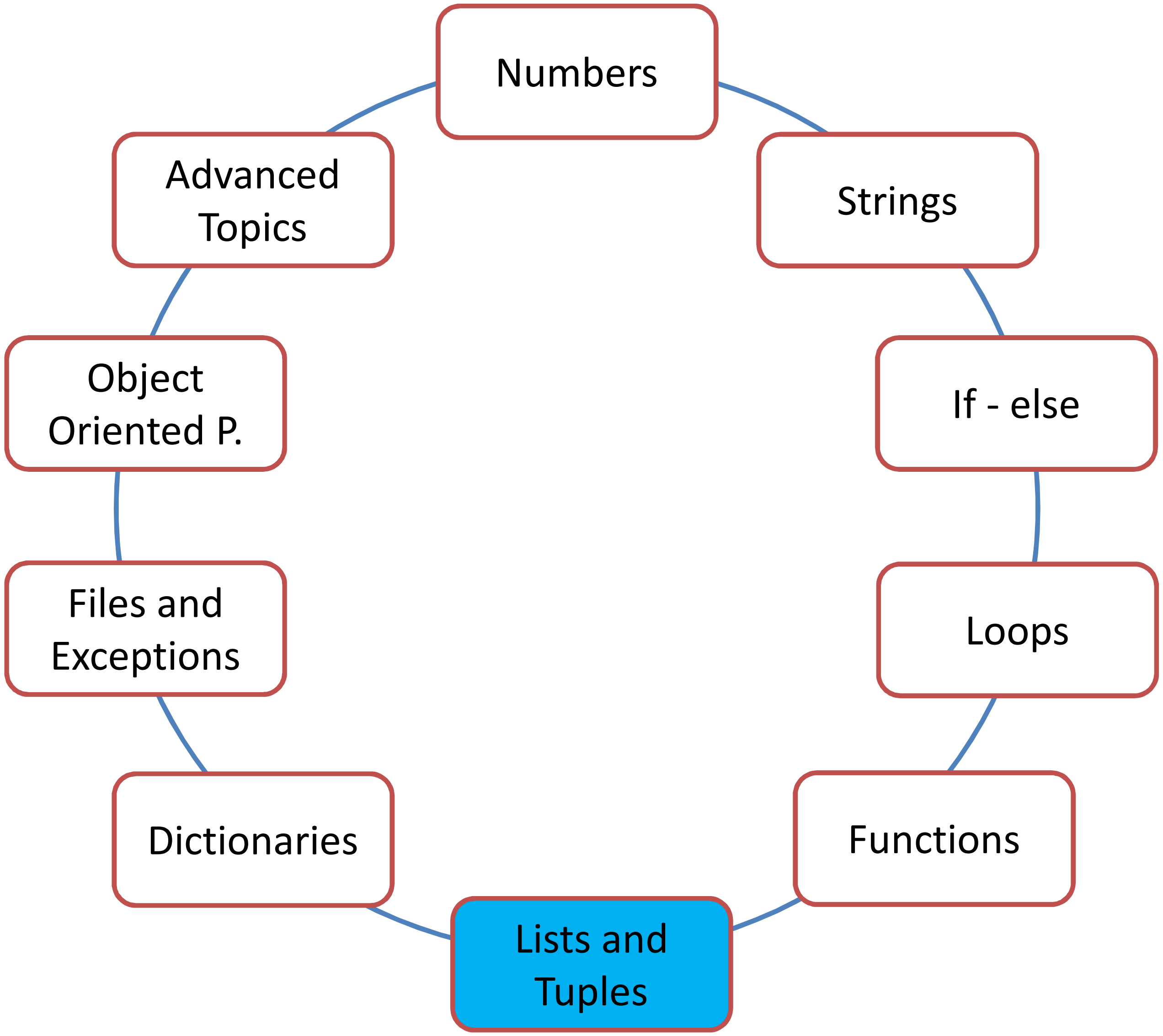
- Random module contains pseudo random generating functions
  - random.random() : Generates [0,1) float
  - random.choice([1,2,3,4,5,6]) : Choose one of these
  - random.choice(['apple','orange','banana'])
  - Suit = ['spades','hearts','clubs','diamonds']
    - ✓ random.shuffle(Suit) ---> Suit is a list(Mutable)
  - random.randint(1,6) : Generate 1,2,3,4,5,6
  - random.randrange(0, 51, 10): Generates 10,20,30,40,50
  - random.seed(5) : Use number 5 initialize pseudo random number generator. Default is system time. Use this to get exact same numbers. (Pseudo Random – Uses Formula)

LABS - Coin and Dice

- LAB: Play the number guessing game between 1..100
- LAB: Read the number from keyboard and toss the coin and show online head and tails with one second intervals. Finally state the result number of heads and tails
- LAB: Throw 2 dice and display output 5 3, 3 2, 1 5 etc.

  If you throw same number like 5 5 then print lucky. If you throw 6 6 then say winner and quit. Print counter and delay one second between throws
- LAB: Write a is_prime() function which test if given number is prime or not. (Check till sqrt(x))
  - Bonus: Use is_prime and print first 100 prime numbers
  - Bonus: Read a number and show prime factors. 6=2x3

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

7. Lists and Tuples

- Lists and Tuples are basic sequence types.
  - Also Range is immutable sequence
- Main difference: list is mutable and tuple is immutable
- L = ['Alice','Bob',28,3.14] (List may contain different types. This may be inefficient for large data => numpy)
- L = [0,5,10,15,20,25]
- L = list(range(0,30,5)) => Use list function to create list
- L * 5 ---> Repeats the elements of list 5 times!
- Process the list

```
for i in L:
    print(i)
```

Lists

- INDEXING. 0 to len(L)-1. Process with while

```
index=0
while index<len(L):
    print('Position',index,'Value',L[index])
    index+=1
```

- SEARCHING with in

```
if 'Monday' in days:
    print('Yes Monday is in the list')
```

Lists

- CONCAT

  list1 = [1,2,3,4,5]; list2 = [6,7,8,9,10]

  list3 = list1 + list2

  list1 += list2
- SLICING

  days = ['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday']

  weekdays = days[1:6]

  weekend = days[0]+days[-1]

  list1 = days[:2] ---> ['Sunday','Monday']

  list2 = days[4:] ---> ['Thursday','Friday','Saturday']

  a,b,*c = days

List Methods

- **append(item)** Adds item to the end of the list
- **insert(index, item)** Inserts item into the list at the specified index. The item that was previously at the specified index, and all the items after it, are shifted by one position toward the end of the list
- **pop()** Removes last element, or give index
- **remove(item**) Removes the first occurrence of item from the list. A ValueError exception is raised if item is not found in the list
- **index(item)** Returns the index of the first element whose value is equal to item. A ValueError exception is raised if item is not found in the list
- **sort()** Sorts the items in the list so they appear in ascending order (from the lowest value to the highest value).
- **reverse()** Reverses the order of the items in the list.

List Methods

- Practice

```
>>> L = list(range(1,10))
>>> L.append(6)
>>> L.index(2)
>>> L.sort()
>>> L.reverse()
>>> L.insert(3,7)        => Insert 7 at index 3
>>> L.pop()              => Remove last item of list
>>> L.insert(0,13)       => Insert 13 at the beginning
>>> L.remove(9)          => Remove element 9 from the list
```

Built-in Functions

- **del(List[1])**
  - L.remove(<item>) removes an item from the list

```
>>> L = [1,2,3,4,5,6,7,8,9]
>>> del(L[1])
>>> L.remove(4)
>>> L.pop(5)
```

- **min(L), max(L), sum(L)**
  - Finds the min, max, and sum of a list

```
>>> min(L)
>>> max(L)
>>> sum(L)
```

List Copy

- How to copy a list?

  >>> L = [1,2,3,4,5]

  >>> A = L (This one creates another reference to the same object. So no real copy here)

  >>> A[0] = 7

  >>> A; L   => Both shows [7,2,3,4,5]

  >>> B = [] => Now new object is created for B

  ```
  >>> for item in L:
                  B.append(item)
  ```

  >>> C = [] + L   => This is nice and easy way to copy

  >>> C = L.copy() => Same as above

Multi-dimensional Lists

- Create a matrix with 3 rows and 4 columns

```
>>> M = [[1,2,3,4],[2,4,6,8],[3,6,9,12]]
>>> M        # Prints whole matrix
>>> for i in range(len(M)):     # Processing Matrix
>>>          for j in range(len(M[0])):
>>>                   print(M[i][j])
```

- Create a 3x5 multi-dimensional list within Python script

```
Table = [ [ 1 for i in range(5) ] for j in range(3) ]
print(Table)
for d1 in range(3):
     for d2 in range(5):
          Table[d1][d2]= d1+d2
```

- Complex list, each item is another list: ['name','address','phone']
- Difference between sort() vs sorted() and reverse vs reversed()

Tuples

- A *tuple* is a sequence, like a list primary difference:
  - Tuples are immutable syntax ()
  - Lists are mutable syntax []
- Tuples support all the same operations as lists, except those that change the contents
  - Methods such as index, count
  - Built-in functions such as len, min, and max
  - Slicing expressions
  - The in operator, The + and * operators
- Tuples do not support methods such as append, remove, insert, reverse, and sort.

Tuples

- What's the point?
  - One reason that tuples exist is performance
  - Another reason is that tuples are safe (immutable)
- How can you convert between Tuples and Lists
  - Use list() and tuple() functions

```
>>> T = (1,2,3,4,5)
>>> T[0] = 6        # You can not change a tuple
>>> L = list(T)
>>> L[0] = 6        # Now you can with a list
>>> T ; L
>>> new = tuple(L)
```

LABS - Lists

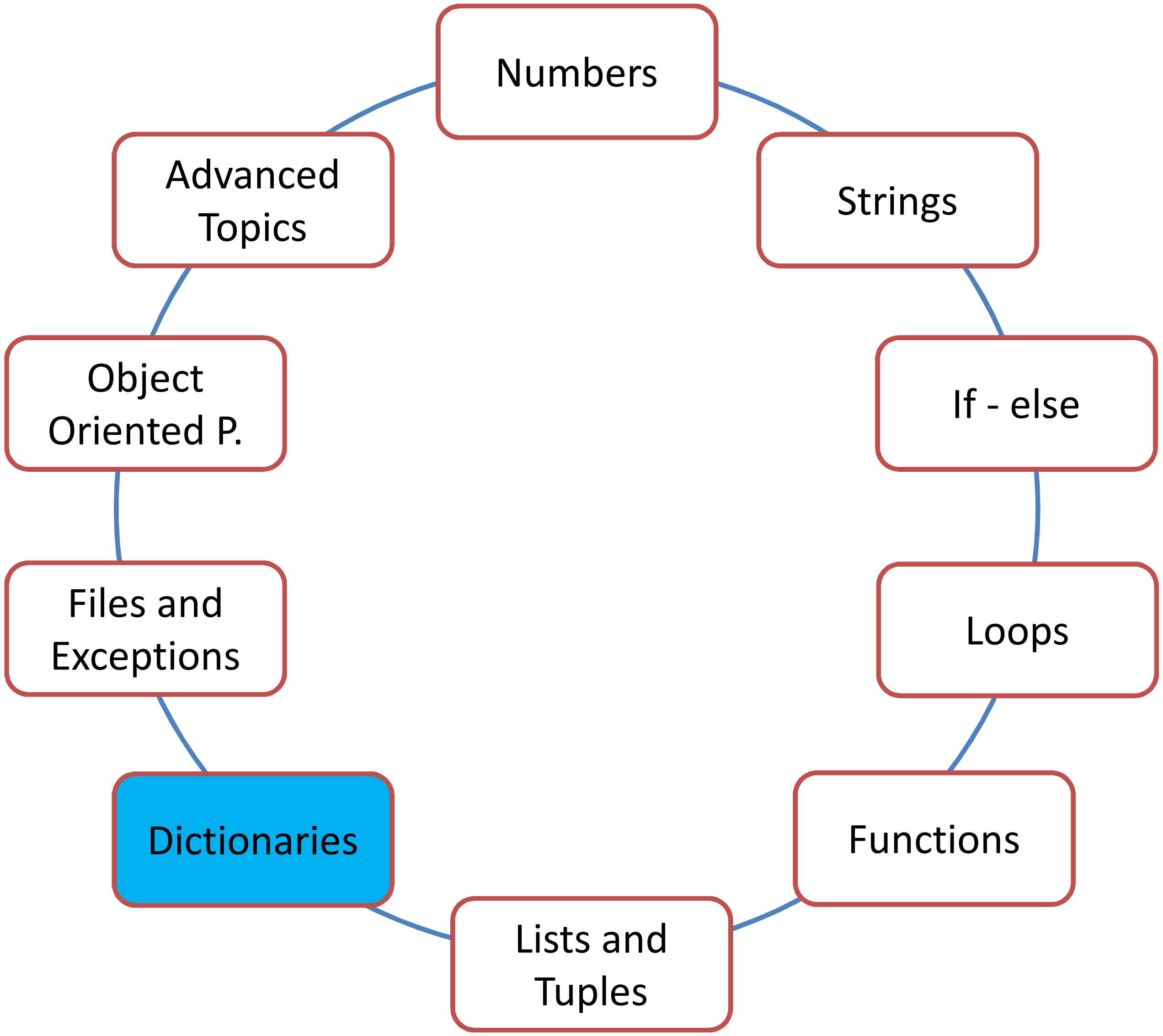
- Play Lottery Game: Generate 6 numbers 1..49, sort. Use set?
- Find the first 1000 prime numbers, use a list to store primes. Efficiency test: Use sqrt and calculate the time difference.
- Create a playing deck using a list then ask user to pick a card. Shuffle the deck and start showing the cards until you reach the card.

  Deck = ['1 of Hearts','2 of Hearts','3 of Hearts','4 of Hearts','5 of Hearts',
  '6 of Hearts','7 of Hearts','8 of Hearts','9 of Hearts','10 of Hearts',
  'Jack of Hearts','Queen of Hearts','King of Hearts',
  '1 of Diamonds','2 of Diamonds','3 of Diamonds','4 of Diamonds','5 of Diamonds',
  '6 of Diamonds','7 of Diamonds','8 of Diamonds','9 of Diamonds','10 of Diamonds',
  'Jack of Diamonds','Queen of Diamonds','King of Diamonds',
  '1 of Spades','2 of Spades','3 of Spades','4 of Spades','5 of Spades',
  '6 of Spades','7 of Spades','8 of Spades','9 of Spades','10 of Spades',
  'Jack of Spades','Queen of Spades','King of Spades',
  '1 of Clubs','2 of Clubs','3 of Clubs','4 of Clubs','5 of Clubs',
  '6 of Clubs','7 of Clubs','8 of Clubs','9 of Clubs','10 of Clubs',
  'Jack of Clubs','Queen of Clubs','King of Clubs']

BONUS LAB - Complex Lists

- Create a list, reading from the keyboard in the following manner:
  - First ask name and read First and Last name as string
  - Second ask address and read whole address as a single string
  - Third ask phone numbers and read as many phones as user supplies. User should enter all phones in one line with comma separated numbers
  - Quit entering values when user give name as **quit**
- Print only names and primary phone number

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

8. Dictionaries and String Extras

- String operations. S = 'Spam'
  - Iteration: for var in S:
  - Indexing: S[0],…,S[-1] ---> S[-1] = S[len(S)-1]
  - Slicing: S[:-1] # Just remove last char
  - Concat: S = S+'hello' (Immutable, new string created)
  - Check: if 'pam' in S: ---> Check if 'pam' is in the S
  - Repetition: S = S * 3 ---> 'SpamSpamSpam'
  - Split: L = S.split(':') ---> Split with : separator. L is a list. Default separator is whitespace.
  - Join: S = ' '.join(L) ---> Join a list into String with ' '
  - Unicode string: S = u'Spam'

String Functions

- S.find('pa') -> 1 (Searches pa and finds at position 1)
- S.replace('pa','XYZ') -> SXYZm (S is not changed)
- S.upper() -> SPAM
- S.lower() -> spam
- S.isalpha() -> True
- S.isalnum() -> True
- S.isdigit() -> False
- S.isdecimal() -> False
- S.isspace() -> False
- S.startswith('Sp') -> True
- S.endwith('m') -> True
- S.rstrip(), S.lstrip(), S.strip()

LABS - Strings

- LAB: Arcan's Word Game. Ask user's first name and start a game randomly generating all chars simultaneously with the length of name. Stop when you reach the original name. Show counter.
  - Math Bonus: What is the probability of reaching arcan at the first try?
  - Math Bonus: How many steps needed to reach word arcan with the probability of %99?
- LAB: Same problem like above but generate random chars one by one. Show them sequentially, like slot machine.

Dictionaries

- Dictionaries are basically key,value pairs and mutable
  - Keys of dictionaries must be immutable types but values can be any type: lists,tuples,strings, etc.
  - Key,Value pairs not stored in particular order
  - Keys are unique. You can not have same key twice!
  - Values can be sequences like tuples!

```
>>> phonebook =
{'Gandalf':'05321234567','Saruman':'05421234567','Saur
on':'05051234567'}
>>> phonebook  ---> Output may not in the same order
>>> phonebook['Gandalf']
```

Dictionaries

- How do you check if Gandalf exist before print?

  `>>> if 'Gandalf' in phonebook: print(phonebook['Gandalf'])`
- How to add an element? (Dictionaries are mutable)

  `>>> phonebook['Frodo'] = '05551234567'`
- How to delete an element?

  `>>> del(phonebook['Saruman'])`
- Size of the phonebook?

  `>>> len(phonebook)`
- How to create an empty dictionary?

  `>>> phonebook={}`
- Iteration

  ```
  >>> for key in phonebook:
  >>>         print(key,'--->',phonebook[key])
  ```

Dictionary Functions

- **clear()** Clears the contents of a dictionary

  ```
  >>> phonebook.clear()  --->  Empty {}
  ```

- **get()** Gets the value associated with a specified key. If the key is not found, the method does not raise an exception. Instead, it returns a default value

  ```
  >>> number = phonebook.get('Saruman','Not here!')
  ```

- **items()** Returns all the keys in a dictionary and their associated values as a sequence of tuples.

  ```
  >>> for key, value in phonebook.items():
                  print(key, value)
  ```

Dictionary Functions

- **keys()** Returns all the keys in a dictionary as a sequence of tuples

```
>>> for key in phonebook.keys():
        print(key,' ---> ',phonebook[key])
```

- **pop()** Returns the value associated with a specified key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.

```
>>> number = phonebook.pop('Saruman','Not found!')
```

- **popitem()** Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.

```
>>> name,number = phonebook.popitem()
```

- **values()** Returns all the values in the dictionary as a sequence of tuples.

```
>>> for val in phonebook.values():
        print(val)  --->  Only numbers
```

LAB – Exam Results

- Read math exam results from keyboard (or from a file) and create a dictionary as keys:names and values:scores.

  Stephen Hawking:95

  Gary Kasparov:67

  Albert Einstein:78

  Paul Erdos:100

  Pierre De Fermat:68

  Cahit Arf:100

  Paul Dirac:93

  Alan Turing:95
- Write a Python program that prints following values:
  - Total number of students, average, min and max scores
  - You need to print all names if more than one person gets the min or max score

BONUS LAB - Birthday

- LAB: Create a birthday dictionary by asking names and birthdays in the following format:

  Enter Name: Arcan

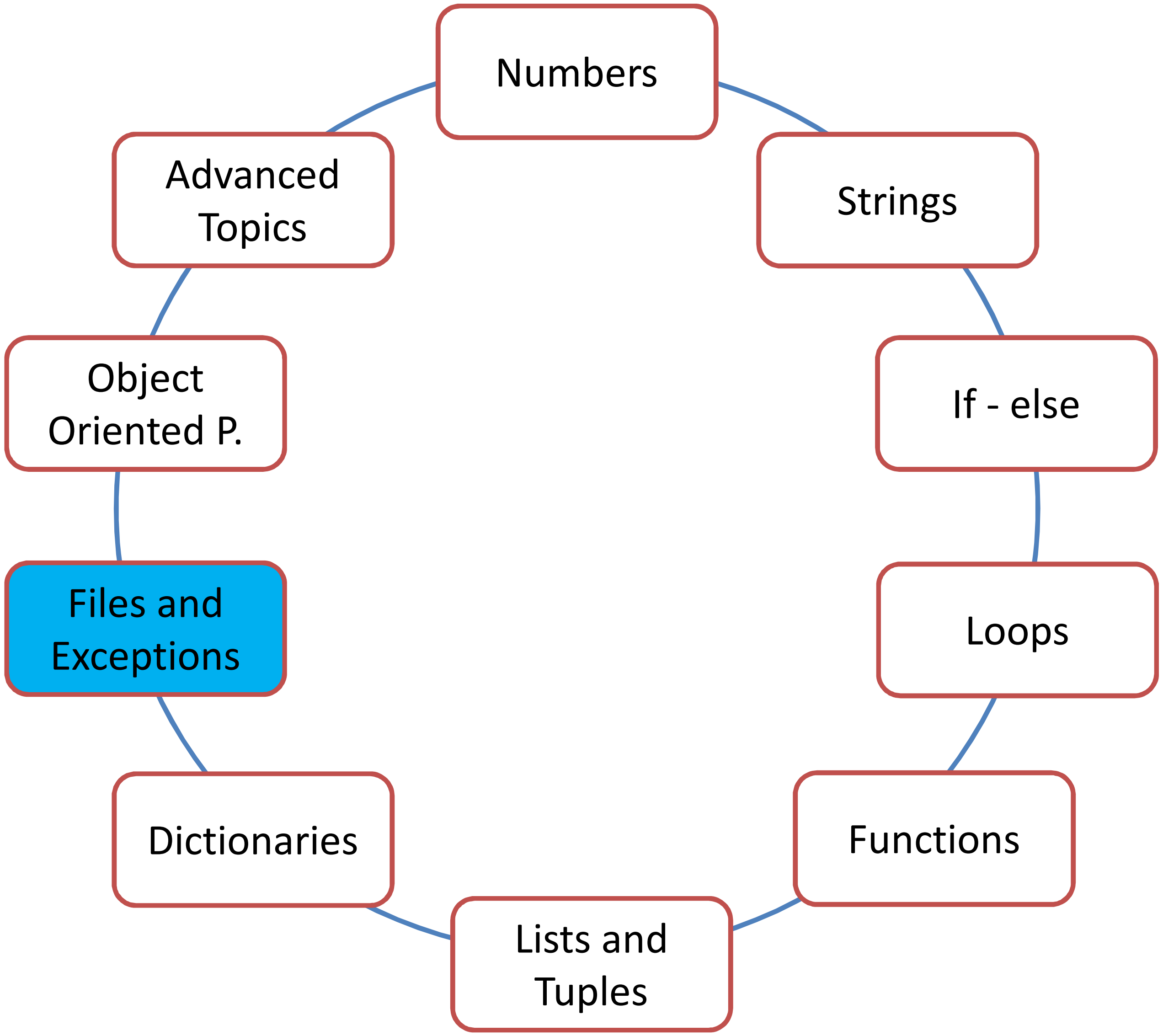
  Enter Month: 8

  Enter Day: 22

  Another record? y/n:

  - Print all names and birthdays in a format like above
  - Read a month and print names who has a birthday in that month
- Get the system date with datetime.date.today() and check and congratulate the person who has birthday today

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

9. Files and Exceptions

- Howto open a file?

  f = open('fruit.txt','r')

  - File object f created to access file fruit.txt
  - Fruit.txt is in the home folder of this user, otherwise use full path. Consider using r' ' raw string not to worry about metachars
  - Opening modes are
    - ✓r – read-only default
    - ✓w – write means create or overwrite if exist!
    - ✓a – append if exists, or create if not
    - ✓r+ – open read and write
  - Default mode is string text, you can add b for byte mode

File Modes

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. 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.
rb+	Opens a file for both reading and writing in binary format. 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.
wb	Opens a file for writing only in binary format. 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.
wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
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.
ab	Opens a file for appending in binary format. 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.
ab+	Opens a file for both appending and reading in binary format. 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.

File Operations

- f.read() : Reads entire file into a single string
- f.readline(): Reads a line from the string. '' at the end
- f.readlines(): Reads all lines from file to a list
- for line in f: (This line processing memory efficient&fast)

  ```
      print(line,end='')
  ```
- if you want to read all lines into a list use
  - list(f) or f.readlines()
- f.write('This is a test\n')   #Write to a file
- f.close() #Close
- f.flush() #Flush but not close
- f.seek(N,[0,1,2]) # Change position N item. 0 from beginning, 1 from current, 2 from end
- f.tell() # Current position in bytes
- open('f.txt',encoding='Latin-1')
- Preferred way of opening a file => Autoclose (No need for f.close())

  ```
  with open('/usr/share/dict/words', 'r') as f:
      data = f.readlines()  # Reads all lines to data list at once
  ```

File and Directory Methods of OS Module

- Rename a file
  - os.rename('test1.txt', 'test2.txt')
- Remove a file
  - os.remove('test2.txt')
- Create a directory 'test'
  - os.mkdir('test')
- Changing a directory to '/home/class'
  - os.chdir('/home/class')
- Print Working Directory
  - os.getcwd()
- Remove '/tmp/test' directory.
  - os.rmdir('/tmp/test')
- Also use shutil module for file operations like copy,move, etc.

File LABS

- LAB: Create a file 'numbers.txt' and write 20 random numbers between 1-100. Close the file then read and display it. Write to 'numbers2.txt' file double the original numbers.
- LAB: Create fruit1.txt and fruit2.txt files manually, write a Python script to show differences at lines one to one.
- LAB: Create an employee record file for the company. First ask for the filename and enter employee data in the following format: Employee ID, First and Last Name, and Department Name. Use '0' for the Employee ID to terminate.
  - For the above script, create a menu that can search/add/delete/append/print employees. Search and delete operations will be based on unique Employe_IDs. Menu: (C)reate/(P)rint/(S)earch/(A)ppend/(Q)uit:

File Extra LABS

- BONUS LAB: Consider employee script. Read **company.txt** file and convert the whole company data into CSV format and write to file **company.csv.** Then read it again using the csv module and just print names of employees
- BONUS LAB: Consider you have a zipped file **numbers.txt.gz**. How can you read and display this file?
- BONUS LAB: Read from a json file called **distros.json** and print name values
- BONUS LAB: How to read and write data in Excel format? Consider installing and using the **openpyxl** module
- BONUS LAB: How to read data in XML format?

Exception

- Exception is an error that occurs while a program is running like division by zero. If you don't handle it in most cases program halts
- List of exceptions

  https://docs.python.org/3/library/exceptions.html
- Simple Code – E07_Exception1.py

```
#!/usr/local/bin/python3
a = int(input('First Number '))
b = int(input('Second Number: '))
c = a/b
print(a,'/',b,'=',c)
```

  - Try b = 0 for this code

Handling Exception

- How can you handle this?
- You can change the program and check if b == 0 first

```
E07_Exception2.py
#!/usr/local/bin/python3
a = int(input('First Number '))
b = int(input('Second Number: '))
while b == 0:
    print('You can not choose b equal to 0, try again!')
    b = int(input('Second Number: '))
c = a/b
print(a,'/',b,'=',c)
```

Handling Exception

- You can not check all possible errors before it happens
  - What is user enter strings instead of numbers?
- Therefore you can use catch structure if you suspect some error for parts of you program

```
#!/usr/local/bin/python3
try:
    a = int(input('First Number '))
    b = int(input('Second Number: '))
    c = a/b
except ZeroDivisionError:
    print('Division by Zero! Be careful..')
```

Handling Multiple Exception

- Howto handle more than one type of exception?

```
try:
    file = open('employee.txt','r')
    a = int(input('First Number '))
    b = int(input('Second Number: '))
    c = a/b
except ZeroDivisionError:
    print('Division by Zero! Be careful next time..')
except FileNotFoundError:
    print('File does not exist! Check name/path')
```

Handling Multiple Exception

- Still we can create un-handled exception by entering string values to a and b. How to handle all exceptions?

```
try:
    file = open('employee.txt','r')
    a = int(input('First Number '))
    b = int(input('Second Number: '))
    c = a/b
except ZeroDivisionError:
    print('Division by Zero! Be careful next time..')
except except FileNotFoundError:
    print('File does not exist! Check name/path')
except:
    print('Enough is enough! Some other exception!')
```

Exception Extras

- How do you exit from Python?

  import sys

  sys.exit(3) -> 3 is error code, default is 0
- How can you run statements if exception did not occur?

```
try:
    statements
exception:
    statements
else:
    statements  # this only executed if no exception
```

Exception Extras

- Finally expression

```
try:
    statements
exception:
    statements
finally:
    statements  # This code is always executed
```

- Finally is created for clean-up purposes. So even you want to exit with sys.exit(), finally statements still executed!

Raise Exception

- How can you raise an exception?

```
a = 42    # if you set other than 42 no exception throws
try:
    if a == 42:
        raise ValueError('Apple')
    print('Everything is ok.')
except ValueError as e:
    print('There was an exception about Universe!')
    print(e)
```

Custom Exception

- How can you create your own exception?

```
class MyException(Exception):
    pass

a = -273
try:
    if a == -273:
        raise MyException('Zero Kelvin!','Freeze')
    print('Everything is ok.')
except MyException as e:
    print(e.args)
```

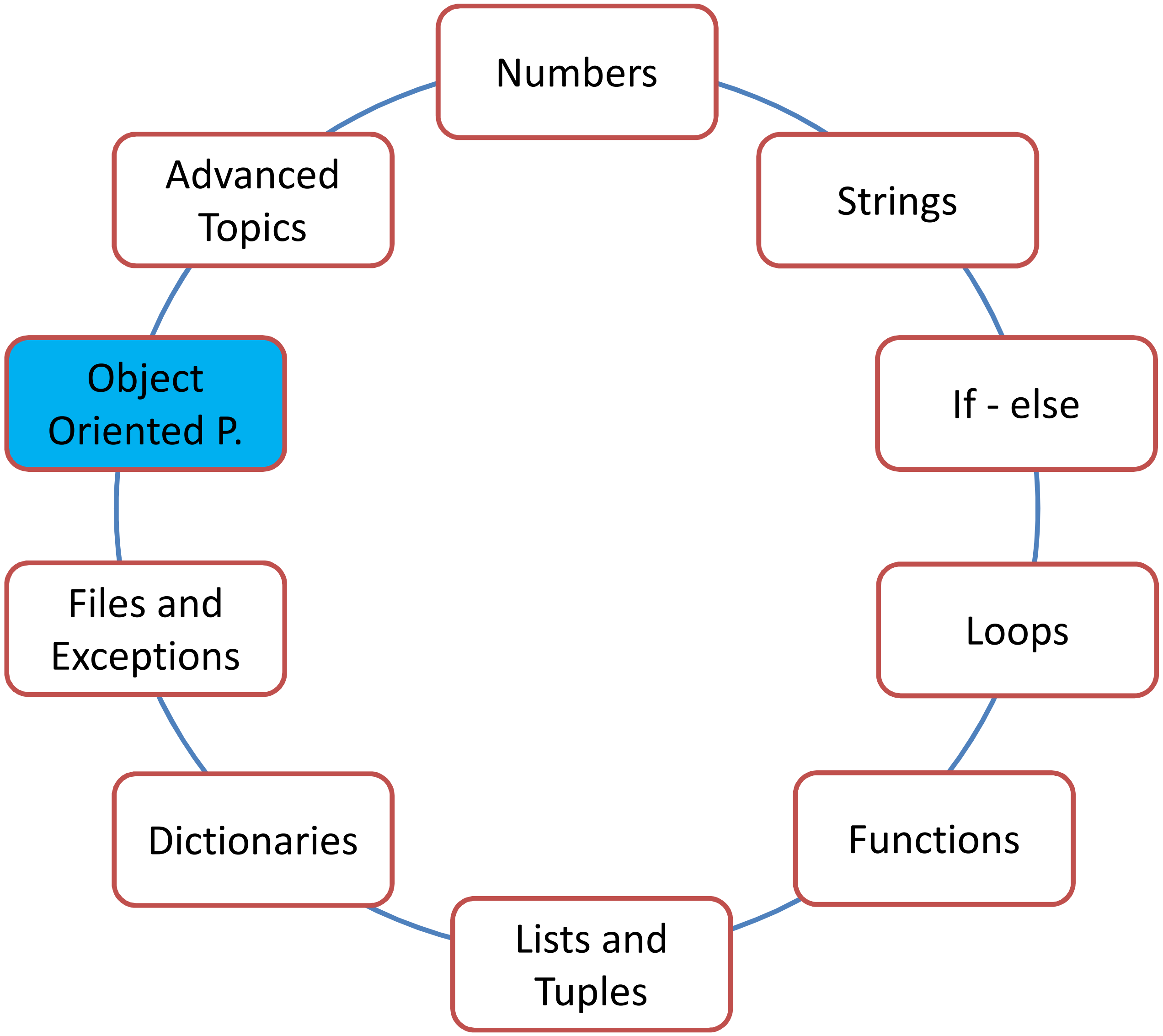
Assertion

- How can you create your own exception?

```
a = -1
try:
    assert(a>0)
    print('Everything is ok.')
except AssertionError as e:
    print('Negative Value is not ok!')
```

- LAB: Read a value from keyboard and check if it is float

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

10. OOP Terminology

- **Object:** A unique instance of a class, contains both data and methods
- **Class:** Prototype or blueprint for an object
- **Instance:** An individual object of a certain class
- **Instantiation:** The creation of an instance of a class
- **Method :** Function defined in class
- **Class variable:** A variable that is shared by all instances of a class
- **Instance variable:** A variable that is defined inside a method and belongs only to the current instance of a class (object)
- **Function overloading:** The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.

OOP Key Concepts

- **Encapsulation:** Binding (or wrapping) code and data together into a single unit is known as encapsulation
  - Employee class both contains data and methods
- **Inheritance:** One object acquires all the properties and behaviors of parent object. It provides code reusability.
  - Employee is a class
  - Manager is a subclass of Employee inheriting all attributes and methods
- **Polymorphism:** Polymorphism is the ability of an object to take on many forms achieved by overriding and overloading
- **Abstraction:** Hiding internal details and showing functionality is known as abstraction -Achieved by interfaces and abstract classes. (vehicle is abstract class, car is subclass)
- Python uses **multiple inheritance** unlike Java

OOP – Practice

```
#!/usr/local/bin/python3
class Employee():
# Common base class for all employees'
   emp_count = 0

   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.emp_count += 1

   def display_count(self):
     print(f'Total Employee {Employee.emp_count}')

   def display_employee(self):
      print('Name : ', self.name,  ', Salary: ', self.salary)
```

OOP – Practice

```
# Create 2 employees Alice and Bob
emp1 = Employee('Alice', 2000)
emp2 = Employee('Bob', 5000)
emp1.display_employee()
emp2.display_employee()
print(f'Total Employee {Employee.emp_count}')
```

- Here we don't see any advantage of OOP
- What about creating new classes like Manager, Sales, Support? We can subclass common Employee class and inherit all data and functions. Also we can add new functions or just override functions in the Employee class. Polymorphism allows us to call same function but get different behaviors for different objects.

OOP – Practice

- How can we create a subclass Manager from Employee?
  - class Manager(Employee):
- How can we create a constructor for Manager?
  - We can call superclass(Employee) constructor and add additional attributes
  - super().__init__(name,salary)
- How can we display Manager objects differently?
  - We want to display additional attributes
  - We use polymorphism and override existing display_employee() method

Inheritance

- **Inheritance and overriding**

```
class Animal:
   def __init__(self, name):    # Constructor
      self.name = name
   def speak(self):             # Abstract method
      raise NotImplementedError("Subclass must implement abstract method")

class Dog(Animal):
   def speak(self):
      return self.name+' says Woof!'

class Cat(Animal):
   def speak(self):
      return self.name+' says Meow!'

dog1 = Dog('Max');  dog2 = Dog('Jack');  cat1 = Cat('Lucy')
print(dog1.speak());  print(cat1.speak()); print(dog2.speak())
```

Overloading

- **Overloading constructor**

```
class Animal:
    def __init__(self,name,legs):
        self.name = name
        self.legs = legs

class Bear(Animal):
    def __init__(self,name,legs=4,hibernate='yes'):
        Animal.__init__(self,name,legs)  # super() is also ok
        self.hibernate = hibernate
    def __init__(self,name='Yogi',legs=4,hibernate='yes'):
        Animal.__init__(self,name,legs)
        self.hibernate = hibernate

bear1 = Bear('Bone-Cracker');  bear2 = Bear()
print(bear1.name);  print(bear1.legs);  print(bear1.hibernate)
print(bear2.name);  print(bear2.legs);  print(bear2.hibernate)
```

Overriding

- **Overriding print**

```
class Account:
   def __init__(self,owner,balance=0):
      self.owner = owner
      self.balance = balance
   def __str__(self):
      return f'Account owner:   {self.owner}\nAccount balance: ${self.balance}'

   def deposit(self,dep_amt):
      self.balance += dep_amt
      print('Deposit Accepted')

   def withdraw(self,wd_amt):
      if self.balance >= wd_amt:
         self.balance -= wd_amt
         print('Withdrawal Accepted')
      else:
         print('Funds Unavailable!')

acct1 = Account('Ali',1000); print(acct1) #print the object with __str__
acct1.owner;  acct1.balance;  acct1.deposit(50);  acct1.withdraw(100);
acct1.withdraw(1500) # Rejected
```

Multiple Inheritance

- **Multiple Inheritance**

```
class Car:
    def __init__(self,wheels=4):
        self.wheels = wheels  # All cars have four wheels by default.

class Gasoline(Car):
    def __init__(self,engine='Gasoline',tank_cap=50):
        Car.__init__(self)
        self.engine = engine
        self.tank_cap = tank_cap # Represents fuel tank capacity in liters
        self.tank = 0
    def refuel(self):
        self.tank = self.tank_cap
```

Multiple Inheritance

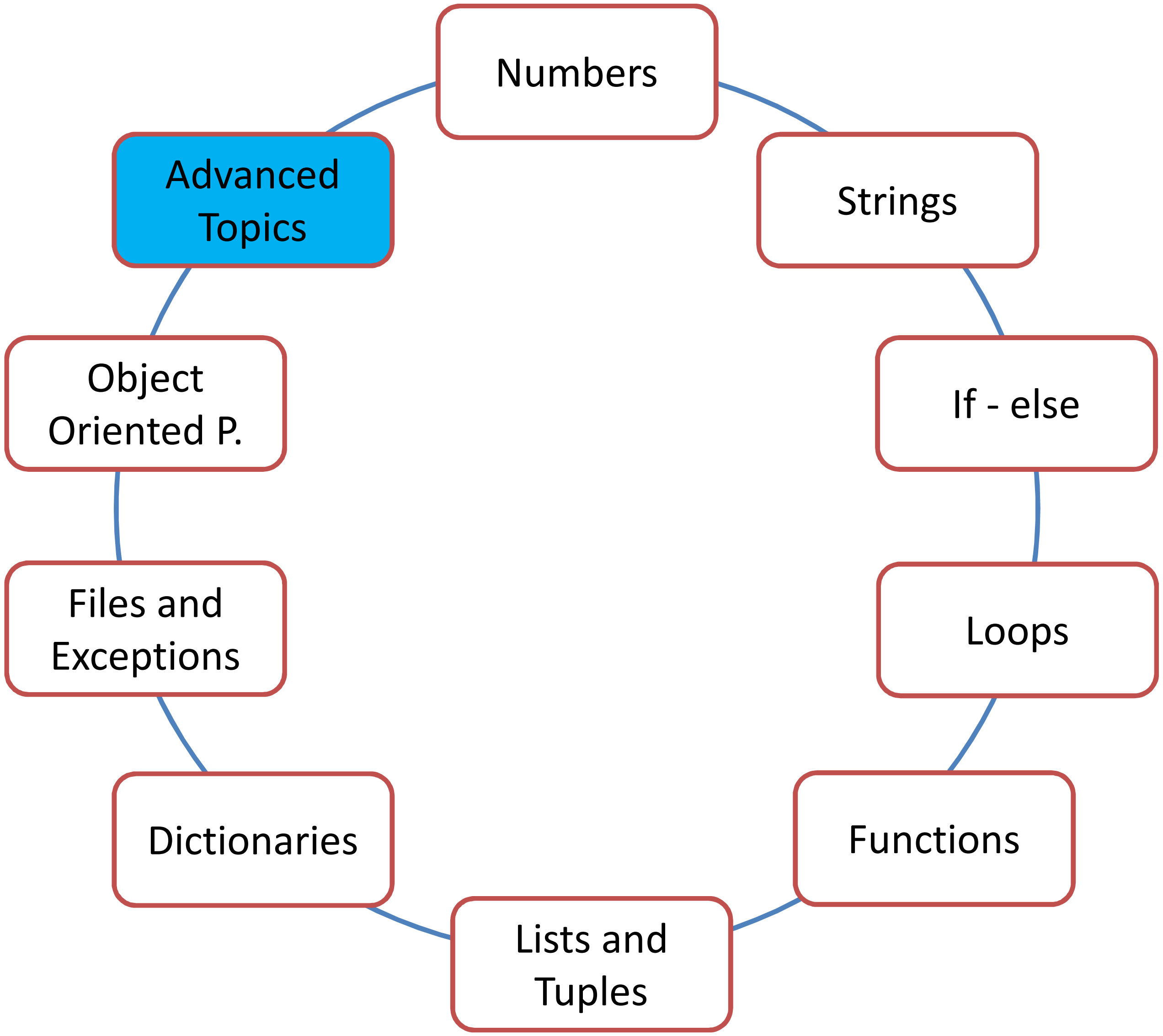
- **Multiple Inheritance**

```
class Electric(Car):
    def __init__(self,engine='Electric',kWh_cap=60):
        Car.__init__(self)
        self.engine = engine
        self.kWh_cap = kWh_cap # Represents battery capacity in kilowatt-hours
        self.kWh = 0
    def recharge(self):
        self.kWh = self.kWh_cap

class Hybrid(Gasoline, Electric):
    def __init__(self,engine='Hybrid',tank_cap=30,kWh_cap=10):
        Gasoline.__init__(self,engine,tank_cap)
        Electric.__init__(self,engine,kWh_cap)

prius = Hybrid();  print(prius.tank);  print(prius.kWh) ;  prius.recharge();
print(prius.kWh)
```

OOP – LAB

- LAB: Using OOP, create a base class Employee with name and salary attributes and display_employee() method. Create a Manager class by subclassing Employee with an additional bonus attribute.
    - Override display_employee() method to display additional bonus field for Manager objects.
    - If Manager is created without bonus value, it should assume default bonus as 2000 (overload constructor)
    - Finally create two Employees Alice and Bob with initial salaries. Also create two Manager Charlie and Steve. Charlie has bonus but Steve created with default bonus. Print credentials of all employees, including total number and bonus salary of Charlie.

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

Regular Expression

- Practice to remind OS metachars and Regex Metachars using linux.words file and egrep command
- Check Website: https://regex101.com/
- Python module for regex → **re**

```
>>> import re
>>> line = '/usr/local/bin/python/test'
>>> new = re.split('/',line)
>>> line2 = '/usr/local:bin/python/test'    # Split / or :
>>> new2 = re.split('[/:]',line)
>>> new2.remove('')
>>> line2 = ' => '.join(new2)
```

Search and Match

- The **search** method is essential for regex searches

  line = 'Once upon a time in China. The year 1003..'

  result1 = **re.search**('(Chi.*)\.',line)

  result1.group(1) => 'China. The year 1003.'

  - Why? Because of default greediness, how to fix?

  result1 = re.search('(Chi.*?)\.',line)

  result1.group(1) => 'China'

  - How to make it case insensitive?

  result1 = re.search('(chi.*?)\.',line, re.I)

  - Exact **match**? result1 = **re.match**('(Chi.*)\.',line)

Regex Grouping and Findall

- The **findall** method returns a list of all matches

  line = 'Littlewood is a little town in England'

  result = re.findall('(lit\w+)', line, re.I)

  print(result)
- Grouping is useful for extraction, repetition, and sub patterns
  - (a.c){2,4} => Repetition
  - (alice|bob|charlie) => Sub patterns
  - result.group(1), result.group(2) => Extraction
- Also keep in mind about metachars, Python and regex both has it. Therefore use r' ' when you perform a search

Regex Find and Replace

- One of the most important re methods is **sub**.
- Syntax: **re.sub**(pattern, repl, string, max=0)
  - This method replaces all occurrences of the RE pattern in string with repl, substituting all occurrences unless max provided. This method returns modified string.

```
phone = '0212-226-3030 # This is Phone Number'
```

- Delete Python-style comments starting with # till end

```
num = re.sub(r'#.*$', '', phone)
```

- Remove anything other than digits

```
num = re.sub(r'\D', '', num)
```

- Finally remove if first digit is 0

```
num = re.sub(r'^0', '', num)
print('Phone Number : ', '+90 (' + num[:3] + ') ' + num[3:])
```

Regex Patterns

Pattern	Description
^	Matches beginning of line.
$	Matches end of line.
.	Matches any single character except newline. Using m option allows it to match newline as well.
re*	Matches 0 or more occurrences of preceding expression.
re+	Matches 1 or more occurrence of preceding expression.
re?	Matches 0 or 1 occurrence of preceding expression.
re{ n}	Matches exactly n number of occurrences of preceding expression.
re{ n,}	Matches n or more occurrences of preceding expression.
re{ n, m}	Matches at least n and at most m occurrences of preceding expression.
a| b	Matches either a or b.
(re)	Groups regular expressions and remembers matched text.

Regex Special Char Classes

Example	Description
.	Match any character except newline
\d	Match a digit: [0-9]
\D	Match a nondigit: [^0-9]
\s	Match a whitespace character: [\t\r\n\f]
\S	Match nonwhitespace: [^ \t\r\n\f]
\w	Match a single word character: [A-Za-z0-9_]
\W	Match a non-word character: [^A-Za-z0-9_]
\b	Match if beginning or at the end of a word
\B	Match if inside of a word

Regex Option Flags

- Regular Expression Modifiers: Option Flags
  - You can provide multiple modifiers using OR (|)

Modifier	Description
re.I	Performs case-insensitive matching.
re.L	Interprets words according to the current locale. This interpretation affects the alphabetic group (\w and \W), as well as word boundary behavior (\b and \B).
re.M	Makes $ match the end of a line (not just the end of the string) and makes ^ match the start of any line (not just the start of the string).
re.S	Makes a period (dot) match any character, including a newline.
re.U	Interprets letters according to the Unicode character set. This flag affects the behavior of \w, \W, \b, \B.
re.X	Permits 'cuter' regular expression syntax. It ignores whitespace (except inside a set [] or when escaped by a backslash) and treats unescaped # as a comment marker.

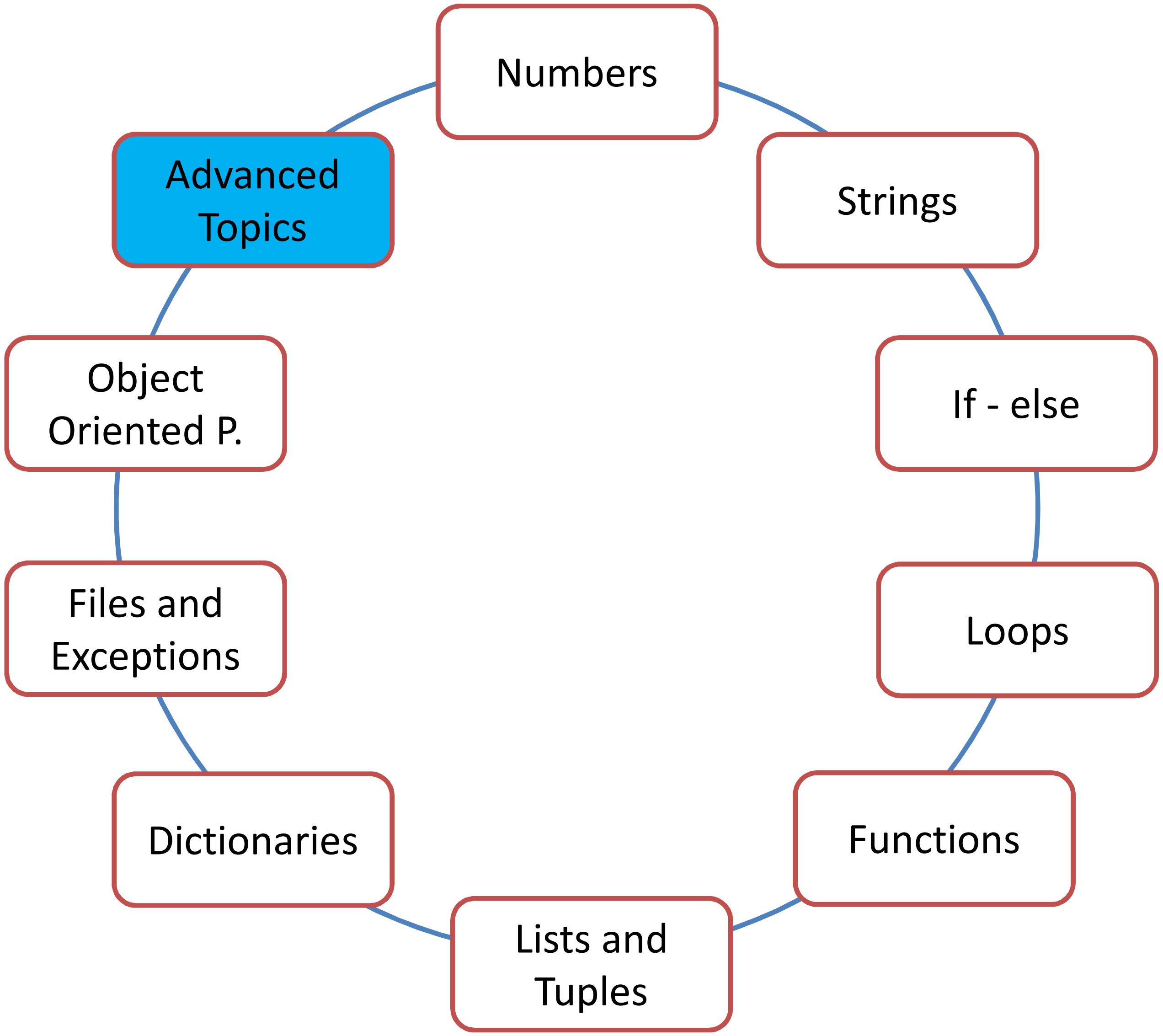
Regex Labs

- Example: Convert camel notation strings to snake notation strings.

```
import re
def convert(name):
    return re.sub('([a-z0-9])([A-Z])', r'\1_\2', name).lower()
print(convert('CamelCaseVariable'))
```

- LAB: Check if keyboard input value is float with regex
- LAB: Find a regex to validate only these values: -255..0..255
- LAB: You are given a file 'email_errors.txt' that contains emails within lots of error messages. You are asked to write a Python Program lab11_regex.py that cleanly extracts emails and prints on the screen.
    - Emails may contain letters,numbers,_,.,-, and one @ char in the middle e.g. tahsin.demiral@gmail.com

Work Flow


Numbers
Strings
If - else
Loops
Functions
Lists and Tuples
Dictionaries
Files and Exceptions
Object Oriented P.
Advanced Topics

OS and Python Communication

- I/O fundamentals and redirection:
  - Stdin
  - Stdout
  - Stderr
- How to send output other than stdout?

```
log = open('log.txt', 'a')
print(6,  28, 42, file=log)       # Prints to file log.txt
print('The Hitchhikers Guide to Galaxy')   # To stdout
log.close()
```

Python on Linux

- Edit Python script **lab12_os_print.py**

```
import fileinput
with fileinput.input() as f:
    for line in f:
    print('Coming from OS:',line, end='')
```

  - # cat /etc/passwd | **lab12_os_print.py**
- How to run UNIX command within Python
  - import os ; os.system('date')   #Deprecated
  - import subprocess ; subprocess.call('date') #New
    - ✓How can you access output to a python parameter?
    - ✓Write a script for ls -l command:**lab12_os_command.py**

Command Line Parameters

- Parameter Passing. Module sys has to be imported

```
# Iteration over all arguments using sys.argv
import sys
for each_arg in sys.argv:
    print(each_arg)
```

- LAB: Using parameter passing, give two numbers and operator +, or * to add or multiply and print result
- LAB: Write a python script to search a keyword within files **./lab12_command_line.py** apple fruit1.txt fruit2.txt
  - Note that script should accept one search key and many filenames

LAB – Password Cracking

- LAB: Write a Python program that cracks passwords using /etc/shadow hashed information. Note that you can use crypt module and crypt function to generate SHA512 hashes. You need to give shadow line of the user and let python tries all possibilities (brute-force) Note that more than 4 chars takes very long time.
- Bonus LAB: Before you start ask whether password consists of alphanumeric, alphabetic, or just numeric characters to speed up trial and error.

Appendix A. Language Extras

- What is not in Python Language?
- ++ and -- operator does not exist. Use +=1 instead
- Case/Switch Statement is not included in Python
  https://www.python.org/dev/peps/pep-3103/
- Is ; used in Python?
  - ; is not the end of expressions
  - a = 1; b = 2; print(a + b) *# Three statements on one line*
- {} not used for blocks. In Python {} is dictionary!
- You must use indentation. Convention is 4 spaces (PEP8)
- Howto exit from Python script?
  - sys.exit(3) => Exit with error code 3. Check $? in Linux

Print and Formatting

- **print**() function details
  - How do you suppress newline and specify seperator?

    ```
    print('apple', 'orange', 'banana', sep=':',  end=' ')
    apple:orange:banana
    ```
  - Formatted output

    ```
    a=16.6667
    print('%8.2f' %a)  # Just Like C printf function!
    S='The result is '
    print('%20s %8.2f' % (S,a))
    print(f'{S} {round(a,2)} ')   # New format string
    print("{0:<20s} {1:8.2f}".format(S, a))
    ```

Import, Input, Eval, and Exec

- **import vs. from/import**

  import time as t

  t.sleep(0.1)

  from time import sleep => import only sleep function

  sleep(0.1) => No need to use time.sleep()
- **input()** always returns raw string if you want to evaluate input then use eval() function
- **eval()** evaluates statement and returns value, **exec()** runs the code returns None

  a = 42

  b = eval('17 + a') or use exec('b = 17 + a')

Set and FrozenSet

- Sets are just like Lists, except un-ordered and no duplication allowed. This is same concept as math sets. Major functions are add() and remove(), there is no indexing. Frozensets are just like tuples (immutable)
- Set example: Create 6 unique numbers within 1..49

```
import random as rd
my_set = set()
while len(my_set) < 6:
   value = rd.randint(1,49)
   my_set.add(value)
print('Winning numbers are', my_set)
print('Sorted numbers are', sorted(my_set))
my_frozen_set = frozenset(my_set)
```

Function Parameters: *args, **kwargs

- In Python, you can use ***args** and ****kwargs** to dynamically choose number of parameters when you call a function

```
def add_numbers(*args):
    total = 0
    for var in args:
        total += var
    return total
print(add_numbers(1,2,3,4,5))

def add_numbers(**kwargs):
    total = 0
    for key, value in kwargs.items():
        print(key,'=>',value)
        total += value
    return total
print(add_numbers(first_number=3,second_number=5, third_number=8))
```

Datetime Module

- Datetime module has many useful methods to process dates, times, and timezones.

  from datetime import datetime, date, timedelta

  import pytz

  today = date.today()

  print("Current date:", today)

  print("Current year:", today.year, "month:", today.month, "day:", today.day)

  print(pytz.all_timezones[:5])

  local = datetime.now()

  print("Local:", local.strftime("%m/%d/%Y, %H:%M:%S"))

  tz_NY = pytz.timezone('America/New_York')

  datetime_NY = datetime.now(tz_NY)

  print("NY:", datetime_NY.strftime("%m/%d/%Y, %H:%M:%S"))

Walrus Operator in Python 3.8

- In Python 3.8 Walrus operator is in use
  - This operator helps you check value and assign to variable at the same time
  - Lets throw a dice, 6 wins

```
import random as rd
print('Throwing a dice, 6 wins')
if (dice := rd.randint(1,6)) == 6:
    print('You Win!!')
else:
    print(f'You throw => {dice}')
```

The End