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
- Strings

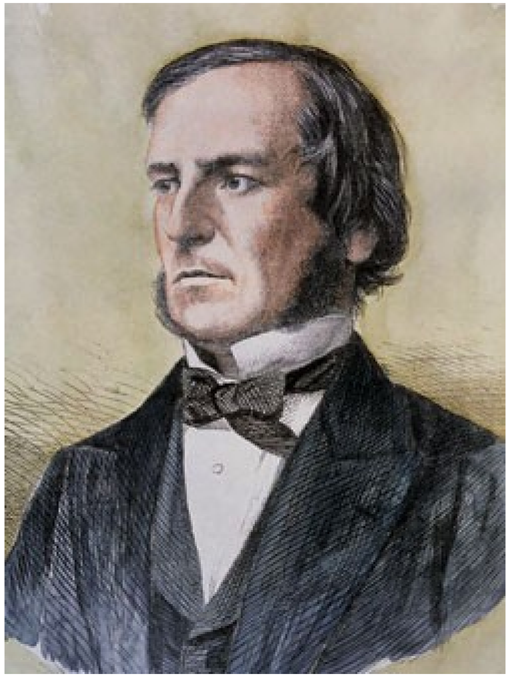
Today’s menu

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

Excursion: Googling things

- Your script fails. The Error shown is TypeError: unsupported operand type(s) for +: 'int' and 'str'. What could this mean? Google the Error and see what pops up.
- You want to capitalize one letter in the middle of the String. How do you do that? If you find a solution: Perfect! In either case: Google it.
- What are good sources for programming advice?
- What are not so great sources?

Booleans



Essentially truth values : True, False

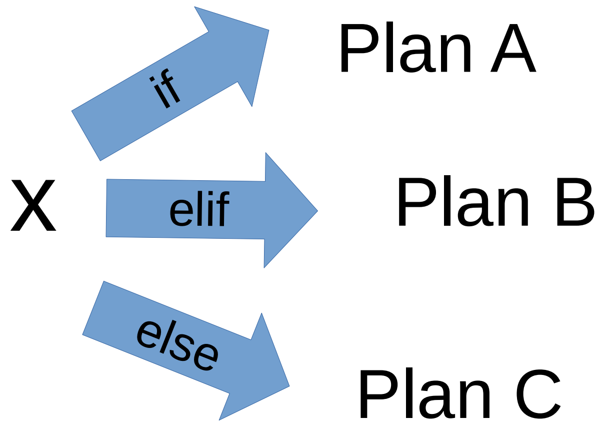
Can calculate as with numbers

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!

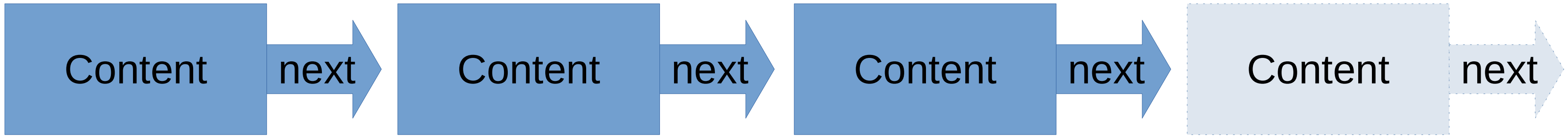
- Test if a number is odd or even:
  - Take an integer as input
  - Test number with if-else condition
  - Print out some message depending on the result

Lists

A collection of items

Linear organization

Anything can be in a list



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

Dictionaries

ex·ist'ence n.
occurrence —ex·ist'ent adj.
ex·is·ten·tial (eg'zis ten'shal)
existentialism
ex'is·ten'tial·ism' n. a philosop
individual existence and holdi
responsible fo
AP

Word: Explanation

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

Loops

Loops repeat an action many times

Rules are the same every repetition

Modification with logic possible

Different kinds of loops

For-loops

While-loops

Indentation

```
Loop
    Loop content part 1
    Loop content part 2
Code continues here
Another loop
    Loop content
More content
```

Indentation

People argue (a lot) over tabs vs spaces

Ultimately it doesn’t matter

Just use things consistently

If you mix tabs and spaces your PC explodes

For loops

Repeats code for ever element of an *iterable*

Strict order: first → last

Very common and very useful construct

Exercises

- Take in a number, print out all the numbers from 0 to your number, using a for loop
- Turtle exercise

Slightly better exercises

- Calculate GC-content in the example file. Do it first using a for loop and if-statement. Then do it using list.count(element).
- Count point mutations in http://rosalind.info/problems/hamm/
- FizzBuzz: Pass a programmer interview question
- Count content of A, T, G, C using a dictionary and ONE loop
- Transcribe DNA into RNA

While loop

Keeps repeating actions while a condition is True

Only stops when condition is False

Inherent danger for endless loops

More specialized use cases than For loop

```
while {condition}:
    loop code here
```

Break conditions

Loop stops when x is False

```
while x:
    {code}
```

While loops need some condition met to stop

These are break conditions

Exercises

- Write a “Guess the number” game.

Break and continue

```
while x:
    if things_broke:
        break
    else:
        do_stuff
```

You can also explicitly cancel loops

break statement stops the loop immediately

Or cancel loop iterations

continue statement skips the current round

```
for element in list:
    if element not in other_list:
        continue
    do_stuff
```

Careful with mutables!

Lists and dictionaries are so called mutables

Mutables are object that can be changed

list1 = list2 creates a pointer, not a copy, to list1

Same goes for dictionaries

List comprehensions

Identical to a for loop

But very compact

```
squares = []
For x in range(1,10):
    x2 = x**2
    squares.append(x2)


squares = [x**2 for x in range(1,10)]
```

Exercises

- Complement a strand of DNA using a list comprehension.
- Transcribe that complemented strain.