Let's Play with Python

>>> _

Ramesh S

“The joy of coding Python should be in seeing short, concise, readable classes that express a lot of action in a small amount of clear code -- not in reams of trivial code that bores the reader to death.”

–Guido van Rossum

01

INTRODUCTION

What is Python

02

PRESENTATION

Facts about Python

03

Examples

You think you can code in Python?
Do it without even knowing it!

04

Data Types

Different Data Types in Python

05 Flow Control

Control and iterate as you wish

06 Functions

Want to reuse *that* particular piece of code?

07 File I/O

Read/Write into a file.
Just like that

08 Modules

Speak about reusability.
Many times

INTRODUCTION TO PYTHON

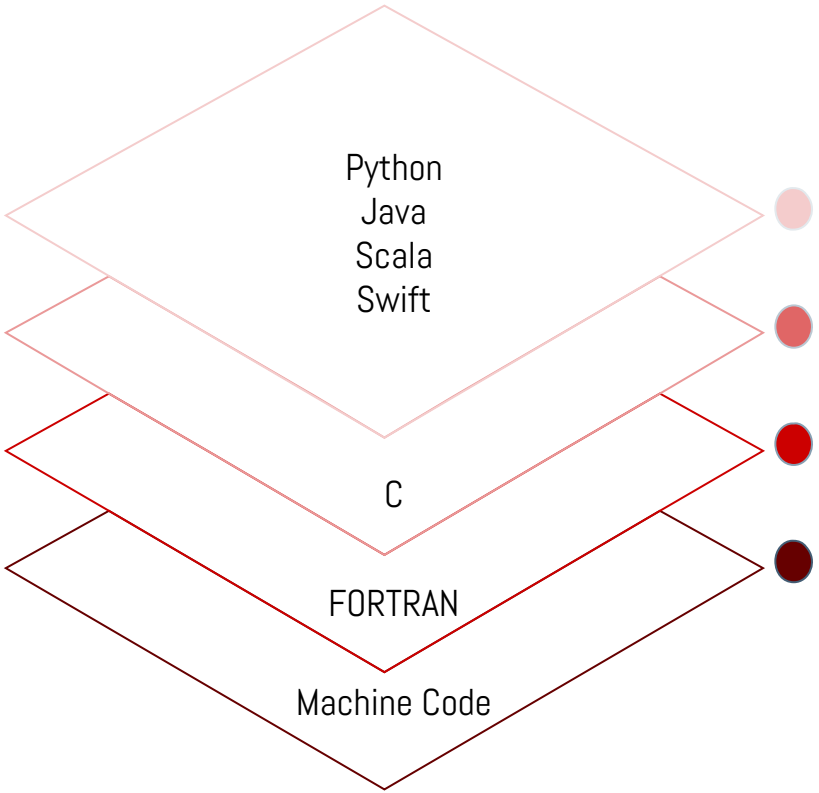
What is Python?

What is Python?

No, Not a Snake!

What is Python?

Python is a general purpose programming language

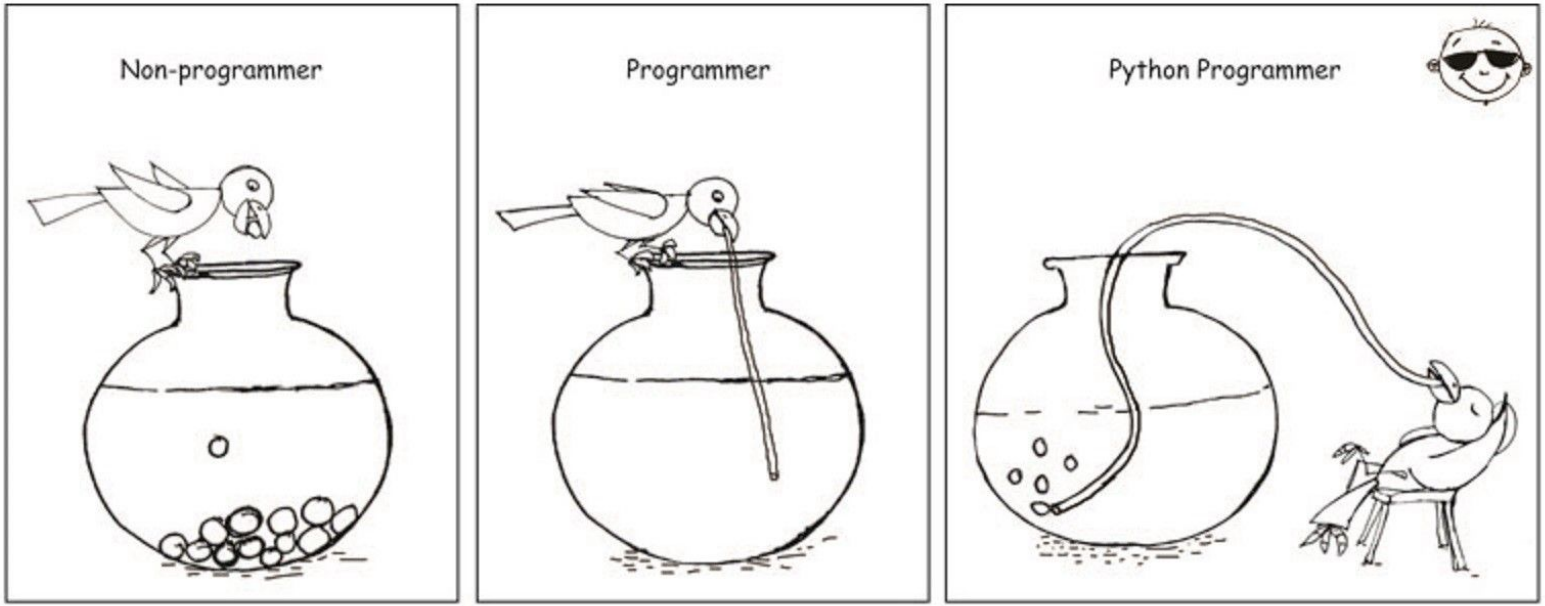


PYTHON

IS A HIGH LEVEL PROGRAMMING LANGUAGE

Python is interpreted

>>>>



Fun Fact #1

What is Python?

Python is

FUN

POWERFUL

FAST

>>>>

>>>>

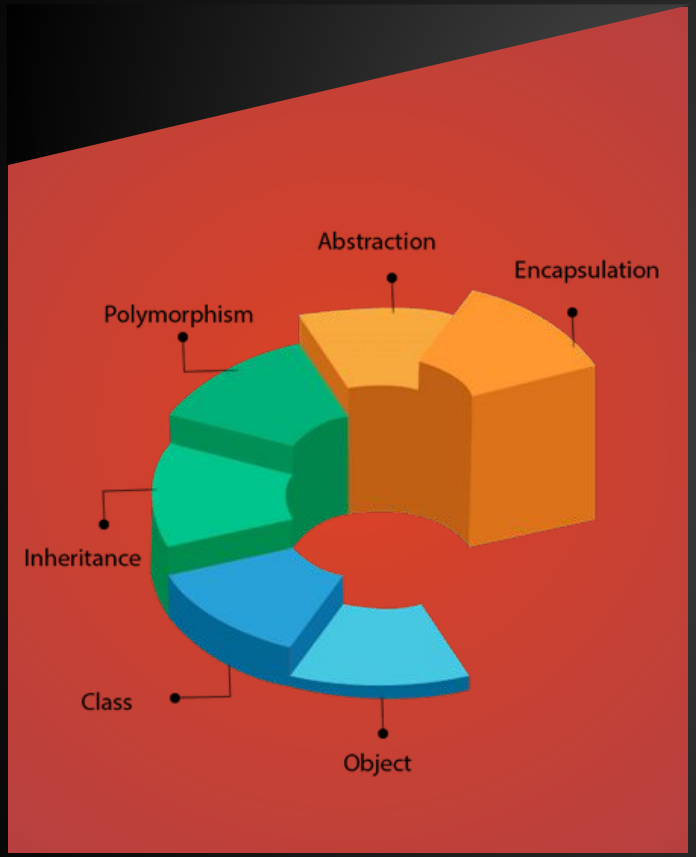
What is Python?

Python has huge standard library

What Developers Do During Compilation?

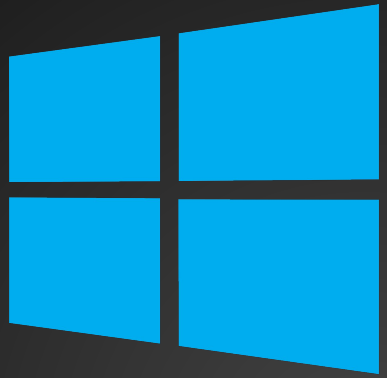
Fun Fact #2

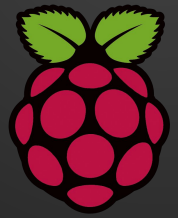
Python is dynamically typed



Python is Object Oriented

Python is Portable





Python is open source

PRESENTATION

Facts about Python

WHO CREATED PYTHON?

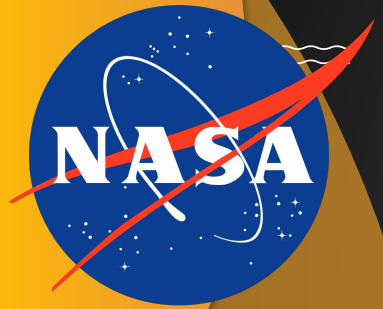
Guido van Rossum ››››

Benevolent Dictator for Life (BDFL)

- Python Software Foundation.

1991

When was Python released?

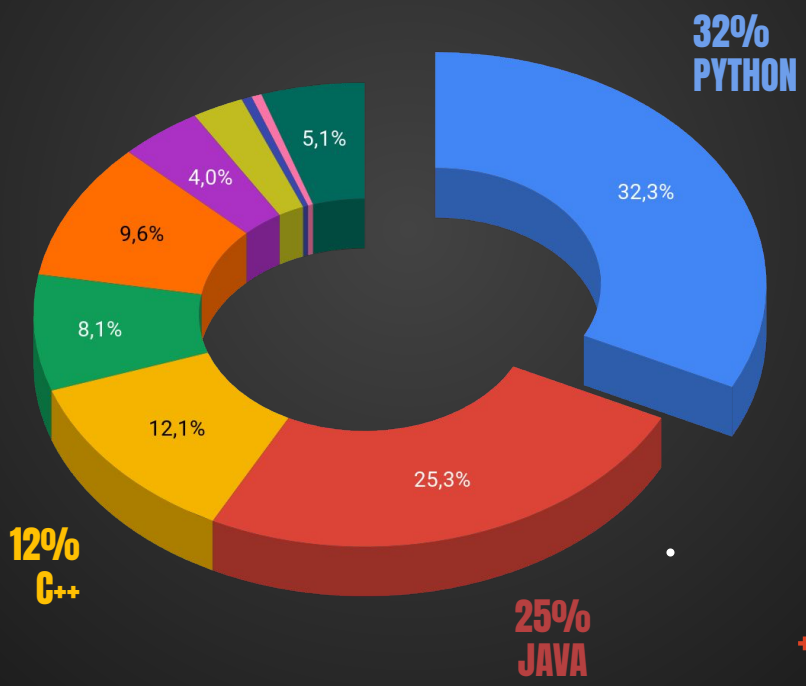


WHO IS USING PYTHON? >>>>

Why is it called Python?

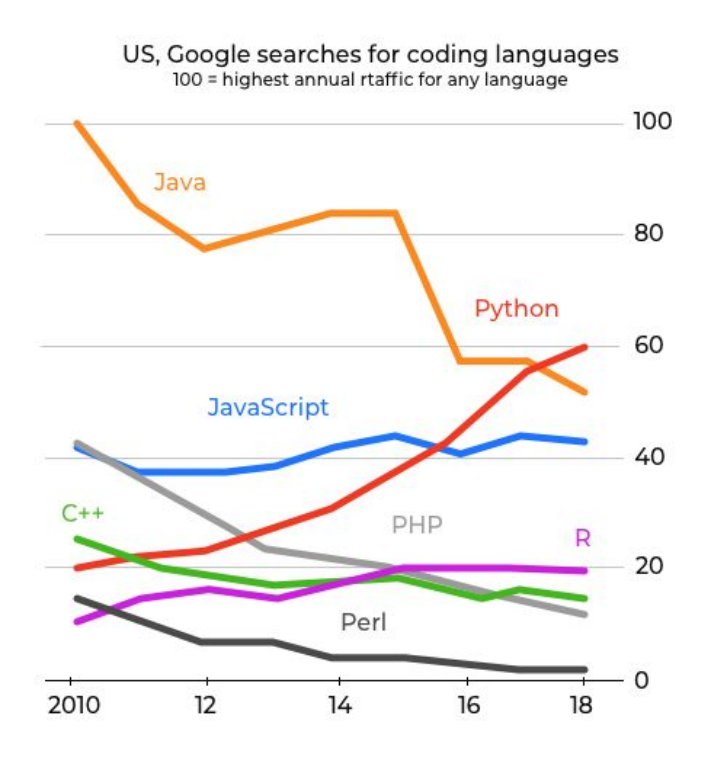
Monty Python's
FLYING
CIRCUS

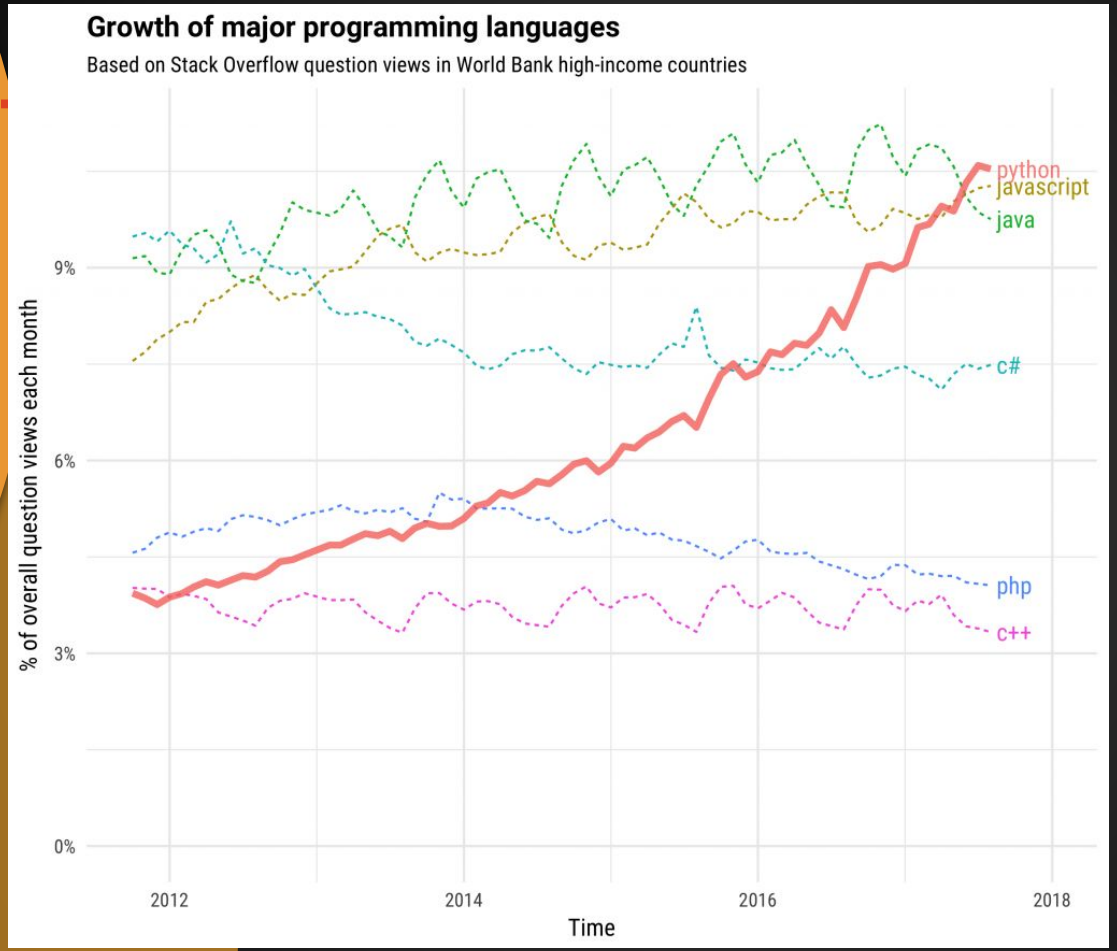
HOW POPULAR IS PYTHON?



Source: codeeval.com Survey

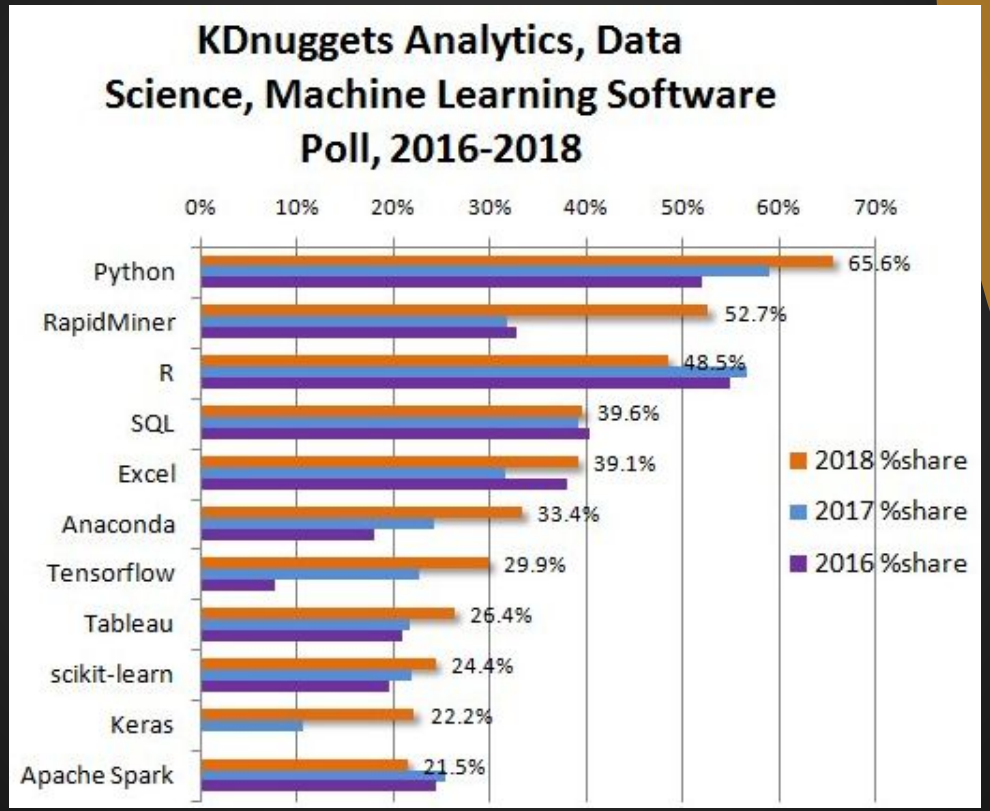
HOW POPULAR IS PYTHON?





HOW POPULAR >>>> IS PYTHON?

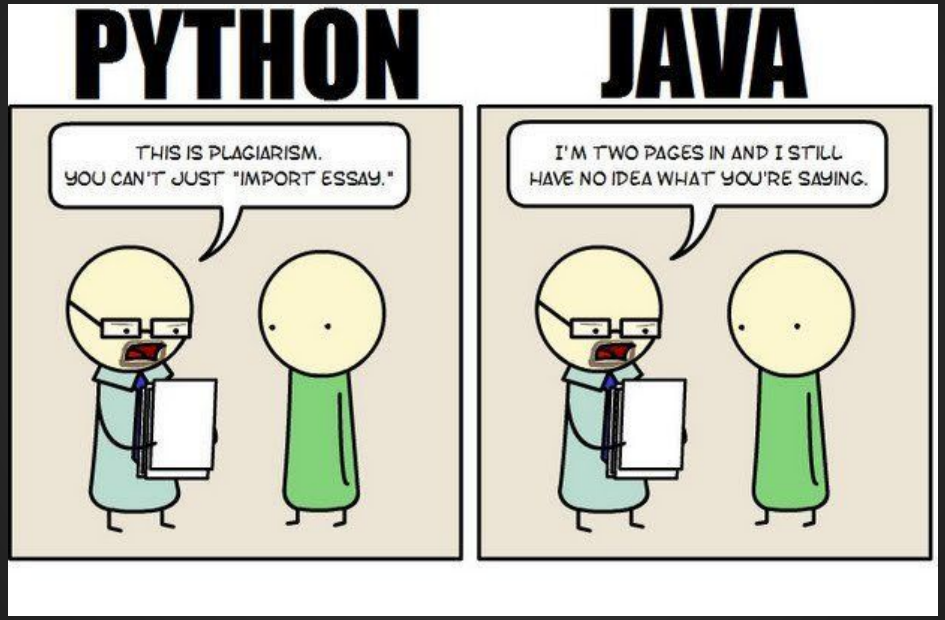
HOW POPULAR IS PYTHON?



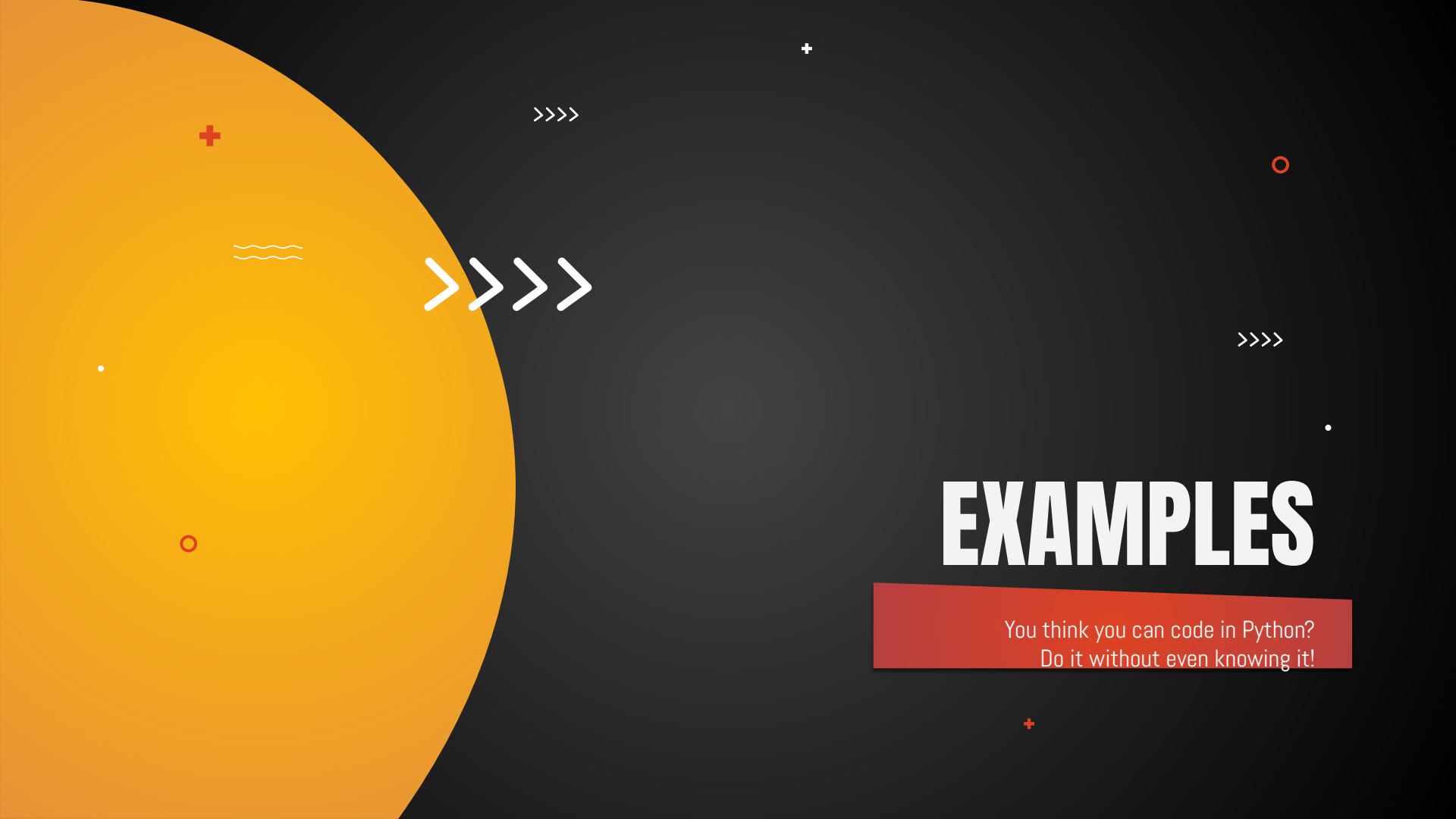
WHERE IS PYTHON USED?

- Scripting
- Rapid Prototyping
- Text Processing
- Web applications
- GUI programs
- Game Development
- Database Applications
- System Administration Automation
- Scientific Computing
- Machine Learning
- Big Data and Data Analysis

NO OFFENCE



Fun Fact #3


EXAMPLES
You think you can code in Python?
Do it without even knowing it!

GUESS WHAT IT DOES?

```
print(“Hello World!”)
```

GUESS WHAT IT DOES?

```
count=0
while count<11:
    print (count)
    count+=1
```

GUESS WHAT IT DOES?

```
kids=['Lahari','Lalitha','Jithu','Vishal','Ujwal','Nitish']
>>>
for kid in kids:
    print (kid)
```

GUESS WHAT IT DOES?

```
kids=['Lahari','Lalitha','Jithu','Vishal','Ujwal','Nitish']
for kid in kids:
    if len(kid)==6:
        print (kid)
```

>>>>

GUESS WHAT IT DOES?

```
kids=['Lahari','Lalitha','Jithu','Vishal','Ujwal','Nitish']
for kid in kids:
    if "it" in kid:
        print (kid)
```

>>>>

GUESS WHAT IT DOES?

```
marks=[96,87,67,81,81]
print(len(marks))
print(sum(marks))
print(min(marks))
print(max(marks))
print(sum(marks)/len(marks))
```

GUESS WHAT IT DOES?

```
marks={'maths':97,'science':'98','history':78}
for subject,marks in marks.items():
    print (subject, marks)
```

GUESS WHAT IT DOES?

```
f=open('dictionary.txt','r')
for line in f:
    if line.startswith('s'):
        print (line)
```

GUESS WHAT IT DOES?

```
for x in range(5):
    print(x)
```

>>>>

GUESS WHAT IT DOES?

```
for x in range(5, 10):
    print(x)
```

>>>

GUESS WHAT IT DOES?

```
for x in range(5, 100, 5):
    print(x)
```

>>>>

GUESS WHAT IT DOES?

```
f=open('words.txt','r')
j=open('new-words.txt','w')
j.write(f.read())
f.close()
j.close()
```

GUESS WHAT IT DOES?

```
def factorial(n):
    fact=1
    for x in range(n, 1, 1):
        fact*=x
    return fact

print factorial(4)
```

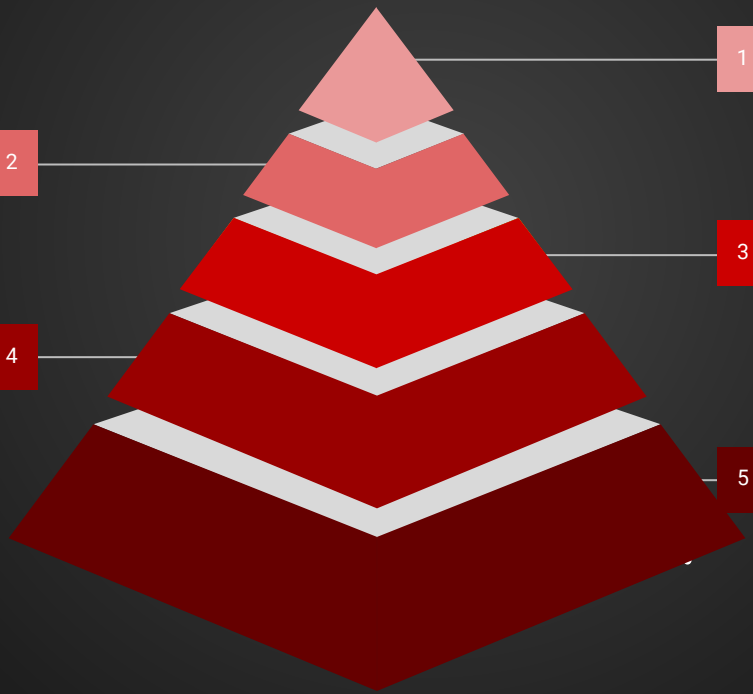
GUESS WHAT IT DOES?

```
squares=[]
for x in range(1,100):
    squares.append(x*x)
    print (squares)
```

PYTHON STACK

How Python is installed in your Machine

PYTHON STACK



1 **Python Program**
Eg.: HelloWorld.py

2 **Python**
Python.exe or python binary

3 **Python Byte Code Interpreter (VM)**
Byte Code

4 **Operating System**
Windows, macOS, Linux

5 **Hardware**
I386, AMD-64, RISC

PYTHON CODING STYLE

NO TABS

Use 4-space indentation, and no tabs

CODE WRAP

Wrap lines so that they don't exceed 79 characters

BLANK LINES

Use blank lines to separate functions and classes, and larger blocks of code inside functions

COMMENTS

When possible, put comments on a line of their own

DOCSTRINGS

Use docstrings

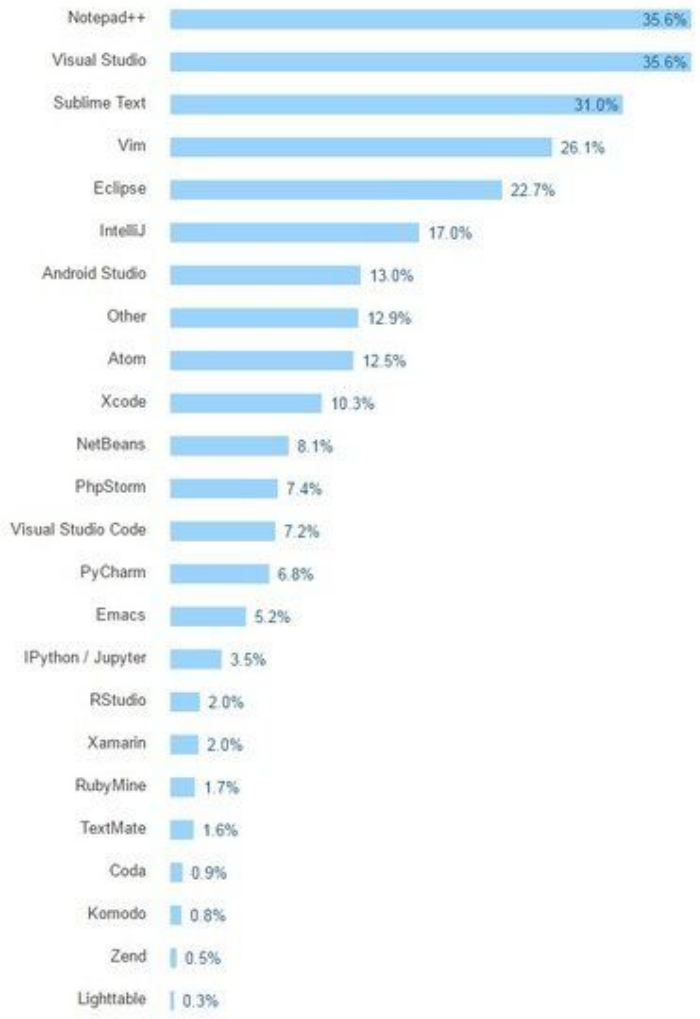
OPERATORS

Use spaces around operators and after commas, but not directly inside bracketing constructs:

```
a = f(1, 2) + g(3, 4)
```

NAMING CONVENTIONS

- Variables:
  - b (single lowercase letter)
  - B (single uppercase letter)
  - lowercase
- Lists. Dictionaries:
  - UPPERCASE
  - UPPER_CASE_WITH_UNDERSCORES
- Classes:
  - CapitalizedWords (or CapWords, or CamelCase)
- Functions:
  - lower_case_with_underscores
  - mixedCase



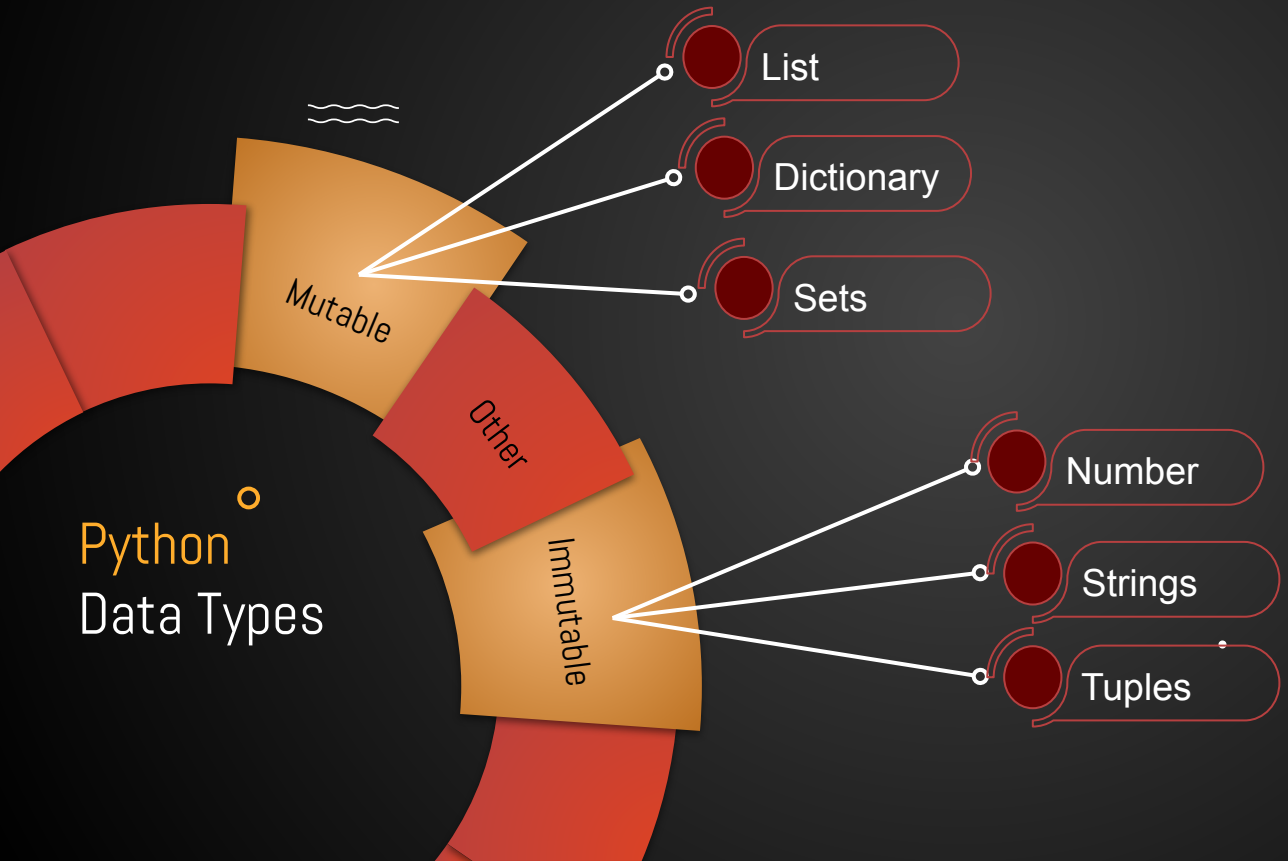
TOP PYTHON IDES


TOP
PYTHON
IDES
PC
Thonny
SPYDER
Vim

DATA TYPES

Different Data Types in Python

DATA TYPES



Python
Data Types

MUTABLE DATA TYPES

```
>>>a=[1,5,8,3,5,9,7]
>>>print(type(a))
>>>print(a[0])
>>>a[1] = 2
>>>a.append(9)
>>>print(a)
>>>a.insert(1,200)
```

LISTS

```
>>>d={'H':'Hydrogen',
     'O':'Oxygen'}
>>>print(type(d))
>>>print(d.keys())
>>>print(d.values())
>>>print(d.items())
>>>print(d['H'])
>>>print(d.get('H'))
```

DICTIONARY

```
>>>s={'a','b','c'} or
>>>s=set(['a','b','c'])
>>>print(type(s))
>>>s.add('d')
>>>s1=set(['b','y','z'])
>>>s2=s.union(s1)
>>>s3=s.intersection(s1)
>>>s4=s.difference(s1)
```

SETS

IMMUTABLE DATA TYPES

IMMUTABLE DATA TYPES

```
>>>a,b,c=10,3,3.0
>>>print (a/b)
>>>print (a/c)
>>>print (a//c)
>>>print (a%b)
>>>print (a**b)
>>>print (abs(b-a))
```

NUMBERS

```
>>>print( "Hello World")
>>>name=input("Please
enter your name:")
>>>print( "Hello", name)
>>>print(len(name))
>>>print(type(name))
```

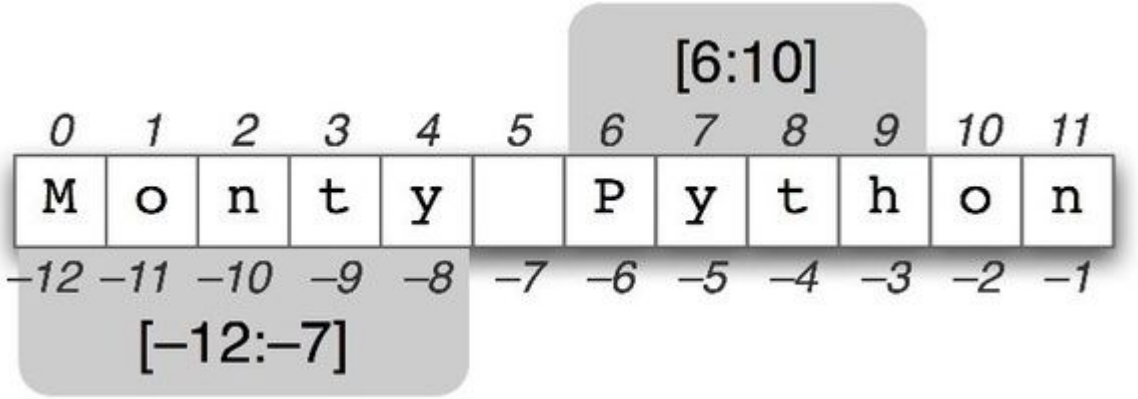
STRINGS

A Tuple is a read-only List

```
>>>a=(1,5,8,3,5,9,7)
>>>print type(a)
>>>print a[0]
>>>a[2]=45
>>>a.append(9)
>>>print sum(a)
>>>print min(a)
```

TUPLES

STRING INDEXING



Strings & String Methods

```
name = “Monty Python”
print(name[0])
print(name[1])
print(name[0:3])
print(name[3:])
print(name[:4])
print(name[0:10:2])
print(name[::1])
print(name[::-1])
print(name[:])

a='hello'
print(a + a)
print(a + “hi”)
print(a * 5)
```

```
capitalize()
casefold()
center()
count()
encode()
endswith()
expandtabs()
find()
format()
format_map()
index()
isalnum()
isalpha()
isdecimal()
isdigit()
isidentifier()
title()
```

```
islower()
isnumeric()
isprintable()
isspace()
istitle()
isupper()
join()
ljust()
lower()
lstrip()
maketrans()
partition()
replace()
rsplit()
rstrip()
rindex()
strip()
```

```
swapcase()
split()
splitlines()
startswith()
translate()
upper()
zfill()
```

OTHER DATA TYPES

Boolean:

Objects of Boolean type has one of two values: `True` or `False`

OPERATOR PRECEDENCE

>>>>

Level	Operator	Description
18	()	Grouping
17	f()	Function call
16	[index:index]	Slicing
15	[]	Array Subscription
14	**	Exponential
13	~	Bitwise NOT
12	+ -	Unary plus / minus
11	* / %	Multiplication Division Modulo
10	+ -	Addition / Subtraction
9	<< >>	Bitwise Left Shift Bitwise Right Shift
8	&	Bitwise AND
7	^	Bitwise XOR
6	|	Bitwise OR
5	in , not in, is , is not <, <=, >, >= == !=	Membership Relational Equality Inequality
4	not	Boolean NOT
3	and	Boolean AND
2	or	Boolean OR
1	lambda	Lambda Expression

```
>>>
print("Hello World")
print("Hello", "India")
print( "{} is a {}".format('Water','Liquid'))
>>>
print("Avg height of Indian men is %f" % 1.62)
print("Avg height of Indian men is %d" % 1.62)
print("Avg height of Indian men is %s" % 1.62)
```

PRINTING

>>>>

```
a=None
b=True
c=45
d=56.3
e="hello"
f=[]
g=3,4,5
h={"a":"apple","b":"Bat"}
print(float(c))
```

Exercise:

- type cast a `str` into `list`
- type cast a `dict` into `list`
- type cast a `tuple` into `list`
- type cast a `list` into `set`
- type cast a `list` into `dict`

>>>>

ADVANCED ASSIGNMENTS

```
#unpacking
a=b=c=2
print(a , b, c)
a, b, c = 2, 3, 4
print(a, b, c)
a=[4, 5, 6]
x, y, z = a
print(x, y, z)
x, y = a
w, x, y, z = a
```

```
#Swapping

a = 10
b = 5
a, b = b, a
print(a)
print(b)
```

LOOPS & CONDITIONALS

Iterate & Control as you like it

LOOPS & CONDITIONALS

If:

```
aura = 2
if aura < 2.5:
     print ("you are not healthy")
```

If-else:

```
aura = 2
if aura <= 1:
     print( "You're dead!" )
else:
     print( "You're alive!" )
```

If-elif-else:

```
aura = 2
if aura <= 1:
     print ("You're dead!")
elif aura > 3:
     print ("You're spiritual!")
else:
     print ("You're alive!")
```

---

for:

```
weapons = ["Arrow", "Mace",
"Spear", "Sword"]
for x in weapons:
     print(x)
for weapon in weapons:
     print(weapon)
     print len(weapon)
```

for with range:

```
for x in range(5):
     print(x)
for x in range(5,15):
     print(x)
for x in range(0,20,3):
     print(x)
```

while:

```
a=0
while a<10:
     print(a)
     a=a+1
# a++ and a-- are not valid
# you may use a+=1 or a-=1
```

FUNCTIONS

Want to reuse *that* particular piece of code?

FUNCTIONS

Simple:

```
def hello():
   print("I am a simple function")

hello()
```

With Arguments:

```
def add(x,y):
      return x+y

c=add(4,5)
print(c)
print(add(5,6))
```

Arguments with Default Values:

```
def add(x,y=10):
      return x+y

c=add(4,5)
d=add(5)
print(c,d)
print(add(5,6),add(8))
```

Keyword Arguments:

```
def wish(name,age):
     print("Hello {} you are {}
years old".format(name,age))

wish('India',67)
wish(67,'India')
wish(age=67,name='India')
```

`global` Keyword:

```
age=16
def grow():
   global age
   print(age)
   age=age+1
   print(age)
grow()
print(age)
```

Returning Multiple Values:

```
def sumdiff(a,b):
   return a+b,abs(a-b)

print(type(sumdiff(4,9)))

mysum,mydiff=sumdiff(4,9)
print(mysum,mydiff)
```

FUNCTIONS

Variable Length Arguments:

```
def average(*num):
    print(type(num))
    print(num)
    print(float(sum(num))/len(num))

average(3,4)
average(3,4,8)
average(3,4,8,90,4.5,5.3,7.8)
#*args will pack all the arguments
#into a tuple called args
```

Variable Length Arguments:

```
def average(a,b,*num):
    print(type(num))
    print(num)
    print((a+b+sum(num))/(2+len(num)))

average(3,4)
average(3,4,8)
average(3,4,8,90,4.5,5.3,7.8)
average()
```

Variable Length Keyword Arguments:

```
def polygon(**kwds):
    print(type(kwds))
    print(kwds)
polygon(width=10,length=20)
polygon(width=10,length=20,h
eight=5)
polygon(width=10,length=20,h
eight=5,units='cm')

#**kwds will pack all the
#arguments into a dict
#called kwds
```

Handle Any Type Arguments:

```
def polygon(a,b,c,*sides,**options):
    print(type(options))
    print(type(sides))
    print(a,b,c)

polygon(8,7,6,4,2,8,units='cm',compute='area')
```

Lambda Functions:

```
add=lambda x,y:x+y

print(add(4,5))
```

```
#these are anonymous functions
#they work like inline functions
```

FILE I/O

Read/Write Files.
Just like that!

Write to a file:

```
f=open('sample.txt','w')
f.write("this is a line\n")
f.write("this is a new line")
f.close()
```

Read file line by line:

```
f=open('sample.txt','r')
print(f.readline())
print(f.readline())
f.close()
```

Read file contents at once:

```
f=open('sample.txt','r')
print(f.read())
f.close()
```

Read all lines into a list:

```
f=open('sample.txt','r')
lines= f.readlines()
print(lines)
f.close()
```

Read file char by char:

```
f=open('sample.txt','r')
print(f.read(1))
print(f.read(1))
f.close()
```

Append to an existing file:

```
f=open('sample.txt','a')
f.write("\nthis is a new line")
f.close()
```

MODULES

Speak about reusability.
Many times

MODULES

Find what all modules are installed:

```
help("modules")
```

Importing a Module:

```
import this
math.sin(90)
```

Import only a function from Module:

```
from random import randint
randint(3,30)
from math import sin,cos,tan
cos(90)
sin(90)
```

Aliasing a module:

```
import random as rd
rd.random()
```

```
from math import factorial as f
f(6)
```

Know what functions are available in a Module:

```
import math
dir(math)
```

Get help on a Function:

```
import math
help(math.sin)
```

MODULES

>>>>

pypi.org

Modules are installed in:

- `lib` folder in python installation directory (Windows)
  `/usr/lib/python` on linux

>>>>

3rd Party Modules are installed in:

`lib/site-packages`

pip

PYMOTW

OS MODULE

- OS module in python provides functions for interacting with the operating system

List of the contents of a directory

```
# os_listdir.py

import os
import sys
print(sorted(os.listdir(sys.argv[1])))
```

Test access rights a process has for a file

```
# os_access.py
import os

print('Testing:', __file__)
print('Exists:', os.access(__file__, os.F_OK))
print('Readable:', os.access(__file__,
os.R_OK))
print('Writable:', os.access(__file__,
os.W_OK))
print('Executable:', os.access(__file__,
os.X_OK))
```

```
os.abort
os.access
os.altsep
os.chdir
os.chmod
os.close
os.closerange
os.cpu_count
os.curdir
os.defpath
os.device_encoding
os.devnull
os.environ
os.error
os.execv
os.fdopen
os.fsdecode
os.fsencode
os.fspath
os.fstat
os.fsync
os.get_exec_path
os.getcwd
os.getcwdb
os.getenv
os.getlogin
os.getpid
os.getppid
os.kill
os.linesep
os.link
os.listdir
os.lseek
os.lstat
os.makedirs
os.mkdir
os.name
os.open
os.path
os.pathsep
os.pipe
os.read
os.remove
os.removedirs
os.rename
os.renames
os.replace
os.rmdir
os.scandir
os.sep
os.set_handle_inheritable
os.set_inheritable
os.spawnl
os.spawnle
os.startfile
os.stat
os.stat_result
os.statvfs_result
os.strerror
os.truncate
os.umask
os.uname_result
os.unlink
os.walk
os.write
```

SYS MODULE

- `sys` module allows you to use `stdin()` and `stdout()`, as well as `stderr()`

Build-time Version Information

```
# sys_version_values.py

import sys

print('Version info:')
print()
print('sys.version      =', repr(sys.version))
print('sys.version_info =', sys.version_info)
print('sys.hexversion   =', hex(sys.hexversion))
print('sys.api_version  =', sys.api_version)
```

Operating System Platform

```
# sys_platform.py

import sys

print('This interpreter was built for:', sys.platform)
```

```
sys.builtin_module_names
sys.builtin_module_names
sys.byteorder
sys.copyright
sys.displayhook
sys.dllhandle
sys.dont_write_bytecode
sys.exc_info
sys.excepthook
sys.exec_prefix
sys.executable
sys.exit
sys.flags
sys.getallocatedblocks
sys.getcheckinterval
sys.getdefaultencoding
sys.getfilesystemencodeerrors
sys.getfilesystemencoding
sys.getprofile
sys.getrecursionlimit
sys.getrefcount
sys.getsizeof
```

```
sys.getwindowsversion
sys.implementation
sys.last_traceback
sys.last_type
sys.last_value
sys.maxsize
sys.maxunicode
sys.setprofile
sys.setrecursionlimit
sys.setswitchinterval
sys.settrace
sys.stderr
sys.stdin
sys.stdout
sys.thread_info
sys.version
sys.version_info
sys.warnoptions
sys.winver
```

re MODULE

- RegEx, or Regular Expression, is a sequence of characters that forms a search pattern

```
import re
#find any number anywhere in the string
print(re.findall("\d+","3 pens cost 20 rupees and 50 paise"))
```

```
#find any number at the beginning of the string
print(re.findall("^\d+","3 pens cost 20 rupees and 50 paise"))
```

```
#find any number at the end of the string
print(re.findall("\d+$","3 pens cost 20 rupees and 50"))
```

```
#Split with multiple separators
print(re.split("[,:-]","Ramesh,223-ramesh@mongofactory.com:male"))
```

OTHER MODULES >>>>

`shutil` Module

Use for daily file/directory management tasks

```
import shutil
#copies data.db to archive.db
shutil.copyfile('data.db', 'archive.db')
#move(source, destination)
shutil.move('/build/executables', 'installdir')
```

File Wildcards

`glob` *module provides a function for making file lists from directory wildcard searches*

```
import glob
glob.glob('*.py')
```

Mathematics

The `math` *module gives access to the underlying C library functions*

for floating point math

```
import math
math.cos(math.pi / 4.0)
math.log(1024, 2)
```

Random

The `random` *module provides tools for making random selections*

```
import random
random.choice(['apple', 'pear', 'banana'])
# sampling without replacement
random.sample(xrange(100), 10)
# random float
random.random()
# random integer chosen from range(6)
random.randrange(6)
```

`datetime` Module

The `datetime` *module supplies classes for manipulating dates and times in both simple and complex ways.*
The module also supports objects that are timezone aware

```
# dates are easily constructed and formatted
from datetime import date
now = date.today()
now
now.strftime("%m-%d-%y. %d%b %Y is a %A on the %dday of %B.")

# dates support calendar arithmetic
birthday = date(1964, 7, 31)
age = now -birthday
age.days
```

THANKS!

For bearing me through this presentation