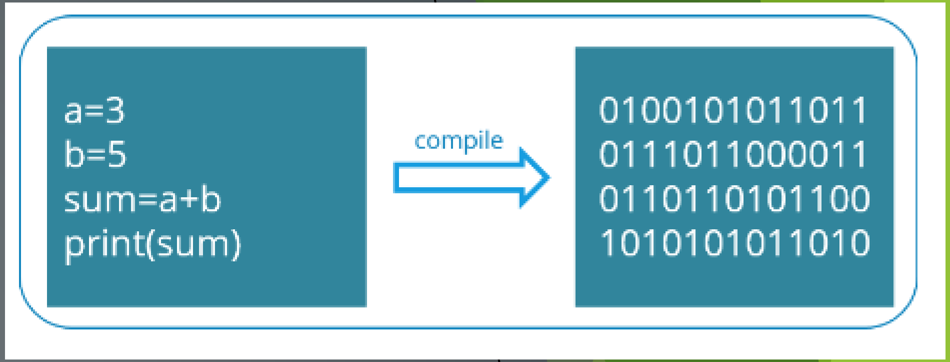
Python

What is Python?

Python is a high-level programming language which is:

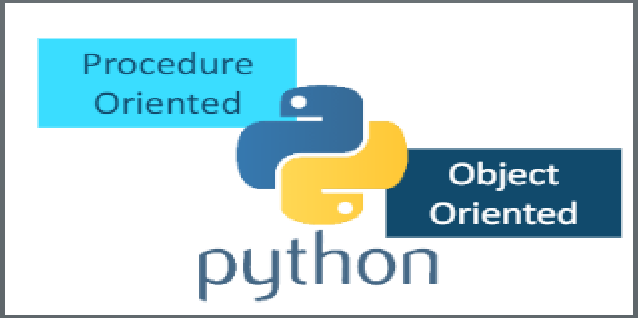
- Interpreted: Python is processed at runtime by the interpreter.
- Interactive: You can use a Python prompt and interact with the interpreter directly to write your programs.
- Object-Oriented: Python supports Object-Oriented technique of programming.
- Beginner’s Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications.

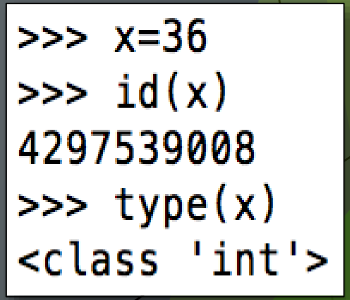
Why Python?



```
a=3
b=5
sum=a+b
print(sum)
```

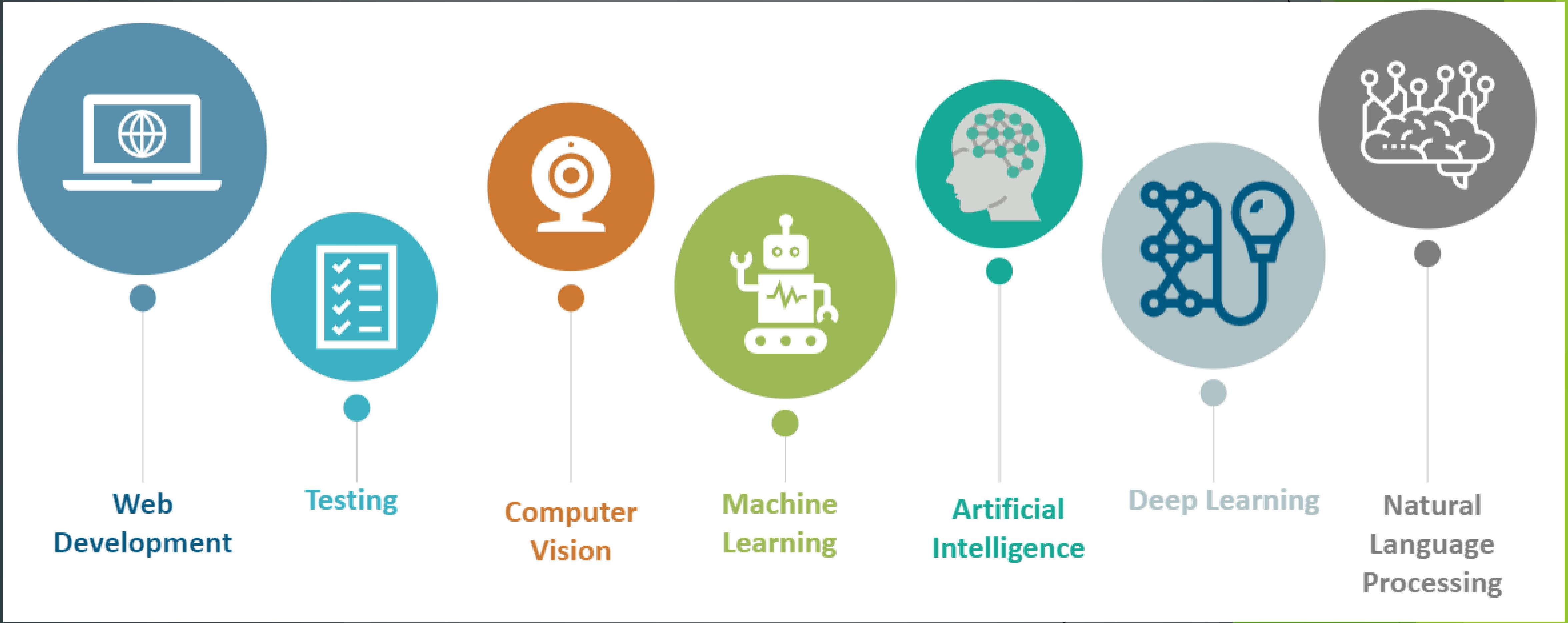
```
0100101011011
0111011000011
0110110101100
1010101011010
```





```
>>> x=36
>>> id(x)
4297539008
>>> type(x)
<class 'int'>
```

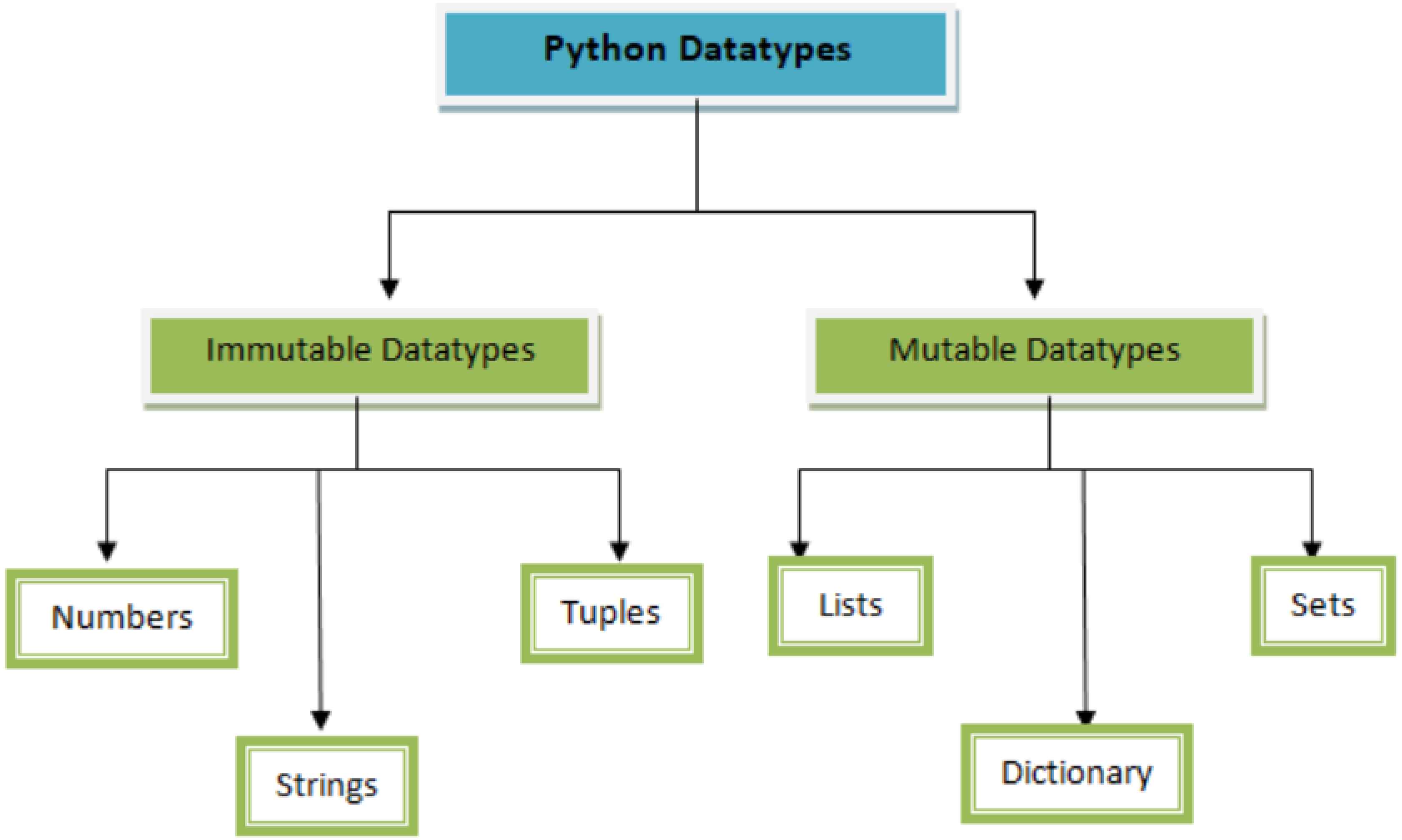
Applications



Variables

- Python is dynamically typed. You do not need to declare variables!
- The declaration happens automatically when you assign a value to a variable.
- Variables can change type, simply by assigning them a new value of a different type.
- Python allows you to assign a single value to several variables simultaneously.
- You can also assign multiple objects to multiple variables.

Python Data Types

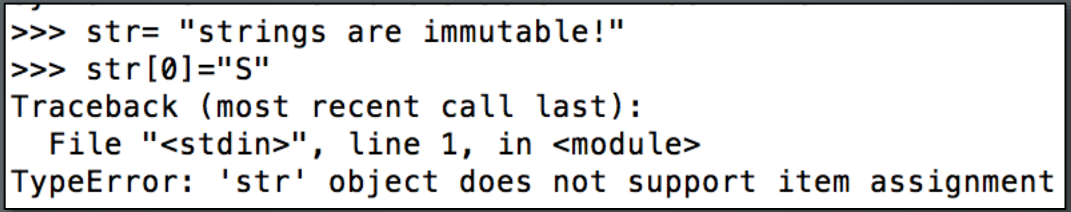

Python Datatypes
Immutable Datatypes
Mutable Datatypes
Numbers
Strings
Tuples
Lists
Dictionary
Sets

Numbers

- Numbers are Immutable objects in Python that cannot change their values.
- There are three built-in data types for numbers in Python3:
  - Integer (int)
  - Floating-point numbers (float)
  - Complex numbers: <*real part*> + <*imaginary part*>j (not used much in Python programming)

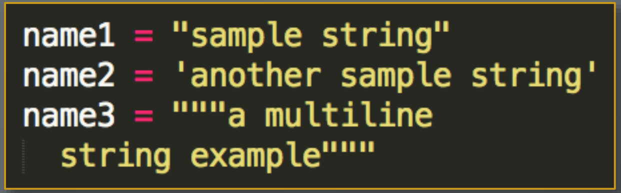
Strings

- Python Strings are Immutable objects that cannot change their values.

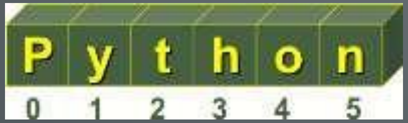


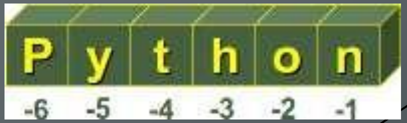
```
>>> str= "strings are immutable!"
>>> str[0]="S"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

- Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals.



```
name1 = "sample string"
name2 = 'another sample string'
name3 = """a multiline
    string example"""
```





Strings

- **Common String Operators**

  Assume string variable a holds 'Hello' and variable b holds 'Python'

Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	a + b
*	Repetition - Creates new strings, concatenating multiple copies of the same string	a*2
[ ]	Slice - Gives the character from the given index	a[1] a[-1]
[:]	Range Slice - Gives the characters from the given range	a[1:4]
in	Membership - Returns true if a character exists in the given string	'H' in a

Strings

- **Common String**

Method	Description
str.count(sub, beg= 0,end=len(str))	Counts how many times sub occurs in string or in a substring of string if starting index beg and ending index end are given.
str.isalpha()	Returns True if string has at least 1 character and all characters are alphanumeric and False otherwise.
str.isdigit()	Returns True if string contains only digits and False otherwise.
str.lower()	Converts all uppercase letters in string to lowercase.
str.upper()	Converts lowercase letters in string to uppercase.
str.replace(old, new)	Replaces all occurrences of old in string with new.
str.split(str=‘ ’)	Splits string according to delimiter str (space if not provided) and returns list of substrings.
str.strip()	Removes all leading and trailing whitespace of string.
str.title()	Returns "titlecased" version of string.

- **Common String Functions**

str(x) :to convert x to a string

len(string):gives the total length of the string

Lists

- A list in Python is an ordered group of items or elements, and these list elements don't have to be of the same type.
- Python Lists are mutable objects that can change their values.
- A list contains items separated by commas and enclosed within square brackets.
- List indexes like strings starting at 0 in the beginning of the list and working their way from -1 at the end.
- Similar to strings, Lists operations include slicing ([ ] and [:]) , concatenation (+),
  repetition (*),

Lists

- Lists can have sublists as elements and these sublists may contain other sublists as well.

- Common List Functions

Function	Description
cmp(list1, list2)	Compares elements of both lists.
len(list)	Gives the total length of the list.
max(list)	Returns item from the list with max value.
min(list)	Returns item from the list with min value.
list(tuple)	Converts a tuple into list.

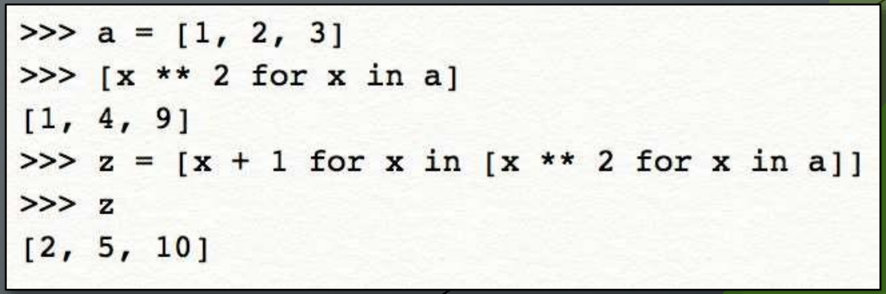
Lists

- **Common List Methods**

Method	Description
list.append(obj)	Appends object obj to list
list.insert(index, obj)	Inserts object obj into list at offset index
list.count(obj)	Returns count of how many times obj occurs in list
list.index(obj)	Returns the lowest index in list that obj appears
list.remove(obj)	Removes object obj from list
list.reverse()	Reverses objects of list in place
list.sort()	Sorts objects of list in place

- **List Comprehensions**

  Each list comprehension consists of an expression followed by a for clause.



```
>>> a = [1, 2, 3]
>>> [x ** 2 for x in a]
[1, 4, 9]
>>> z = [x + 1 for x in [x ** 2 for x in a]]
>>> z
[2, 5, 10]
```

Tuples

- Python Tuples are Immutable objects that cannot be changed once they have been created.
- A tuple contains items separated by *commas* and enclosed in *parentheses* instead of square brackets.
- Tuples are faster than lists and protect your data against accidental changes to these data.
- The rules for tuple indices are the same as for lists and they have the same operations functions as well.
- To write a tuple containing a single value, you have to include a *comma*, even though there is only one value. e.g. t = (3,)

Dictionary

- Python's dictionaries are kind of hash table type which consist of key-value pairs
  of unordered elements.
  - Keys : must be immutable data types ,usually numbers or strings.
  - Values : can be any arbitrary Python object.
- Python Dictionaries are mutable objects that can change their values.
- A dictionary is enclosed by *curly braces* ({ }), the items are separated by *commas*, and each key is separated from its value by a *colon* (:).
- Dictionary's values can be assigned and accessed using square braces ([]) with a key to obtain its value.

Dictionary

Method	Description
dict.keys()	Returns list of dict's keys
dict.values()	Returns list of dict's values
dict.items()	Returns a list of dict's (key, value) tuple pairs
dict.get(key, default=None)	For key, returns value or default if key not in dict
dict.has_key(key)	Returns True if key in dict, False otherwise
dict.update(dict2)	Adds dict2's key-values pairs to dict
dict.clear()	Removes all elements of dict

Conditions

- In Python, True and False are Boolean objects of class 'bool' and they are immutable.
- Python assumes any non-zero and non-null values as True, otherwise it is False value.
- Python *does not* provide switch or case statements as in other languages.

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

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

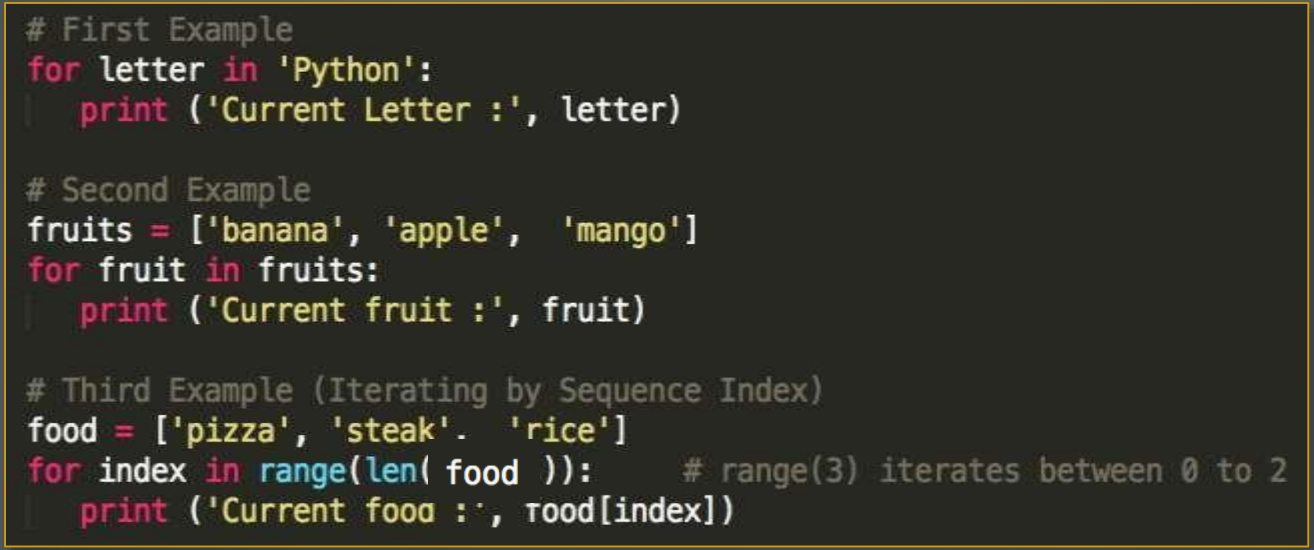
```
if expression1:
    statement(s)
elif expression2:
    statement(s)
elif expression3:
    statement(s)
else:
    statement(s)
```

- Example

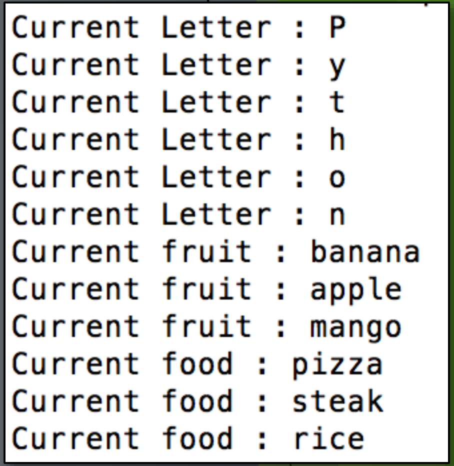
```
x = int(input("Please enter an integer: "))
if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More’)
```

Loops

- **The For Loop**

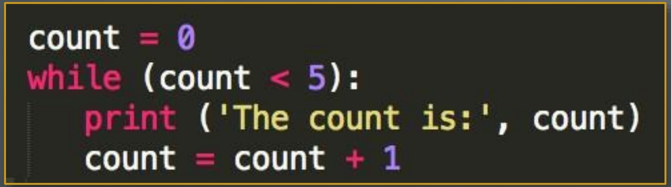


```python
# First Example
for letter in 'Python':
    print ('Current Letter :', letter)

# Second Example
fruits = ['banana', 'apple',  'mango']
for fruit in fruits:
    print ('Current fruit :', fruit)

# Third Example (Iterating by Sequence Index)
food = ['pizza', 'steak',  'rice']
for index in range(len( food )):      # range(3) iterates between 0 to 2
    print ('Current food :', food[index])
```

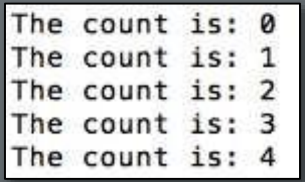


```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
Current food : pizza
Current food : steak
Current food : rice
```

- **The while Loop**



```python
count = 0
while (count < 5):
    print ('The count is:', count)
    count = count + 1
```

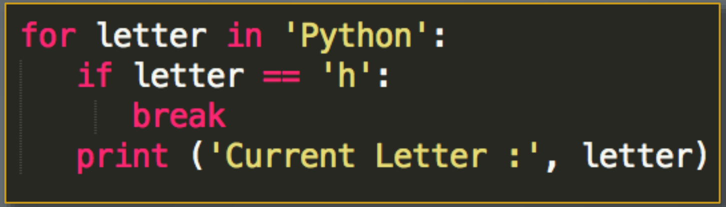


```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
```

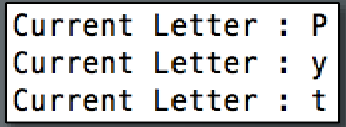
Loops

Loop Control Statements

- break:Terminates the statement and transfersexecution to loop

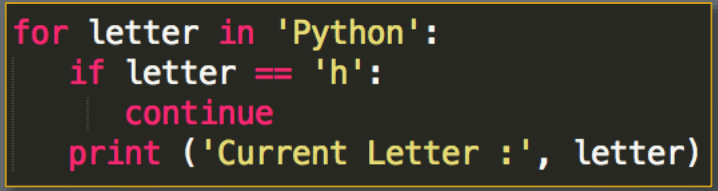


```python
for letter in 'Python':
   if letter == 'h':
      break
   print ('Current Letter :', letter)
```

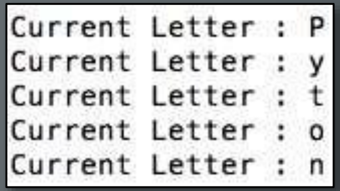


```
Current Letter : P
Current Letter : y
Current Letter : t
```

- continue :Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

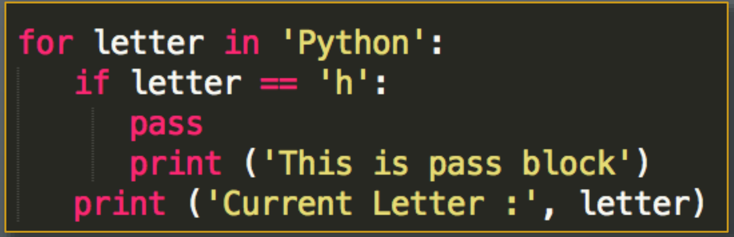


```python
for letter in 'Python':
   if letter == 'h':
      continue
   print ('Current Letter :', letter)
```

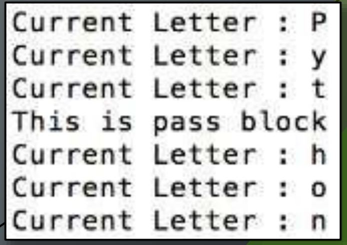


```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
```

- pass:Usedwhena statement is required syntactically but you do not want any command or code to execute.



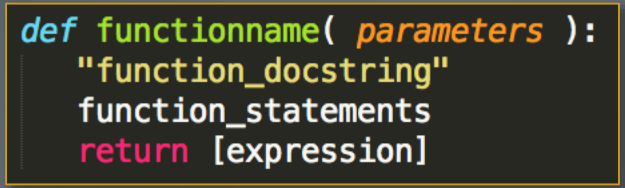
```python
for letter in 'Python':
   if letter == 'h':
      pass
      print ('This is pass block')
   print ('Current Letter :', letter)
```



```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
```

Functions

- **Function Syntax**

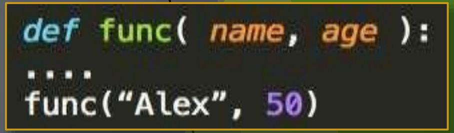


```
def functionname( parameters ):
   "function_docstring"
   function_statements
   return [expression]
```

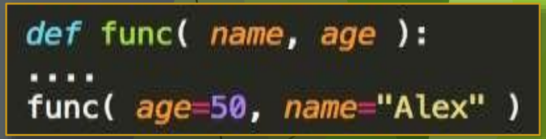
- **Function Arguments**

You can call a function by using any of the following types of arguments:

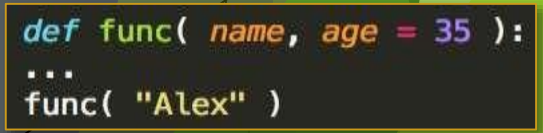
- **Required arguments: the arguments passed to the function in correct positional order.**
- **Keyword arguments: the function call identifies the arguments by the parameter names.**
- **Default arguments: the argument has a default value in the function declaration used when the value is not provided in the function call.**



```
def func( name, age ):
....
func("Alex", 50)
```



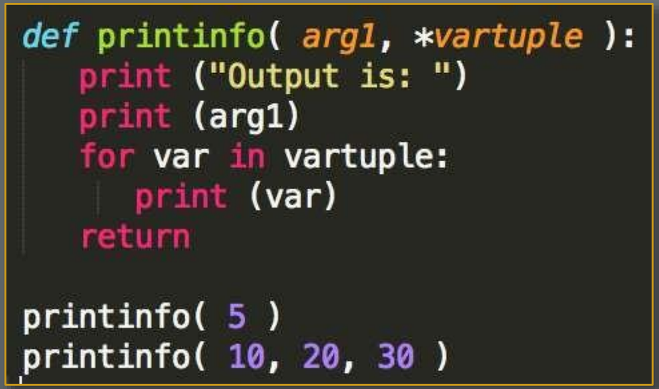
```
def func( name, age ):
....
func( age=50, name="Alex" )
```



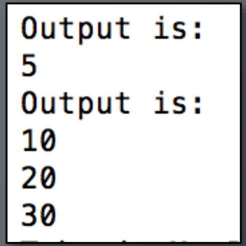
```
def func( name, age = 35 ):
...
func( "Alex" )
```

Functions

- **Variable-length arguments: This used when you need to process unspecified additional arguments. An asterisk (*) is placed before the variable name in the function declaration.**



```
def printinfo( arg1, *vartuple ):
    print ("Output is: ")
    print (arg1)
    for var in vartuple:
        print (var)
    return

printinfo( 5 )
printinfo( 10, 20, 30 )
```

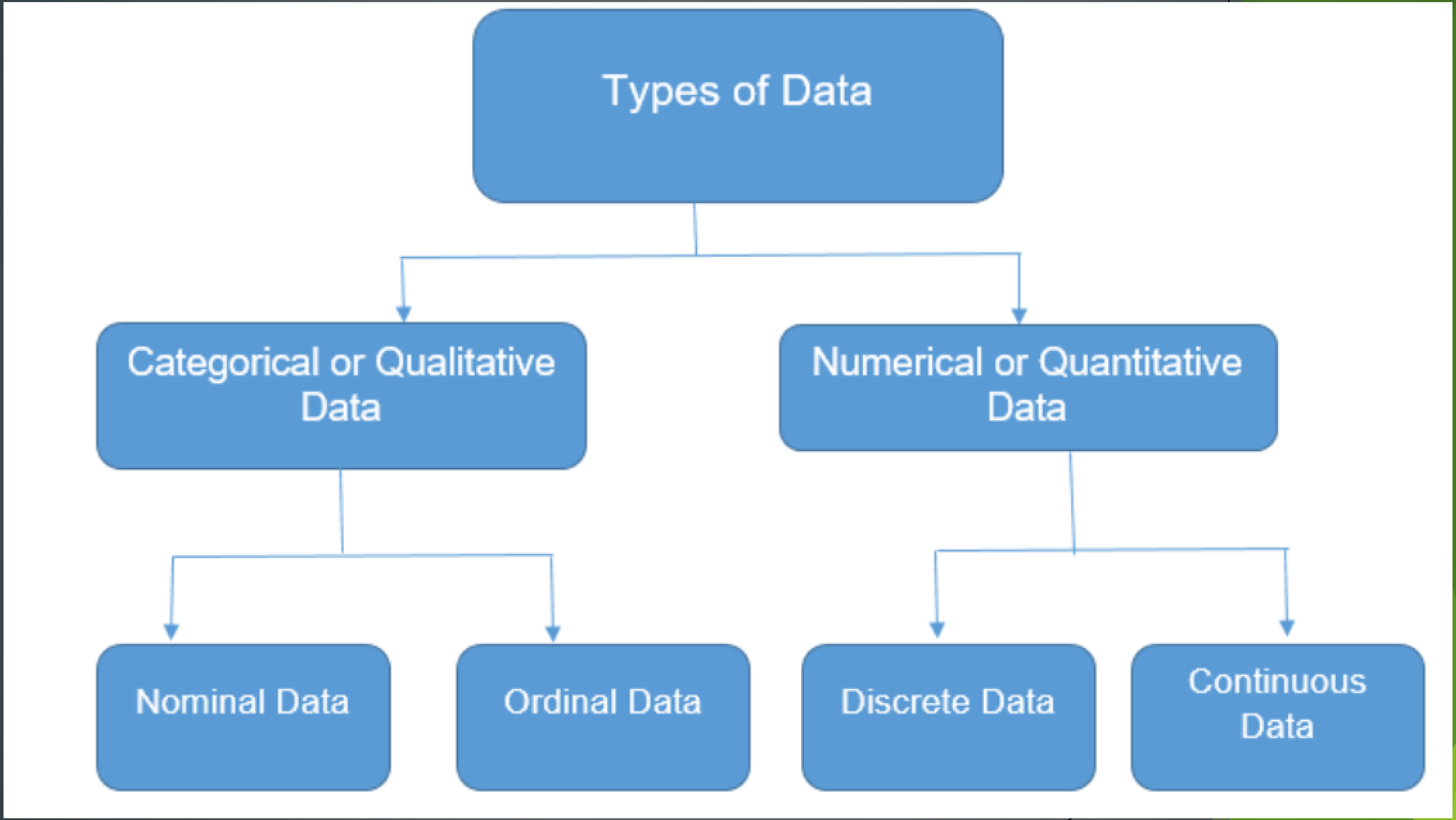


```
Output is:
5
Output is:
10
20
30
```

Who Uses Python?

Organizations Use Python

- **Web Development** :Google, Yahoo
- **Games** :Battlefield 2, Crystal Space
- **Graphics** :Walt Disney Feature Animation, Blender 3D
- **Science** :National Weather Service, NASA, Applied Maths
- **Software Development** :Nokia, Red Hat, IBM
- **Education** :University of California-Irvine, SchoolTool
- **Government** :The USA Central Intelligence Agency (CIA)


Types of Data
Categorical or Qualitative Data
Numerical or Quantitative Data
Nominal Data
Ordinal Data
Discrete Data
Continuous Data