Python Kick-off (Extended)

Dr Peet Morris
peet.morris@it.ox.ac.uk

iT Centre Learning

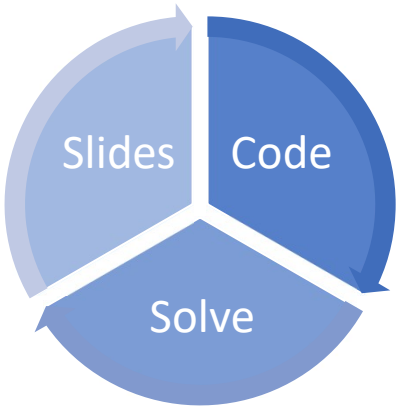
iT services

UNIVERSITY OF OXFORD

What's Covered in this Kick-Off

- Main aim – to provide Python language basics, e.g…
  - *What is Python and how do I use it?*
  - *Options for writing and running Python programs (code editors and the like)*
  - *Python syntax and basic operations on objects*
    - *Iteration & Decision Making*
    - *Python's Fundamental Data Types*
      - *integers, floats, strings, lists, tuples, sets, dictionaries*
  - *The makeup of a Python program*
    - *Scripts, Modules, Packages*
    - *Python's Standard Library*
    - *Installing additional modules with Pip and PyPi*

How it Runs



- Repeated
  - *Slides – me talking, showing you stuff; you asking questions*
    - *Followed by*
  - *Together as a group we'll solve a problem by writing code*
    - *Followed by*
  - *Individual or group problem-solving (a sort of real-time homework)*

- At the end of the Kick-off
  - *You'll get a copy of the slides (inc notes)*
  - *Various post-course 'Challenge Problems' for you to work on to consolidate your knowledge (c/w code snippets and model answers)*

Python

- Created by Guido van Rossum (1991)
- Latest major version 3.10.5 (06/06/2022)
  - *What's New in 3.10*
- Interpreted & Compiled to Byte Code (CPython)
  - *Other special implementations available. For example, a version that runs on microcontrollers.*
- Dynamically typed

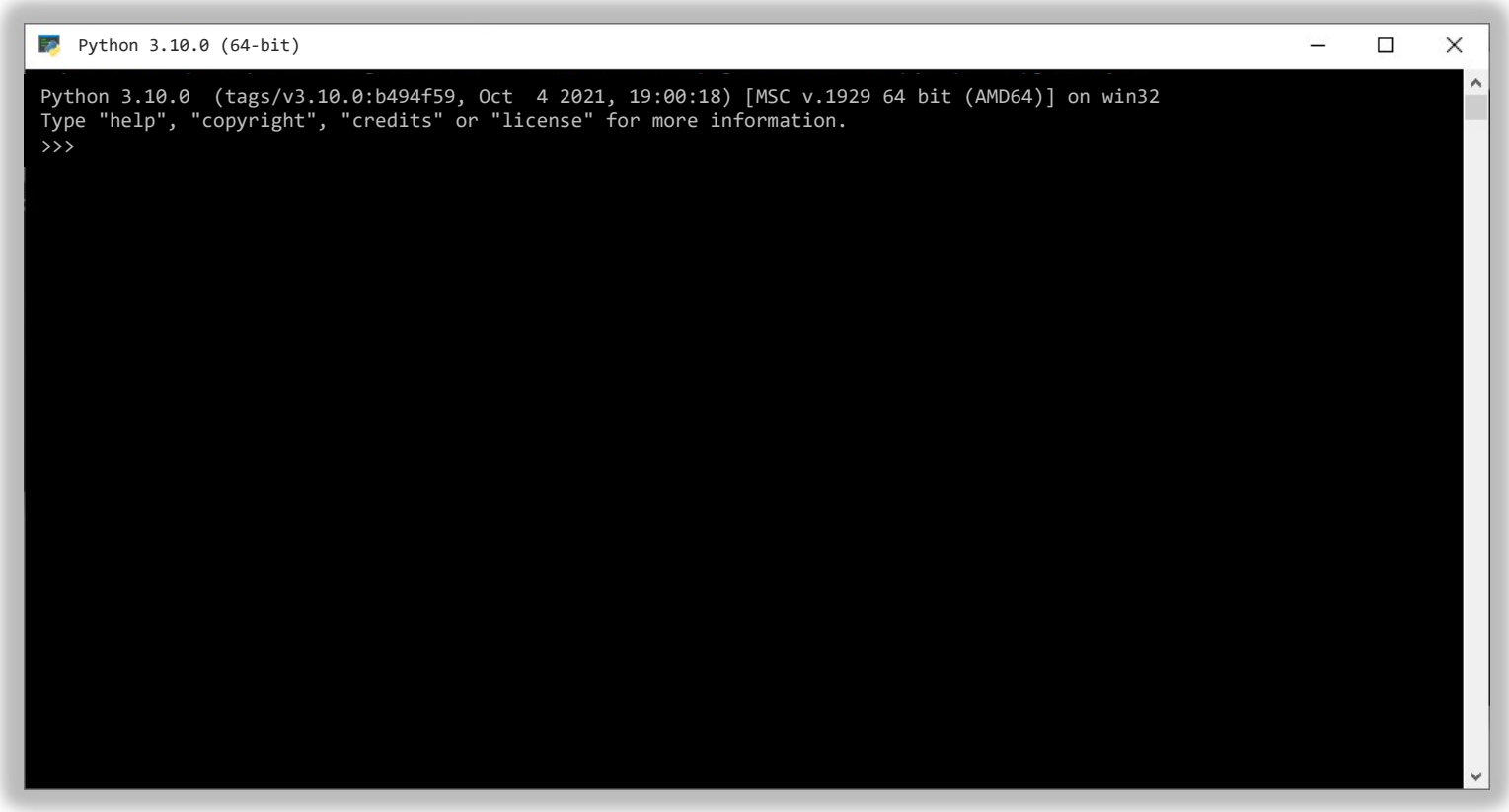
Python – The Library, and other Packages

- Standard Modules/Packages (337)
  - *docs.python.org/3/py-modindex.html*
- Others
  - *pypi.python.org/pypi*
  - ≈270,000 packages/projects
    - *Graphical user interfaces*
    - *Web frameworks*
    - *Multimedia*
    - *Databases*
    - *Networking*
    - *Test frameworks*
    - *Automation*
    - *Web scraping*

- *System administration*
- *Scientific computing*
- *Text and NL processing*
- *Image processing*
- *Machine learning and AI*
- *Games development*
- *Home automation*
- *Financial*
- *...*

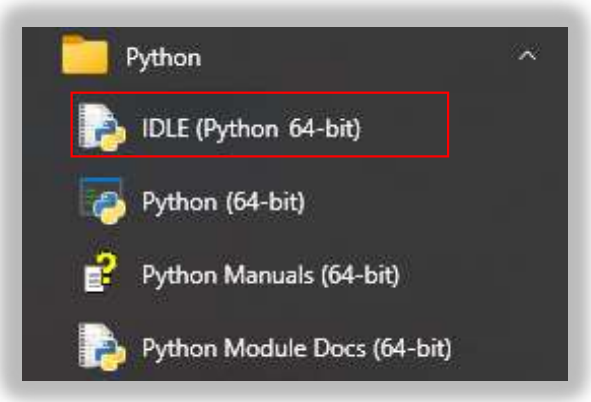
docs.python.org/3.10/library/index.html

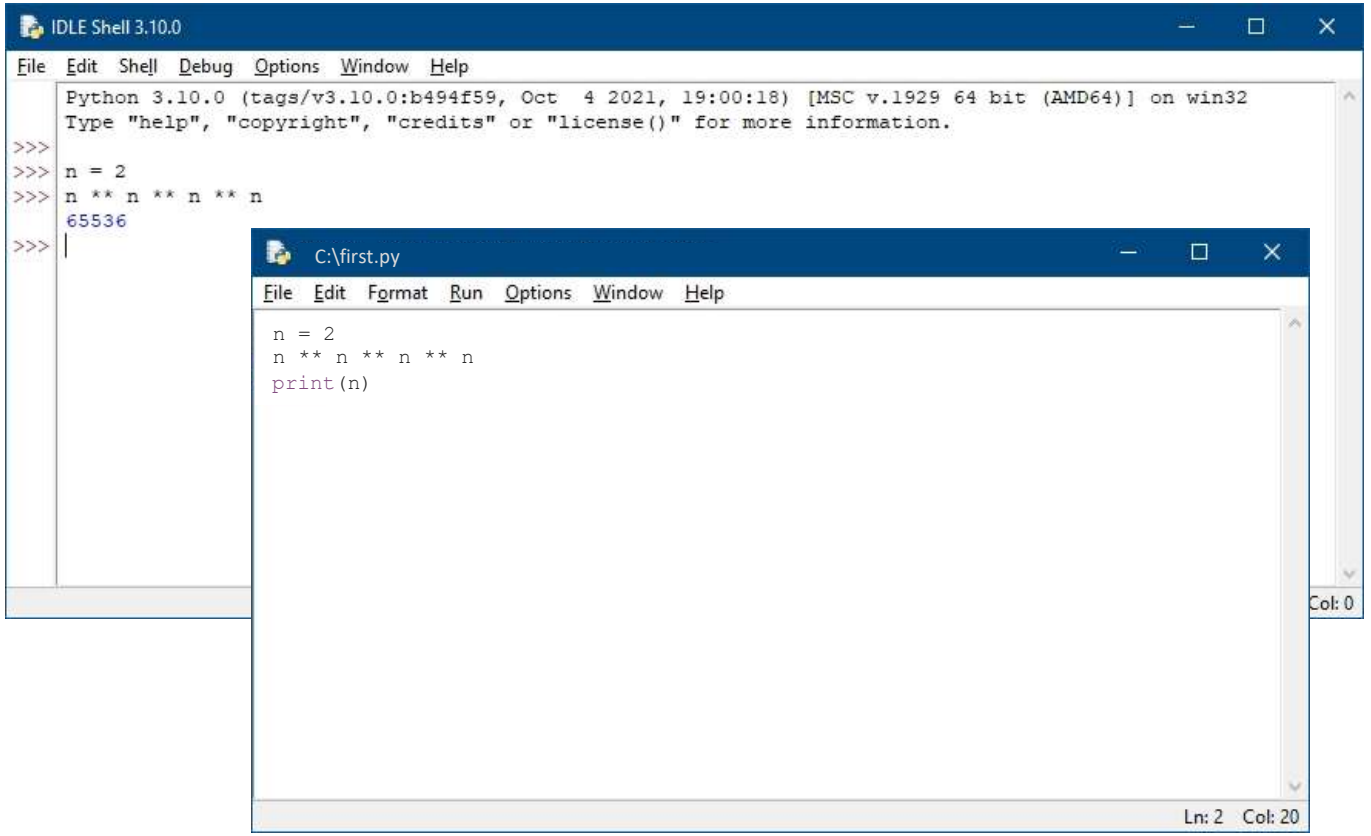
The Python REPL



(Eric) Idle

sourceforge.net/projects/idlex





Installing a Python IDE

wiki.python.org/moin/IntegratedDevelopmentEnvironments

pyscripter

pycharm

code.visualstudio

Set the default configuration

anaconda.com

sublimetext.com

CodeSkulptor3

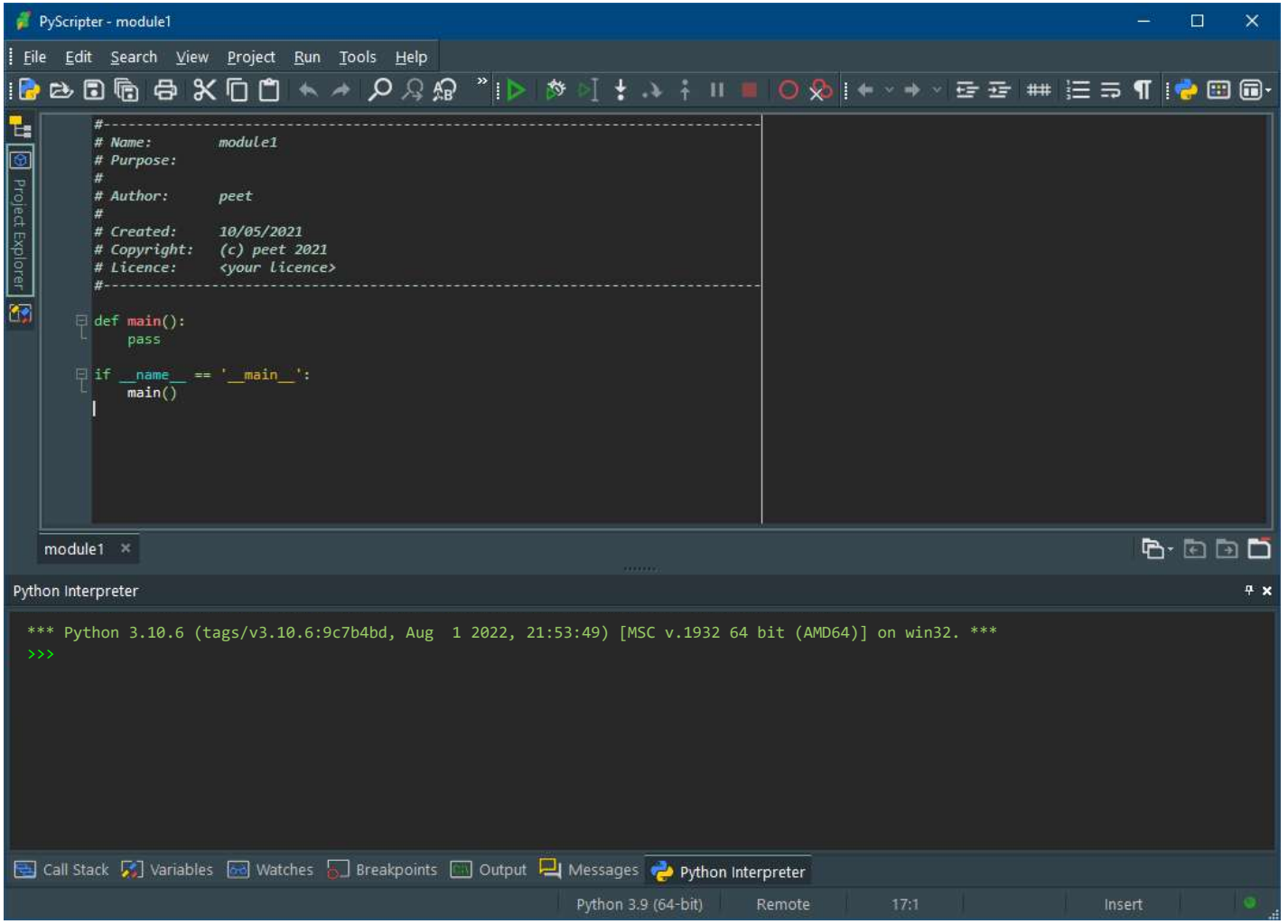
codeskulptor.org

repl.it

jupyter.org

colab.research.google.com

PyScripter (Windows)



Using variables. $E = mc^2$

```
m = 1/1000                          # kilograms.

c = 299_792_458                     # metres per second.

e = m * (c ** 2)                    # joules.

eInTonOfTNT = 4.184 * 10 ** 9       # gigajoules.

print(round(e / eInTonOfTNT))       # energy in 1 gram of anything
                                    # equal to 21,481 tons of TNT.
21484
>>>
```

Built-in Data Types

Type	Mutable	Iterable
`int`	No	n/a
`bool`	No	n/a
`float`	No	n/a
`str`	No	n/a
`set`	Yes	Yes
`dict (immutable keys, values)`	*Yes*	Yes
`list (indexes numbers)`	Yes	Yes
`tuple`	No	Yes

See also: Collections – other high-performance container datatypes

Operators

- Operators: `+ - / * // % **` *(% is modulus; // is integer division)*
- Bitwise: `& | ~ ^ << >>` *(treating numbers as binary bits)*
- Logical: `and or not` (*if this or that, and not the other*)
- Other: `in` (membership) `is` (identity)
- Comparison: `== != >= <= > <`

Decision Making

```
# value check.
#
# is x's value the same as n's value?

if x == n: # testing == results in either True or False

    do something

else: # x and n don't have the same value. How do they differ?

    do something else
```

```
if ____ and ____ and not ____ or ____:
```

Decision Making

```
if httpStatus == 400:
    print("Bad request")

elif httpStatus == 401 or httpStatus == 403: # elif means 'else if'
    print("Forbidden")

elif httpStatus == 404:
    print("Not found")

elif httpStatus == 418:
    print("I'm a teapot!")

else: # Nothing else matched httpStatus' value, so do this.
    print("Something went wrong, but who knows what!")
```

Decision Making

```
if httpStatus == 400:
    print("Bad request")

elif httpStatus in (401, 403): # (401, 403) a 'Tuple' containing two values.
    print("Forbidden")

elif httpStatus == 404:
    print("Not found")

elif httpStatus == 418:
    print("I'm a teapot!")

else:
    print("Something went wrong, but who knows what!")
```

Decision Making – `match`, new in 3.10.0

```
match httpStatus:

    case 400:
        print("Bad request")

    case 401 | 403:
        print("Forbidden")

    case 404:
        print("Not found")

    case 418:
        print("I'm a teapot!")

    case _:
        print("Something went wrong, but who knows what!")
```

Much more powerful than this.

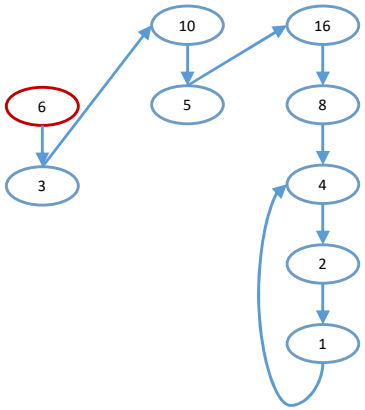
Iteration/Repetition

```
n = 5
i = 1

# Or
# n = 5; i = 1
# n, i = 5, 1

while i <= n:  # while True
               #
    print(i)   #     do this
               #
    i = i + 1  #     and this

print('Done')
```

```
for i in [1, 2, 3, 4, 5]: # a 'List'.

    print(i)

print('Done')


n = 5

for i in range(1, n + 1):

    print(i)

print('Done')
```

Problem Work-Through

- Write a program that continually re-calculates a value in a *loop*; stopping only if the calculated value is ever equal to 1:
  - Rules:
    - 1. Prompt for an arbitrary positive whole number.
    - 2. If the number's value is 1, the program terminates
    - 3. Otherwise, the program outputs the current value, and then checks whether the value is odd or even
      - If it is even, the program sets the current value to a new one by dividing it by 2, otherwise the program updates the current value by multiplying it by 3 and adding 1
    - Return to step 2.



Individual or Group Problem Solving

Real-time Problem (choose 1)

- Write a program that loops through *n* values, 1 – 100:
  - *if n is exactly divisible by 3, output 'fizz'*
  - *if n is exactly divisible by 5, output 'buzz'*
  - *if n is exactly divisible by both 5 and 3, output 'fizzbuzz'*
  - *if none of the above, just output n's value*

- Modify the 3*n* + 1 code:
  - *Modify the code so as to count the number of loops it makes before stopping, e.g., an input value of 100 decays to 1 in 26 loop-cycles*
  - *Try different starting values, what is your personal record for the number of cycles taken for a particular input?*
  - *Can you find a starting value so that the code never completes (n never goes to 1)?*

Fizzbuzz – a solution

```
for n in range(1, 100 + 1):

    if n % 3 == 0 and n % 5 == 0: print('fizzbuzz')

    elif n % 3 == 0: print('fizz')

    elif n % 5 == 0: print('buzz')

    else: print(n)
```

Fizzbuzz – a solution

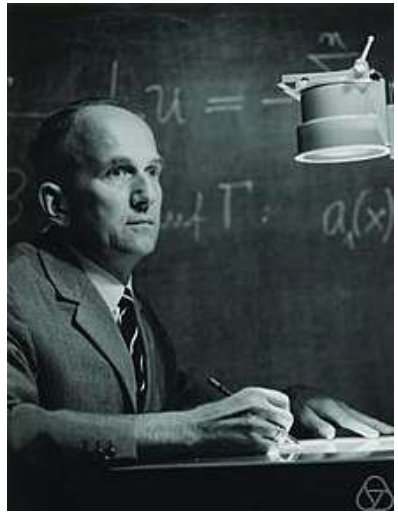
```
for n in range(1, 100 + 1):

    if n % 15 == 0: print('fizzbuzz')

    elif n % 3 == 0: print('fizz')

    elif n % 5 == 0: print('buzz')

    else: print(n)
```

Fizzbuzz – a solution

```
for n in range(1, 100 + 1):

    s = '' # empty string, length is zero.

    if n % 3 == 0: s += 'fizz' # same as s = s + 'fizz'.

    if n % 5 == 0: s += 'buzz'

    if len(s) == 0: s = str(n) # s is still empty? Set it to string version of n.

    print(s)
```

$3n + 1$ Problem (the Collatz Conjecture)

```
n = abs(int(input('Enter a whole positive integer value')))

while True:

    print(n)

    if n == 1:
        break # we're done, breaks out of the loop.

    if n & 1: # n is Odd. Could also use if n % 2 == 1, or simply, if n % 2:

        n *= 3 # same as n = n * 3.
        n += 1 # same as n = n + 1.

    else: # n is Even.

        n //= 2 # same as n = n // 2.
```



Lothar Collatz

Paul Erdős said, "Mathematics may not be ready for such problems".

Further reading

Outputting values

```
print(x, y)

print("x is ", end = ''); print(x, end = ''); print(", y is ", end = ''); print(y)

print("x is ", x, ", y is ", y)

print("x is " + str(x) + ", y is " + str(y))

print("x is %d, y is %.1f" % (x, y)) # printf style, from C/C++

print("x is {}, y is {}".format(x, y)) # format >= Python 2.6

print("x is {0}, x + y is {0} + {1} = {2}".format(x, y, x + y))

print(f"x is {x}, x + y is {x} + {y} = {x + y}") # f-string >= Python 3.6
```

old ways to format output

new way to format output

import

Math module functions

- `import math`
  - *You can now access any math function by putting **math.** in front of it:*
- `print(math.sqrt(5))`
  - *2.2360679774997898*
- `from math import *`
- `print(floor(log(255, 2) + 1))`

8

A few math module functions (use `dir(math)` for entire list)

Name	Description
`ceil(x)`	Ceiling of x
`cos(x)`	Cosine of x
`degrees(x)`	Converts x from radians to degrees
`exp(x)`	e to the power of x
`factorial(n)`	Calculates n! = 1*2*3*…*n n must be an integer
`log(x)`	Base e logarithm of x
`log(x, b)`	Base b logarithm of x
`pow(x, y)`	x to the power of y
`radians(x)`	Converts x from degrees to radians
`sin(x)`	Sine of x
`sqrt(x)`	Square root of x
`tan(x)`	Tangent of x

Built-in Data Types

Type	Mutable	Iterable	Subscriptable
`int`	No	n/a	n/a
`bool`	No	n/a	n/a
`float`	No	n/a	n/a
`str`	No	Yes	Yes
`list`	Yes	Yes	Yes
`dict`	Yes *(values)*	Yes	Yes *(keys)*
`tuple`	No *(ish)*	Yes	Yes
`set`	Yes	Yes	No

See also: Collections - High-performance container datatypes

List – ordered, indexed

- Created using `[ ]` or `list()`, e.g.,
  - `l = [0, '1', 2]` `print(l)` → `[0, '1', 2]`
  - `l = list('012')` `print(l)` → `['0', '1', '2']`
  - `l = [int(n) for n in '012'] # a list comprehension, [0, 1, 2].`
- Iterable
  - `for n in l: print(n)`
- Mutable / sliceable
  - `l.append(4)`
  - `l = l[1:] # Slicing.`
- Indexable
  - `print(l[1])`

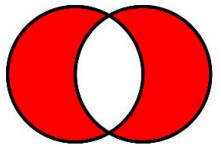
Dictionaries – ordered (3.6), indexed

- Created using `{}` or `dict()`, e.g.,
  - `d = {0:'1', 1:'2', 2:'3'}`
    `print(d)` → `{0:'1', 1:'2', 2:'3'}`
    `print(list(d))` → `[0, 1, 2] # keys.`
  - `d = {a:str(b) for a, b in enumerate(range(1, 4))} # dictionary comprehension.`
- Iterable
  - `for n in d: print(n) # prints keys. print(n, d[n]) prints keys and values.`
- Mutable
  - `d[len(d)] = '4' # changes the 2 key's value to '4'.`
- Indexable
  - `print(d[1]) # prints the value associated with the key of 1, which is '2'.`

tuple – ordered, indexed

- Created using `( )` or `tuple()`, or `,` e.g.,
  - `t = (0, '1', 2)` `print(t)` → `(0, '1', 2)`
  - `t = 0, '1', 2` `print(t)` → `(0, '1', 2)`
  - `t = tuple('012')` `print(t)` → `('0', '1', '2')`
  - `t = (int(n) for n in '012') # a generator.`
- Iterable
  - `for n in t: print(n)`
- Immutable / sliceable
  - `t = t[1:] # Slicing.`
- Indexable
  - `print(t[1])`

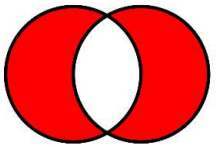
Set – unordered, unindexed



- Created using `{?}` or `set()`, e.g.,
  - `s = {0, '1', 2}` `s.add(2)` `print(s)` → `{0, '1', 2}`
  - `s = set('012')` `print(s)` → `{'0', '1', '2'}`
  - `s = {int(n) for n in '012'}` `print(3 not in s) # same for any iterable.`
- Iterable
  - `for n in s: print(n)`
- Mutable
  - `s.add(4); s.remove(3)`
- Set operations
  - `s.difference(k);` `s - k`
  - `s.intersection(k);` `s & k`
  - `s.union(k);` `s | k`
  - `s.symmetric_difference(k);` `s ^ k`

- `s.subset(k)`
- `s.superset(k)`

Set

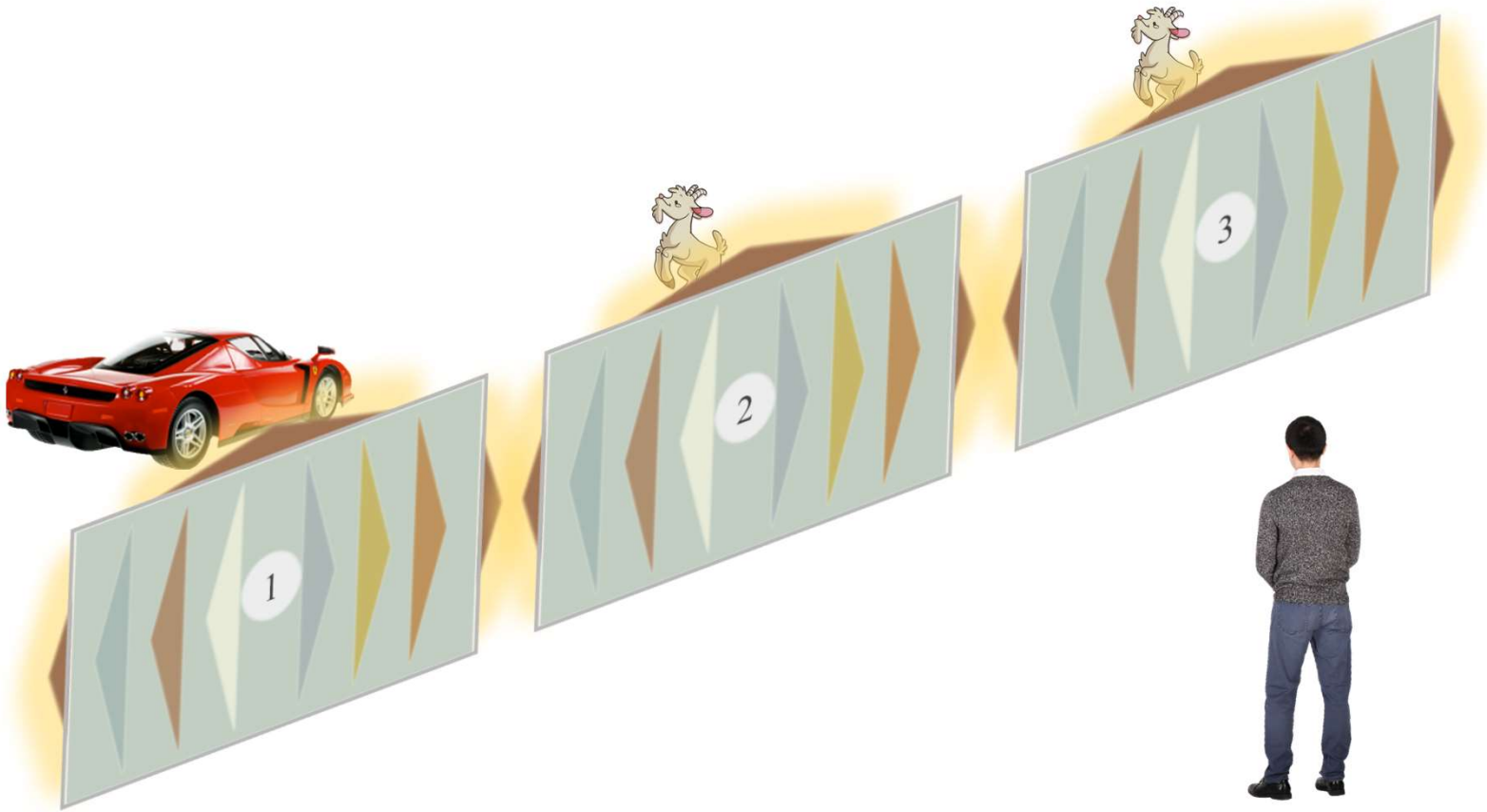


- `s.difference(k);` `s - k`
- `s.intersection(k);` `s & k`
- `s.union(k);` `s | k`
- `s.symmetric_difference(k);` `s ^ k`

```
s = set([1, 2, 3]); k = set([3, 4, 5])
print(s)          # {1, 2, 3}
print(k)          # {3, 4, 5}
print(s - k)      # {1, 2}
print(s & k)      # {3}
print(s | k)      # {1, 2, 3, 4, 5}
print(s ^ k)      # {1, 2, 4, 5} … {(s – k) U (k – s)}
```

Problem Walk-Through

The Monty Hall Problem


1
2
3

Ask Marilyn™

BY MARILYN VOS SAVANT

Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?
—Craig F. Whitaker, Columbia, Md.

Yes; you should switch. The first door has a one-third chance of winning, but the second door has a two-thirds chance. Here's a good way to visualize what happened. Suppose there are a *million* doors, and you pick door No. 1. Then the host, who knows what's behind the doors and will always avoid the one with the prize, opens them all except door #777,777. You'd switch to that door pretty fast, wouldn't you?

Let me explain: If one door is shown to be a loser, that information changes the probability to 1/2. As a professional mathematician, I'm very concerned with the general public's lack of mathematical skills. Please help by confessing your error and, in the future, being more careful.

—Robert Sachs, Ph.D.,
George Mason University, Fairfax, Va.

Your answer to the question is in error. But if it is any consolation, many of my academic colleagues also have been stumped by this problem.

—Barry Pasternack, Ph.D.,
California Faculty Association

You blew it, and you blew it big! I'll explain: After the host reveals a goat, you now have a one-in-two chance of being correct. Whether you change your answer or not, the odds are the same. There is enough mathematical illiteracy in this country, and we don't need the world's highest IQ propagating more. Shame!

—Scott Smith, Ph.D., University of Florida

Your logic is in error, and I am sure you will receive many letters on this topic from high school and college students. Perhaps you should keep a few addresses for help with future columns.

—W. Robert Smith, Ph.D.
Georgia State University

I am in shock that after being corrected by at least three mathematicians, you still do not see your mistake.

—Kent Ford,
Dickinson State University

You are utterly incorrect about the game-show question, and I hope this controversy will call some public attention to the serious national crisis in mathematical education. If you can admit your error, you will have contributed constructively toward the solution of a deplorable situation. How many irate mathematicians are needed to get you to change your mind?

—E. Ray Bobo, Ph.D.,
Georgetown University

You are the goat!

—Glenn Calkins
Western State College

You're wrong, but look at the positive side. If all those Ph.D.s were wrong, the country would be in very serious trouble.

—Everett Harman, Ph.D.,
U.S. Army Research Institute

Maybe women look at math problems differently than men.

—Don Edwards, Sunriver, Ore.

May I suggest that you obtain and refer to a standard textbook on probability before you try to answer a question of this type again?

—Charles Reid, Ph.D.,
University of Florida

Problem Work-Through

- To test Marilyn's assertion, let's write a simulation of 'The Monty Hall Problem'.
  - *Monte Carlo Simulation (inferential statistics)*

Monty Hall

```
import random

doors = [1, 2, 3]

games = 1000

carsWon = 0

for n in range(games):

  carDoor    = random.choice(doors)
  playerDoor = random.choice(doors)

  # The host opens a different door to reveal a goat
  # (always able to do this as there are 2 goats).
  montyDoor = random.choice(list(set(doors) - set([carDoor, playerDoor])))

  # ~~~~ To stick, just comment out the next line -
  # it implements that the player is swapping doors.
  playerDoor = list(set(doors) - set([playerDoor, montyDoor]))[0]
```

```
if playerDoor == carDoor:
    carsWon += 1

print('The player won the car ' +
 str(round(carsWon / (games / 100))) +
 '% percent of the time')
```

See cslab.com/monty for more.

Individual or Group Problem Solving

Real-time Problem

- Modify the 3n + 1 code:
  - *If we didn't break out of the loop when we get to 1, we would loop endlessly over 4, 2, 1.*

    Alter your code to use a Python `set` *to detect that we've previously seen an output, and break out of your loop when that's seen.*
  - *Can you find other (hailstone) sequences by altering the algorithm slightly, i.e., those not terminating in 4, 2, 1? Hint, you'll need to start with negative starting values and look no further than, say, -20.*

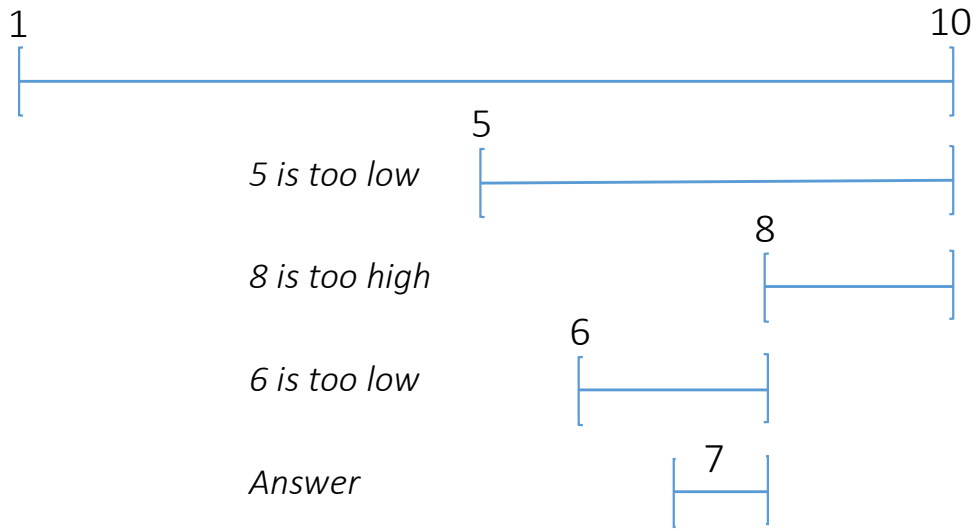
    Can you do this programmatically (test a range of inputs consecutively in an outer loop)?

- Use a *list comprehension* to create a `list` of the squares of the integers 1 – 10 inclusive.
  - *Then; a bit more complex – use* ***`enumerate()`*** *to build a* ***`list`*** *of two-element* ***`lists`*** *– the thing being squared, and the squared result, e.g.,*

```
[[1, 1],      # First enumeration, 1² is 1
 [2, 4],      # Second enumeration 2² is 4.
 [3, 9],      # ...
 [4, 16],
 [5, 25],
 [6, 36],
 [7, 49],
 [8, 64],
 [9, 81],
 [10, 100]]
```

Real-time Problem

- Write the beginnings of a guessing game in which the program gives hints to the user. 'Pick a number between say, 1 and 10. Count how many guesses are required.



$3n + 1$ Problem (using a set)

```
n = int(input('Enter a whole integer value'))

s = set() # using a set.

while True:

    print(n)

    if n in s: # set membership test.

        break

    s.add(n) # here only if n not a member of set s.

    if n & 1: # n is Odd. Could also use if n % 2 == 1:

        n *= 3
        n += 1

    else: # n is Even.

        n //= 2
```

import

Math module functions

- `import math`
  - *You can now access any math function by putting **math.** in front of it:*
- `print(math.sqrt(5))`
  - *2.2360679774997898*
- `from math import *`
- `print(floor(log(255, 2) + 1))`

8

A few math module functions (use `dir(math)` for entire list)

Name	Description
`ceil(x)`	Ceiling of x
`cos(x)`	Cosine of x
`degrees(x)`	Converts x from radians to degrees
`exp(x)`	e to the power of x
`factorial(n)`	Calculates n! = 1*2*3*…*n n must be an integer
`log(x)`	Base e logarithm of x
`log(x, b)`	Base b logarithm of x
`pow(x, y)`	x to the power of y
`radians(x)`	Converts x from degrees to radians
`sin(x)`	Sine of x
`sqrt(x)`	Square root of x
`tan(x)`	Tangent of x

Functions

```
import math

n = 100

result = math.factorial(n)
```

```
def factorial(n):

    prod = 1

    for i in range(2, n + 1):

        prod *= i # same as prod = prod * i

    return prod


n = 100

result = factorial(n)
```

Functions

```
import math

n = 100

result = math.factorial(n)
```

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)


n = 100

result = factorial(n)
```

Recursion vs Iteration

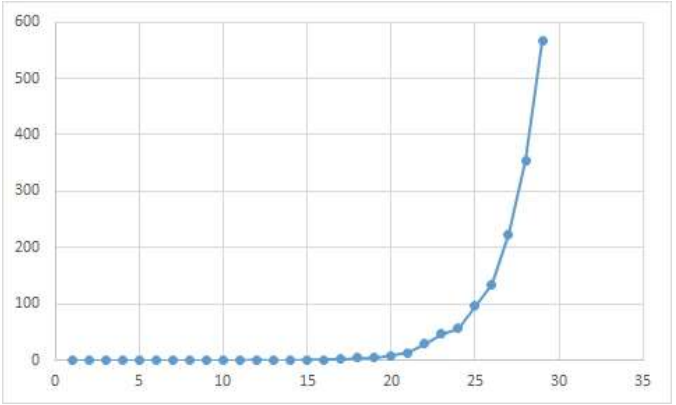
```
def fibonacci(n):
    if n < 2:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

n = 11
print([fibonacci(i) for i in range(n)])
```

```
def fibonacci(n):
    a = 1
    b = 1
    for _ in range(1, n):
        t = a
        a = b
        b = t + b
    return a

n = 11
print([fibonacci(n) for n in range(n)])
```

```
>>> 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

Recursion - fibonacci

```
import time

def fibonacci(n):
    if n < 2:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

for n in [1, 5, 10, 15, 20, 25, 30, 35]: # Increase in small increments.
    t1 = time.perf_counter_ns() # Nano second timer.
    k  = fibonacci(n)
    t2 = time.perf_counter_ns()
    print(f'Computing the {n}th fib number ({k}) took {t2 - t1:,} nano seconds')
```



$$O(2^n)$$

A bit more *pythonic*

```
def fibonacci(n):
    a, b = 1, 1
    for _ in range(1, n):
        a, b = b, a + b
    return a
print(fibonacci(10))
```

Monty Hall

```
import random

doors = [1, 2, 3]

def playGame():

  carDoor    = random.choice(doors)
  playerDoor = random.choice(doors)

  # The host opens a different door to reveal a goat
  # (always able to do this as there are 2 goats).
  hostDoor = random.choice(list(set(doors) - set([carDoor, playerDoor])))

  # ~~~~ To stick, just comment out the next line -
  # it implements that the player is swapping doors.
  playerDoor = (set(doors) - set([playerDoor, hostDoor])).pop()

  return True if playerDoor == carDoor else False
```

```
carsWon = 0

for games in range(1000):
  if playGame():
    carsWon += 1

print('The player won the car ' +
   str(round(carsWon / (games / 100))) +
   '% percent of the time')
```

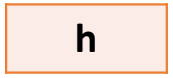
Strings – immutable, ordered, indexed

```
s = 'hello world'               # variable called s
print(len(s))                   # number of characters in s. 11
s = s.title()                   # title case s, assign back to s
print(s)                        # 'Hello World'
print(s.find('o'))              # o is 4th char from left starting from 0
print(s.rfind('o'))             # different o in position 7
print(''.join(reversed(s)))     # 'dlroW olleH'
print('Wo' in s)                # True
n = s.find('World')             # n is 6
print(s[n:])                    # 'World'
print(s[0::2])                  # 'HloWrd'
print(s[::-1])                  # 'dlroW olleH'
```

Strings and 'Slicing'

```
s = 'hello'

# s[start:xstop]
```

`s[0]` same as `s[0:1]`



`s[1]` same as `s[1:2]`

`s[-4:-2]` or `s[1:3]`

Class

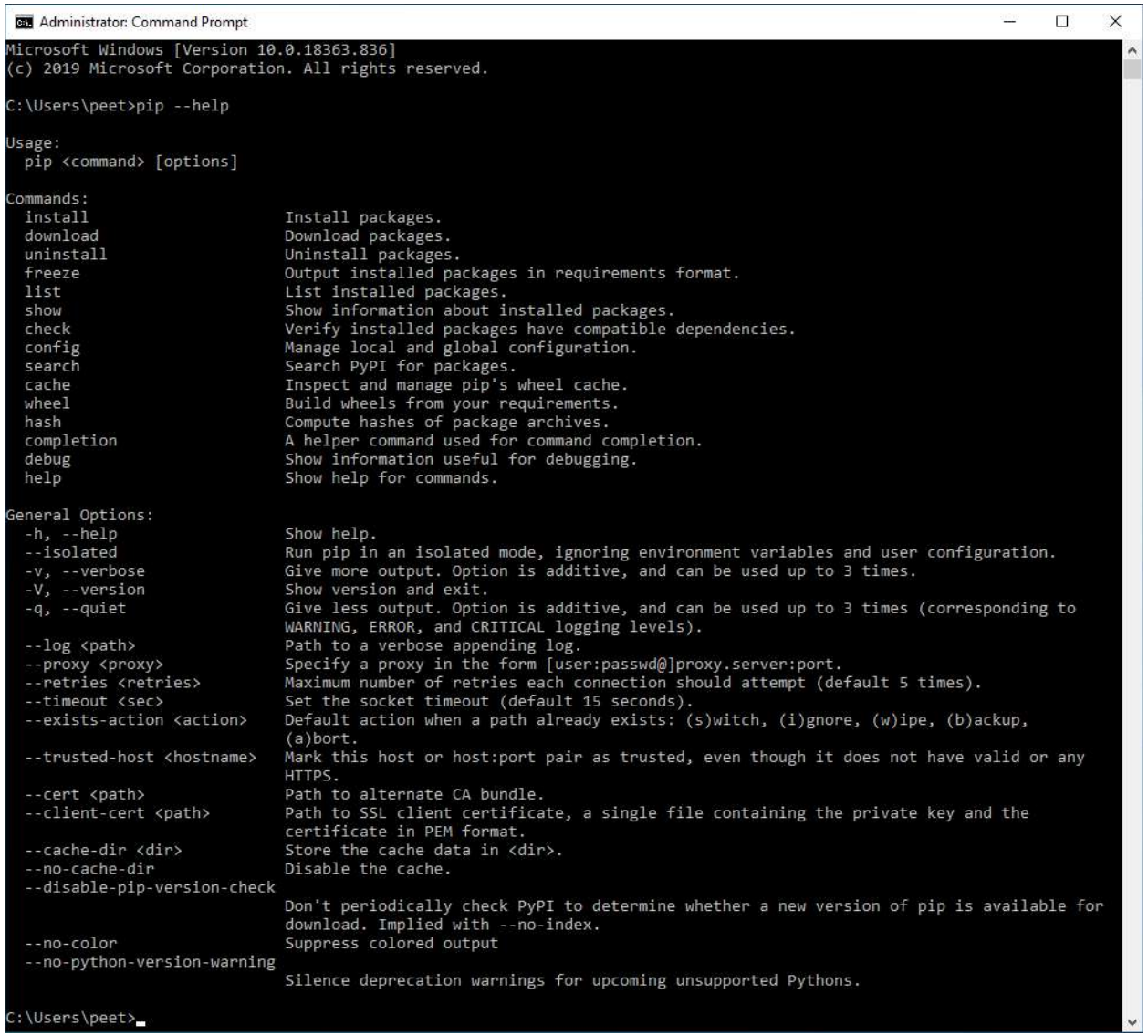
```
import random

class dice():

    def __init__(self, sides = None):

            self.sides = 6 if sides is None else sides

            self.throws = [n for n in range(1, self.sides + 1)]

    def throw(self): # throw is a method of dice.

        return random.choice(self.throws)


d1 = dice(); d2 = dice(20)

print(d1.throw(), d2.throw())
```

Problem Work-Through

- Write a program to generate a sequence of coin tosses. e.g., HTH.
- On average, how many coin-tosses are needed to see a given sequence?

Sequence	Tosses
HTTTHHTTHHTTTTHTH	17
HHHTTTHHTTHHTH	14
HHHHTH	6
HHTH	4
HHHTTHHHTTTTHHTTHTH	19
HHTTHHHHHTTHHHTTHHHHTH	22
HTTHHHTH	8
TTHHTTTHTTTHTH	14
TTTHHTTTTTHTH	13
TTHTH	5
THTTHTH	7
HTH	3
THHTH	5
HHTTTTTHHHHHTTTHTH	18
HHHTH	5

Average # of tosses to see HTH was …



Package installation (pip/pip3)

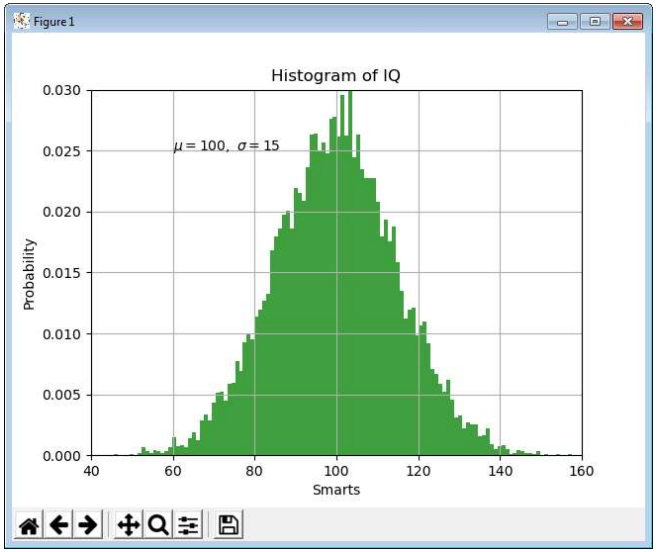
- Use pip to
  - *pip install numpy*

    and

  - *pip install matplotlib*
- Used on the following slide

Copy the code below into a new module to test your installation

```
import numpy as np
import matplotlib.pyplot as plt

mu, sigma = 100, 15
data_set = mu + sigma * np.random.randn(10000)

plt.hist(data_set, 125, density = 1, facecolor = 'g', alpha = 0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.text(60, .025, '$\mu=100, \sigma=15$')
plt.xlim(40, 160)
plt.ylim(0, 0.03)
plt.show()
```



[Data Science Courses - https://www.linkedin.com/...]

Homework

If HTT beats HTH, what if anything beats HTT? And does something beat that etc? Is there an optimal sequence?

You can use this code to produce a list `l` of all the possible starting permutations:

```
import itertools

rep = 3

l = list(itertools.product('HT', repeat = rep))

for n in range(len(l)):

    l[n] = ''.join(l[n])

print(l, len(l))
```

Homework

- Along with the slides from today, and the coin tossing simulation and other code, I am giving you the code of a program that can access and parse RSS feeds, in particular, the BBC's.

- At the end of the code there are some suggestions for modifying the it.

BBC NEWS

RSS Feed For: **BBC News - Technology**

Below is the latest content available from this feed. This isn't the feed I want.

Location data collection firm admits privacy breach
Huq says two apps it collects data from did not seek correct consent from users.

Inside the controversial US gunshot-detection firm
BBC News has been given access to ShotSpotter, the controversial US technology company that detects gunshots.

Facebook: We are putting safety before profits
Facebook Vice President says Meta is about making sure people feel good about using their services.

Meta: Facebook's new name ridiculed by Hebrew speakers
The social media giant joins a number of companies that have fallen foul of translation blunders.

Facebook changes its name to Meta in major rebrand
The social media giant says the new name will better encompass what it does.

UK's electric Arrival promises greener deliveries
A modular design lets vehicles carry the batteries needed for expected trips, avoiding extra weight.

Arm-Nvidia: Europe investigates chip-designer sale
Regulators have "serious doubts" about the technology giant's $40bn takeover of British company Arm.

Chinese payment-terminal company searched by FBI
Pax Technology is a major provider of payment terminals worldwide.

Facebook changes its name to Meta in major rebrand
The social media giant says the new name will better encompass what it does.

UK's electric Arrival promises greener deliveries
A modular design lets vehicles carry the batteries needed for expected trips, avoiding extra weight.

Arm-Nvidia: Europe investigates chip-designer sale
Regulators have "serious doubts" about the technology giant's $40bn takeover of British company Arm.

Chinese payment-terminal company searched by FBI
Pax Technology is a major provider of payment terminals worldwide.

Resources

- *Real Python – Link*
- *Python Programming Tutorials – Link*
- *Linked-In Learning (of course!)*
- *Socratica Python - Link*
- *Anything Python by Corey Schafer - Link*
- *Any YouTube videos by Raymond Hettinger*
- *Pandas (Python Data Analysis Library) - Link*
- *A little more technical: Python as C++'s Limiting Case - Link*

Resources – Me!

If you would like to book a one-to-one session with me, please just email me to arrange a suitable date and time.

Always happy to help. And it's free.

Any final questions?