Day 2

Data Types and Structures

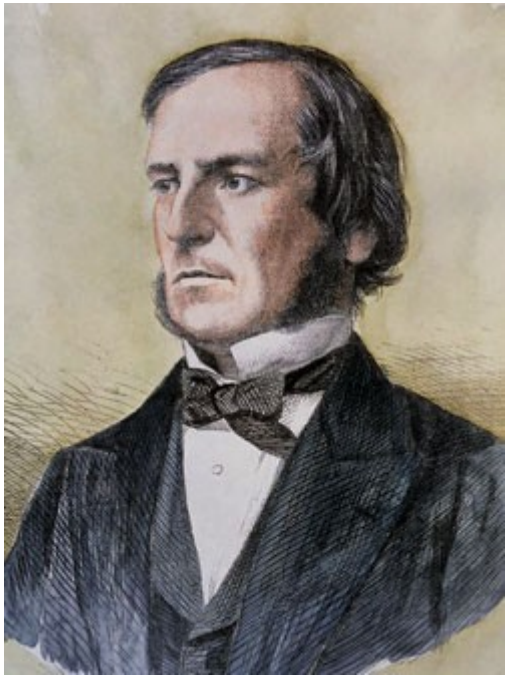
Recap:

- How to use the Bash or Powerpoint shell
- How to start Python Interactive shell
- How to run Python scripts
- Differences between scripts and interactive mode
- Variables
- Numbers
- Strings

Today’s menu

- Booleans: To Be or Not To Be
- If-else statements
- Lists
- Dictionaries
- Sets
- Tuples
- More coding!

Booleans



Essentially truth values : True, False

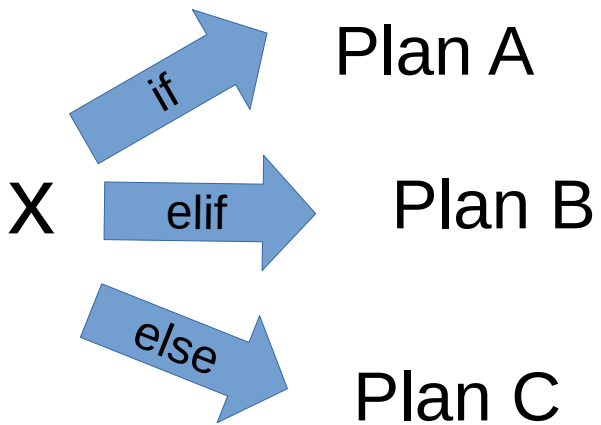
Can calculate as with numbers

Often used to make decisions

If-statements

You can make choices based on data

Often useful in loops




```
if {condition}:
    executed if condition == True
elif {condition2}:
    executed if condition2 ==True
else:
    executed in all other cases
```

Now you!

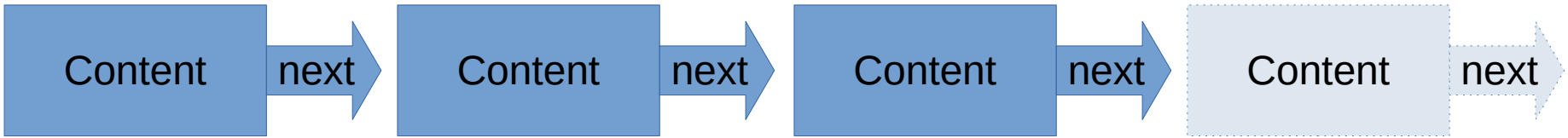
- Test if a number is odd or even:
  - Take an integer as input
  - Test number with if-else condition
  - Print out some message depending on the result

Lists

A collection of items

Linear organization

Anything can be in a list



List = [‘Spam’, ‘Spam’, ‘Bacon’, ‘Eggs’, ‘Spam’]

Common list methods

append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Dictionaries

Word: Explanation

dictionary = {key1: value1, key2:value2, …}
dictionary[key1] = value1

Common dictionary items

clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing the tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Set

- Collection of non-redundant, immutable things
- Unordered
- Can be modified
- Indicated by curly brackets { }

myset = {‘a’, 2, True}

Can a set hold a list?
myset = {[1,2], 4}

Common set operations

Operation	Equivalent	Result
`len(s)`		number of elements in set *s* (cardinality)
`x in s`		test *x* for membership in *s*
`x not in s`		test *x* for non-membership in *s*
`s.issubset(t)`	`s <= t`	test whether every element in *s* is in *t*
`s.issuperset(t)`	`s >= t`	test whether every element in *t* is in *s*
`s.union(t)`	`s | t`	new set with elements from both *s* and *t*
`s.intersection(t)`	`s & t`	new set with elements common to *s* and *t*
`s.difference(t)`	`s - t`	new set with elements in *s* but not in *t*
`s.symmetric_difference(t)`	`s ^ t`	new set with elements in either *s* or *t* but not both
`s.copy()`		new set with a shallow copy of *s*

Tuple

- Immutable collection of things
- Has order
- Basically an unmodifiable list of things, but more memory efficient than a list
- Indicated by smooth brackets ()

```
mytuple = (0, 1, 2)
```