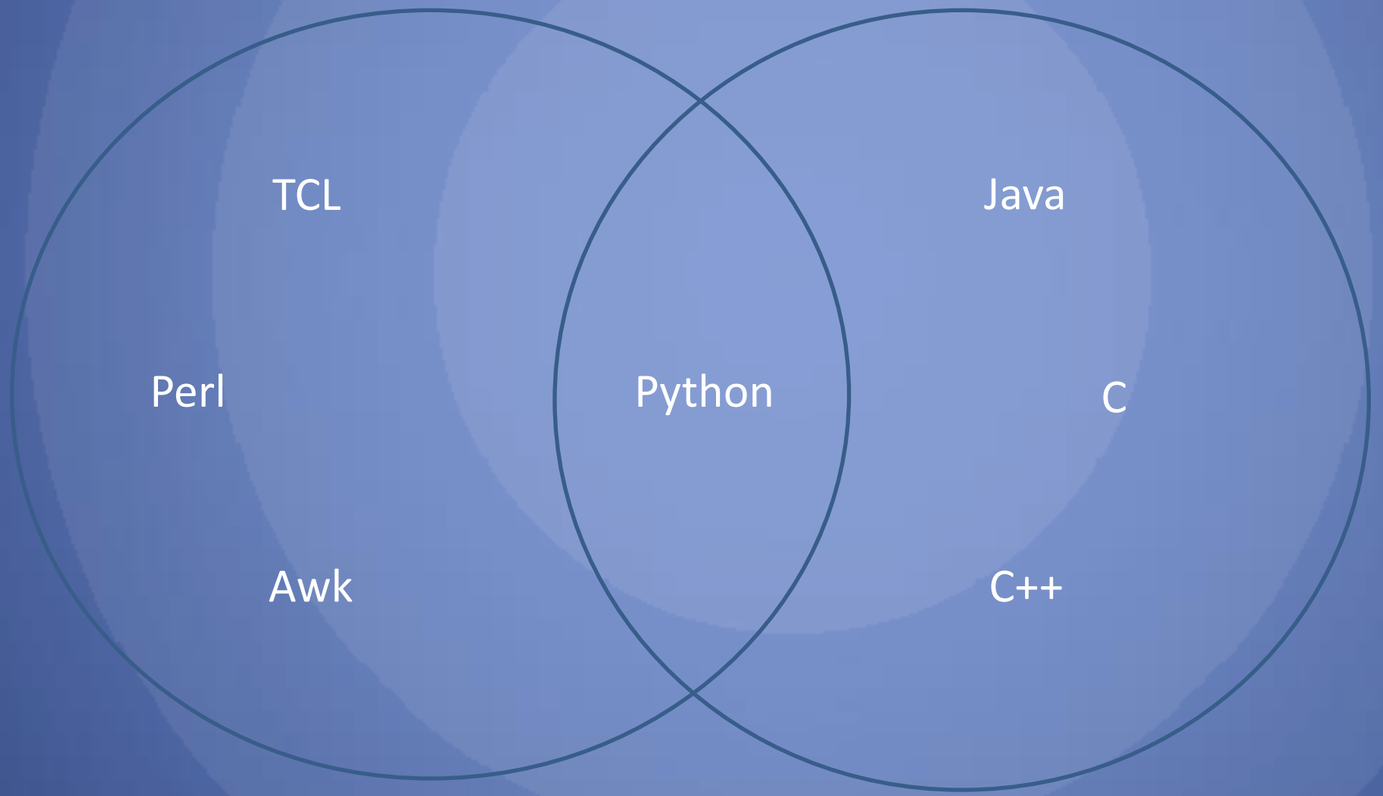
PYTHON

What is Python?

- **Python is a high-level programming language.**
- **Optimized for programmer productivity.**
- **Code readability, and software quality.**

- **Created by Guido van Rossum.**
- **First released in 1991.**
- **Named after the famous British Comedy group Monty Python and the Holy Grail.**

Python: A Complete Language



Features of Python

Portable

- Python compiles and runs on every major platform currently in use.
- Python can be used on 21 different operating systems and environments.

debian

redhat

gentoo

Powerful

- **Python's toolset places it between traditional scripting languages (Tcl, Scheme and Perl).**
- **And systems development languages (C ,C++ and Java).**

Mixable

Python Programs can be Easily glued to Components Written in other Languages.

Big Names in Python's Users

Major Uses Of Python

- Embedded Scripting
- Image Processing
- GUIs (Tkinter, gtk, Windows)
- Scientific and Numeric Programming
- Database Programming
- Internet Scripting

WHY PYTHON WAS NEEDED?

- Need of a language that was easy to read and use.
- At the Same time also fulfilling the requirement of a language to be object oriented.
- Need of an highly extensible language that can be embedded in to existing applications.
- Need of an alternative to traditional available languages, so that new opportunities can be explored.

Getting Python

1) **Python Official Website :** http://www.python.org/

2) You can download the Python documentation from the following site. The documentation is available in HTML, PDF, and PostScript formats.
   Python Documentation Website : www.python.org/doc/

Which Python?

- Python 2.7
  - Current version on Eniac, so we'll use it
  - Last stable release before version 3
  - Implements some of the new feature in version 3, but fully backwards compatible
- Python 3
  - Released a few years ago
  - Many changes (including incompatible changes)
  - Much cleaner language in many ways
  - Strings use Unicode, not ASCII
  - But a few important third party libraries are not yet compatible with Python 3 right now

Ease of Syntax

Java Code to print “Hello World” :

```
class Hello {
public static void main (String[] args)
{ System.out.println (“Hello, world.”);
}
```

Python code to print “Hello World”:

```
print  “Hello, World”
```

Assignment

- Names in python do not have an intrinsic type.
- Python determines the type of the reference automatically based on what data is assigned to it.

- e.g.

```
>>>    x=3

>>>    x
3
```

Multiple Assignments

- You can also assign to multiple names at the same time.

e.g.

```
>>>    x, y = 2, 3

>>>    x
2

>>>    y
3
```

Naming Rules

- Names are case sensitive.
- Cannot start with a number.
- They can contain letters, numbers, and underscores.

e.g.

bob Bob _bob _2_bob_ bob_2 BoB

Sequence Types

- Tuples (tuples)
- Lists [list]
- Strings “ string”

The ‘in’ Operator

- Boolean test whether a value is inside a collection (often called a container in Python).

```
>>>     t = [1, 2, 4, 5]
>>>     3 in t
False

>>>     4 in t
True

>>>     4 not in t
False
```

The ‘+’ Operator

- Used for concatenation of numbers and strings.

```
>>>     (1, 2, 3) + (4, 5, 6)
(1, 2, 3, 4, 5, 6)

>>>     [1, 2, 3] + [4, 5, 6]
[1, 2, 3, 4, 5, 6]

>>>     “Hello” + “ “ + “World”
‘Hello World’
```

Lists

```
>>>    li = ['abc', 23, 4.34, 23]

>>>    li [1] = 45

>>>    li

['abc', 45, 4.34, 23]
```

Tuples

```
>>>   t = (23, ‘abc’, 4.56, 3, ‘def’)

>>>   t[2] = 3.14                  //Error
                                   //doesn’t support item assignment
```

//You can make a fresh tuple and assign its reference to a previously used name.

```
>>>   t = (23, ‘abc’, 3.14, 3, ‘def’)
```

Creating and accessing dictionaries

```
>>> d = {'user':'bozo', 'pswd':1234}

>>> d['user']
'bozo'

>>> d['pswd']
1234

>>> d['bozo']

Traceback (innermost last):
  File '<interactive input>' line 1, in ?
KeyError: bozo
```

Updating Dictionaries

```
>>> d = {'user':'bozo', 'pswd':1234}

>>> d['user'] = 'clown'
>>> d
{'user':'clown', 'pswd':1234}
```

- Keys must be unique
- Assignment to an existing key replaces its value

```
>>> d['id'] = 45
>>> d
{'user':'clown', 'id':45, 'pswd':1234}
```

- Dictionaries are unordered
  - New Entries might appear anywhere in the output.

Removing Dictionary entries

```
>>> d = {'user':'bozo', 'p':1234, 'i':34}

>>> del d['user']                  # Remove one. Note that del is
                                   # a function.
>>> d
{'p':1234, 'i':34}

>>> d.clear()                      # Remove all.
>>> d
{}
>>> a=[1,2]
>>> del a[1]                    # (del also works on lists)
>>> a
[1]
```

The *if* statement

Syntax:

```
if expression:
        statement(s)
```

e.g.

```
a=9
if a>5:
        print “Statement is true”
```

The *else* Statement:

E.g.

```
a=9
if a>5:
    print “Statement is true”
else:
    print “Statement is false”
```

The *elif* Statement

e.g.

```
a, b, c = 20, 10, 30

if a>b and a>c:
    print “a is greater”
elif b>a and b>c:
    print “b is greater”
else:
    print “c is greater”
```

Thank You

See you all tomorrow….!!!!