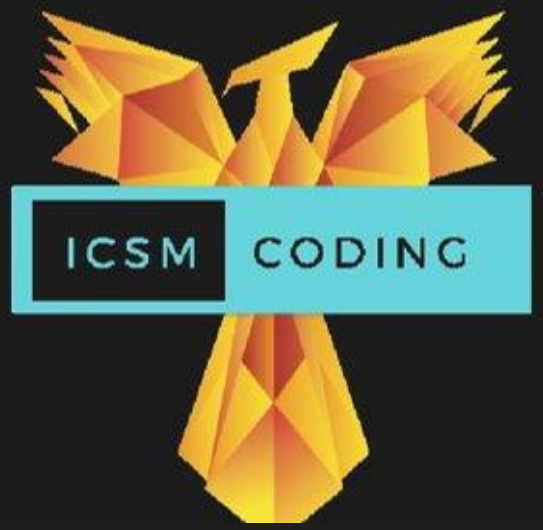


Welcome!!

What will we cover?

	Saturday 14/11/20
10:00 – 10:30	Intro / Getting Anaconda
15min Breakout!	Getting familiar with Jupyter
10:45 – 11:30	Variable, Types and Operators
30min Breakout!	Vs, Ts & Os worksheet
12:00 – 13:00	Lunch
13:00 – 13:45	Loops and Flow
45min Breakout!	Loops and Flow worksheet
14:30 – 14:45	Debrief / See you tomorrow

	Sunday 15/11/20
10:00 – 10:45	Welcome back / Functions
30min Breakout!	Functions Worksheet
11:15 – 12:00	Classes / Objects
30min Breakout!	Cs and Os Worksheet
12:30 – 13:30	Lunch
13:30 – 14:15	Example program
30min Breakout!	Make something!
14:45 – 15:00	Thanks

What is a computer programme?

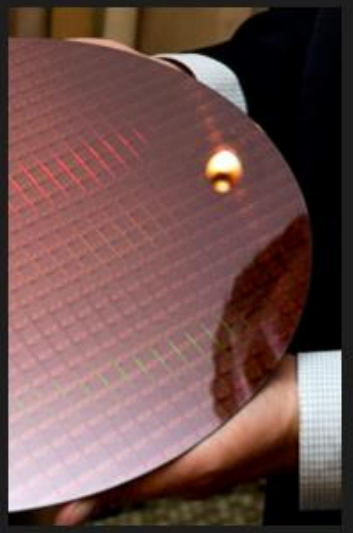
What is a computer programme?

Mechanical

Vacuum Tubes

Transistors

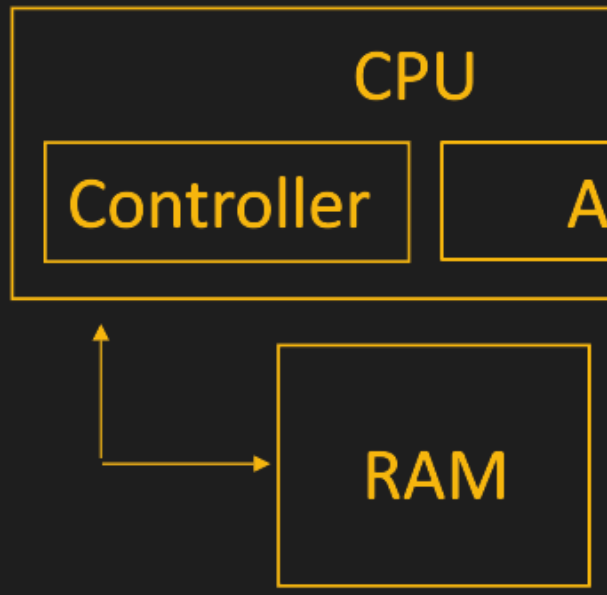
Smaller Transistors



Even smaller Transistors

What is a computer programme?

```
1    # A most polite greeting
2
3    name = input("Whats the name bossman? \n")
4    print("Wagwaan,", name)
```

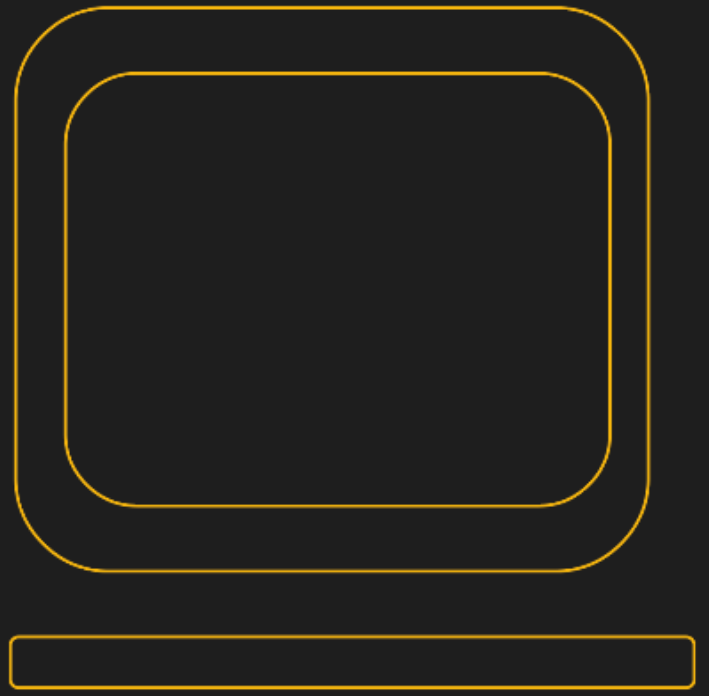


```
1    0x 60 01 00 84
2    0x B6 01 72 9C
3    0x 44 D1 10 00
4    0x 6B FF 43 F1
5    0x 12 3E 45 DA
6    0x 00 01 BB A9
7    ..
8    ..
9    ..
```

Compiled VS Interpreted

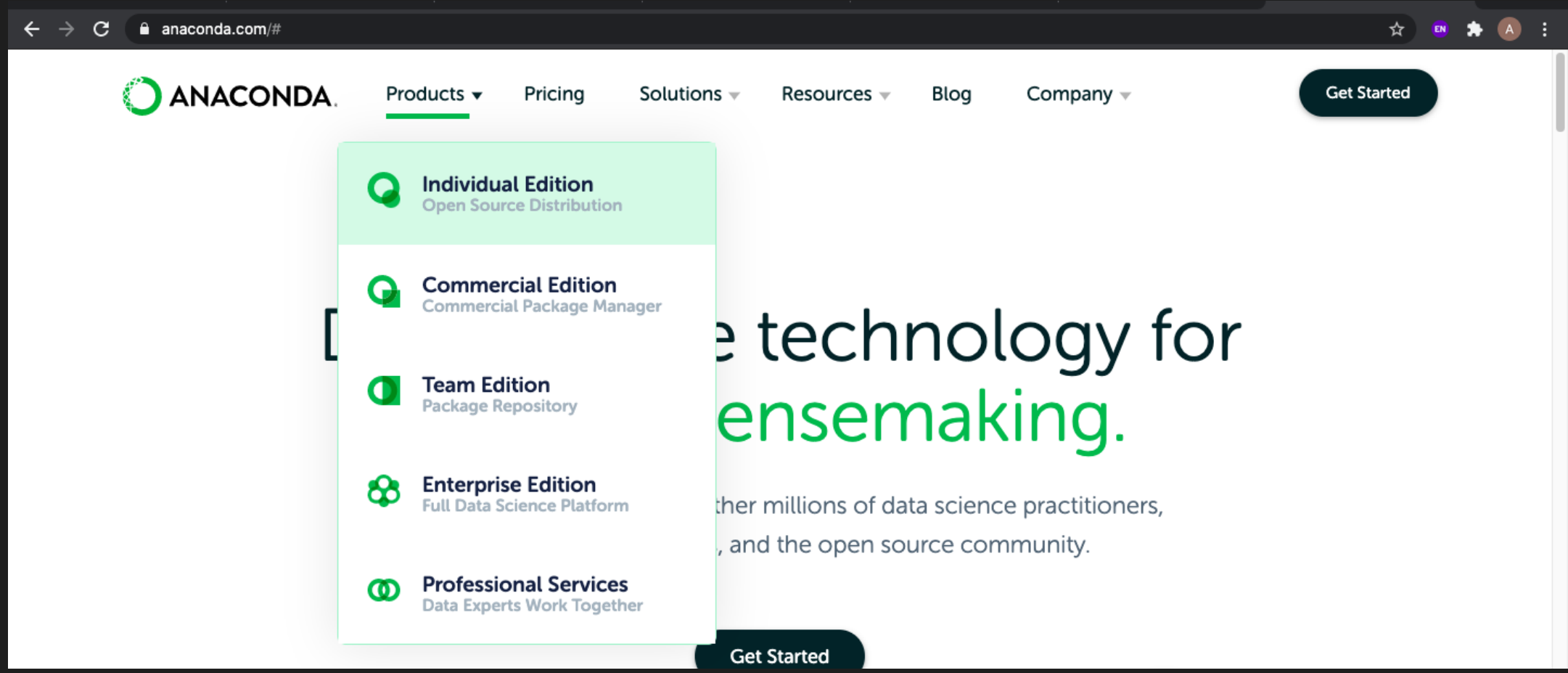
Interpreted

```python
# programme that calculates combinatorials

#function to calculate factorials recursively
def factorial(number):
    if number is 1: return 1
    else: return number * factorial(number - 1)

#function that return the number of ways to choose 'r' items from 'n' items
def combinatorial(n, r):
    return factorial(n) // ( factorial(r) * factorial(n - r) )

def main():
    item_count = int(  input("How many things bro? \n")  )
    number_to_choose = int(  input("How many do you want to choose man? \n")  )
    combinations = combinatorial(item_count, number_to_choose)
    answer_string = "There are {} ways to choose {} items out of {}"
    print(answer_string.format(combinations, number_to_choose, item_count))

if __name__ == "__main__":
    main()
```



- One download.
- Everything you need and more.

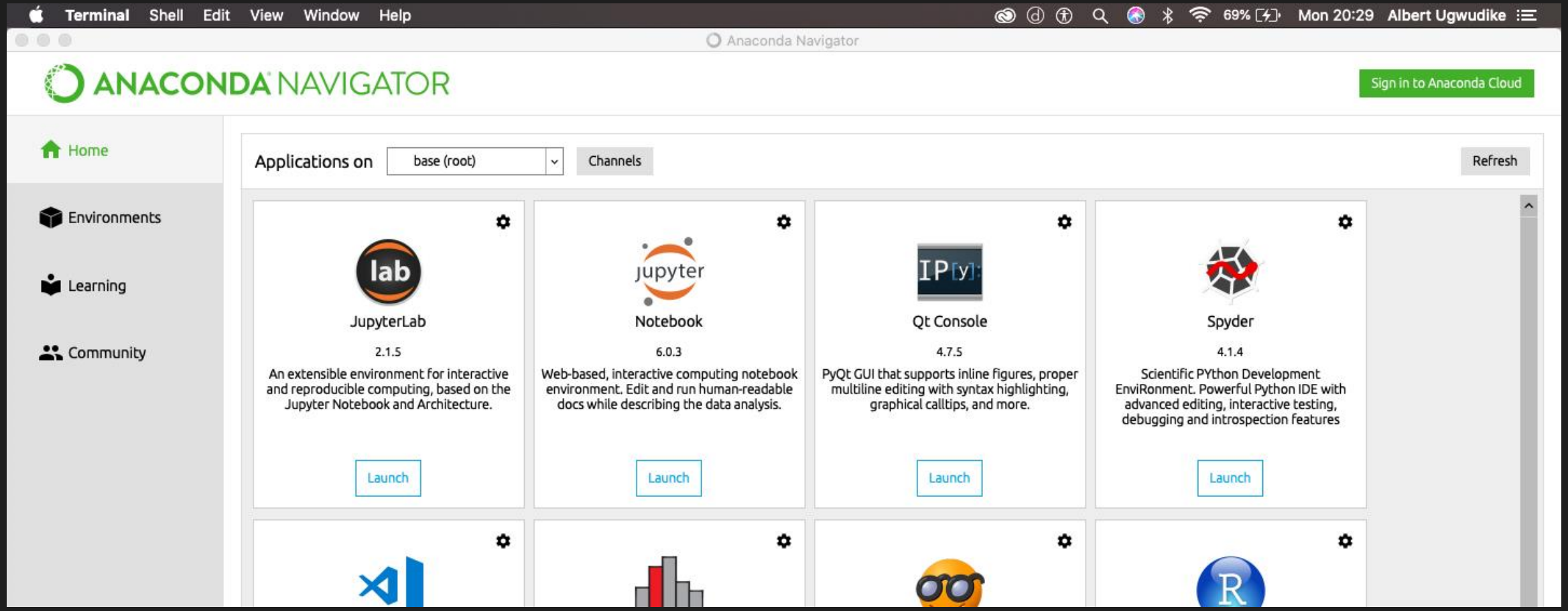
Getting Anaconda



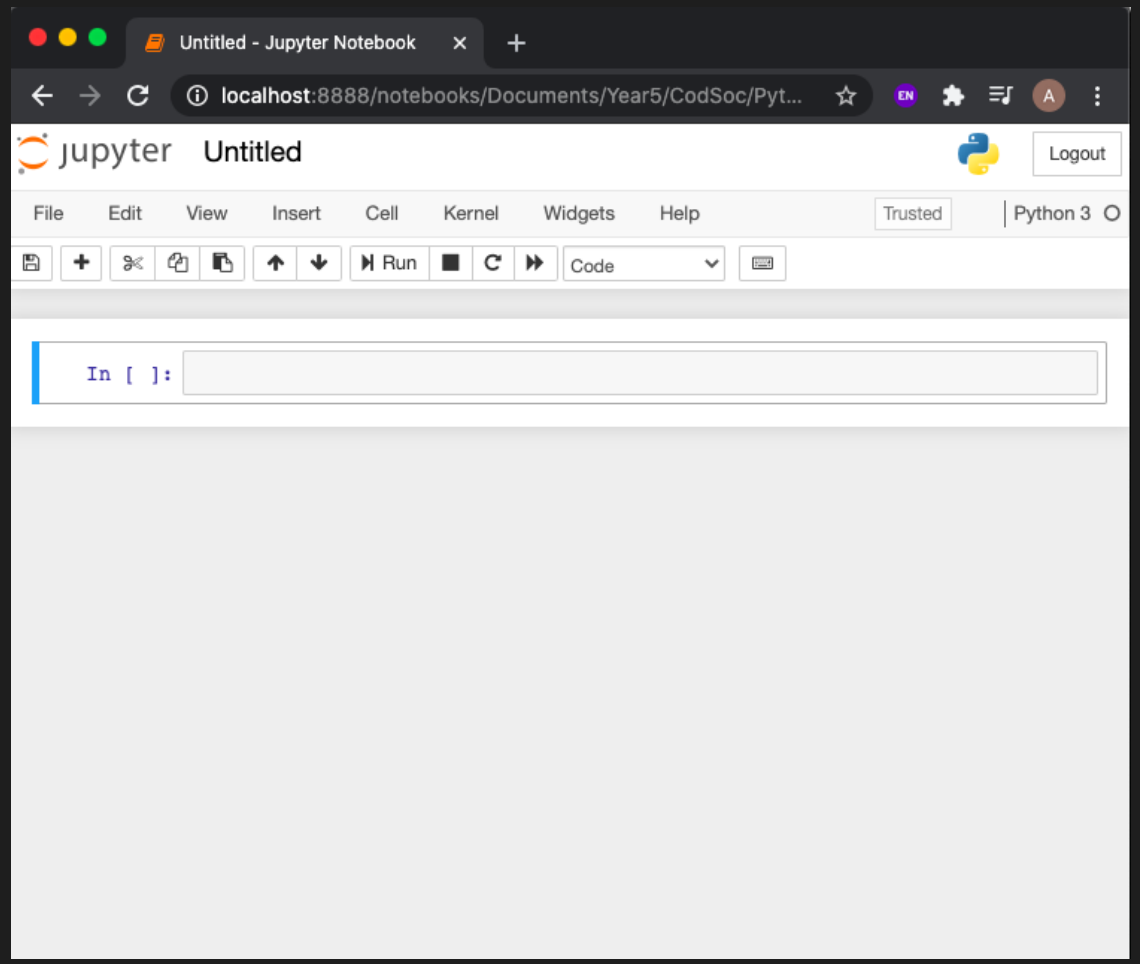
Getting Anaconda

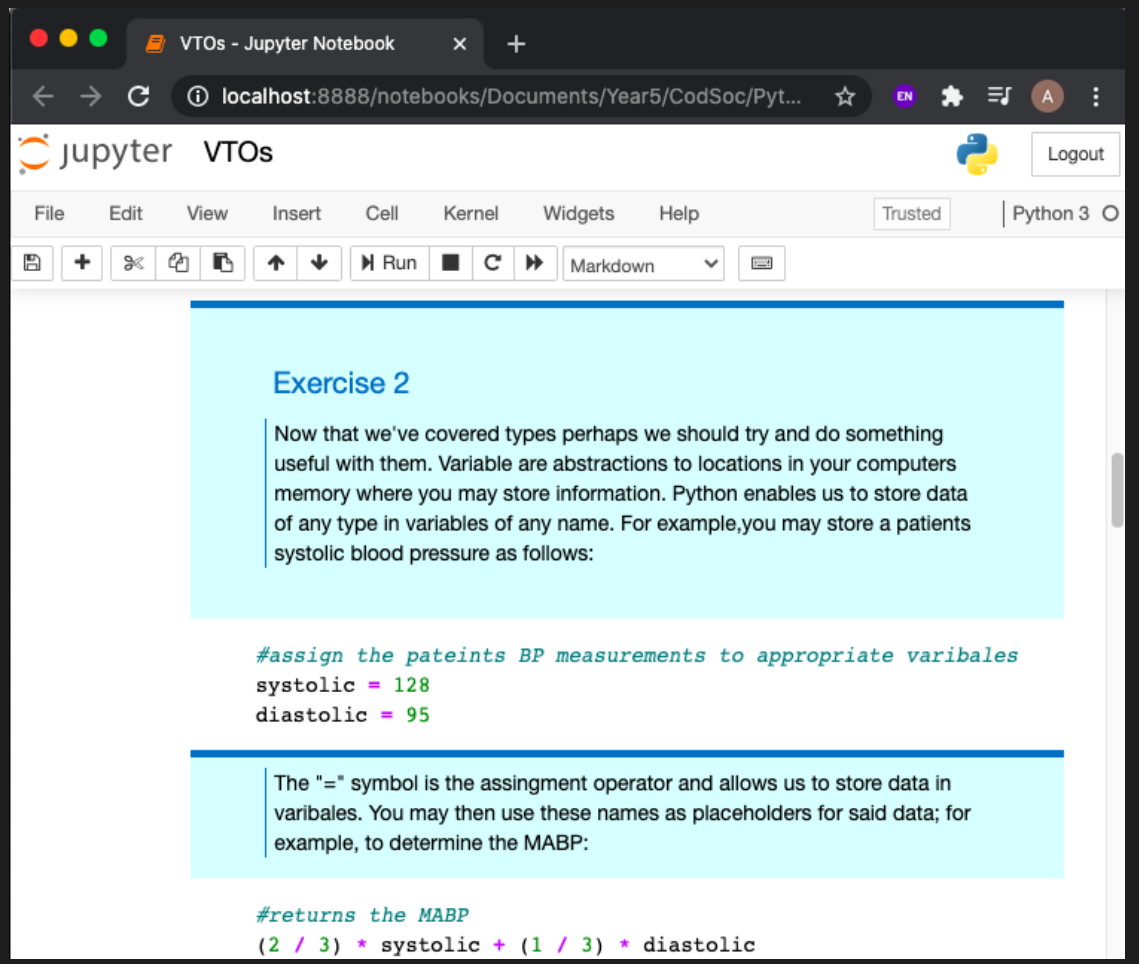
- The easiest way to use python
- Documentation, Graphics and more!

Open Jupyter



Create a notebook!





Commands are your friends

Command	Action
Enter	Edit Cell
Shift + Enter	Run Cell
Esc	Stop Editing
A / B	New cell Above / Below
M	Markdown Mode
Y	Code Mode
DD	Delete Cell

Lets try this out....

Code Cells


```
In [5]: #lets calculate a factorial
        def factorial(n):
            if not n: return 1
            else: return n * factorial(n - 1)

        print(factorial(10))
```

```
3628800
```

```
In [ ]:
```

The python interpreter at your fingertips

Display output below the cell

Allow for less painful debugging

Markdown cells


```
# Markdown

### What is Markdown
<pre>
Markdown is a markup language. Simply put, markup
languages like this and HTML enable you to
change the appearance of text. This particular
language was designed to make markup as simple as
possible
```

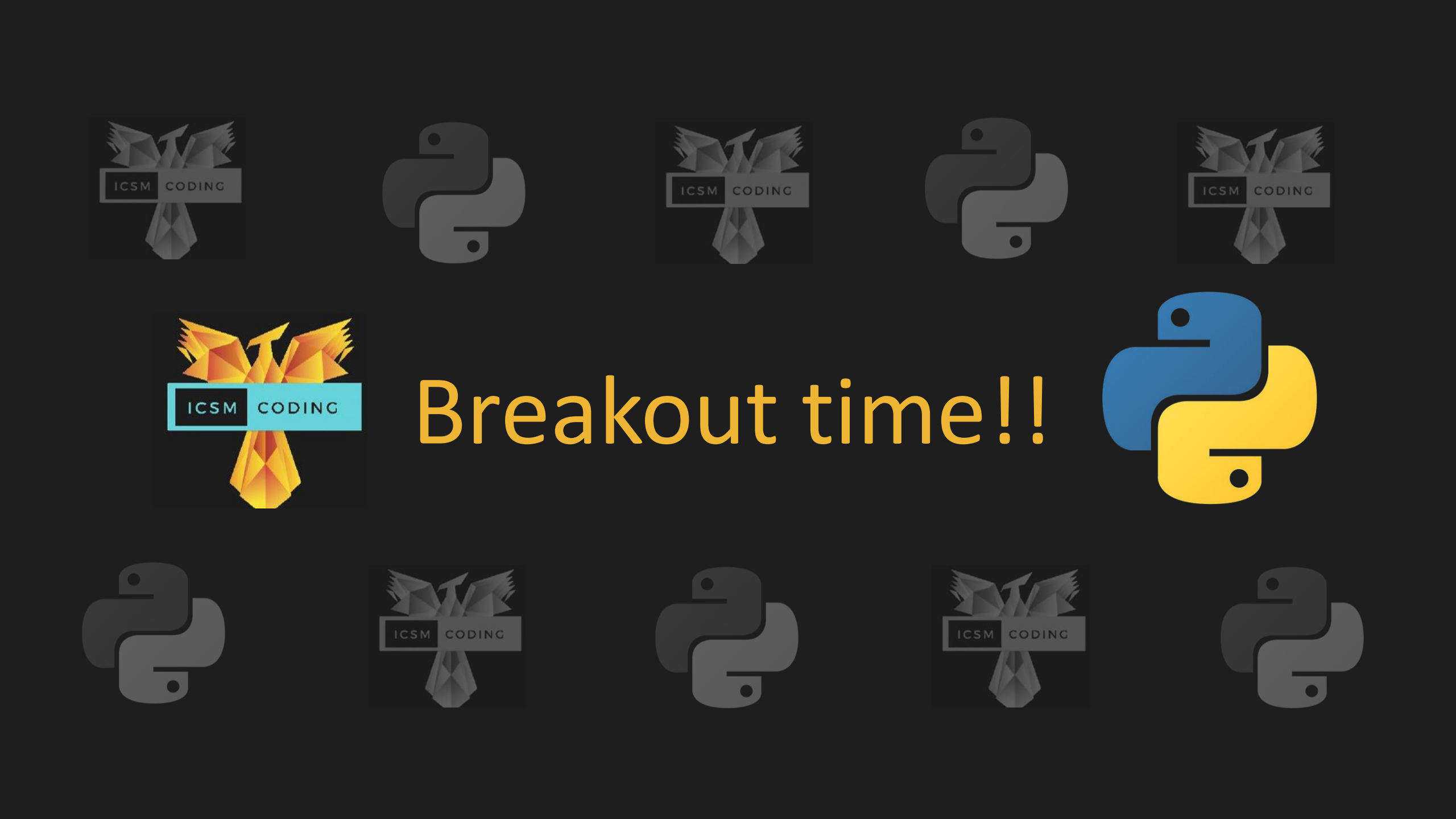
In [ ]:

Markdown

What is Markdown

Markdown is a markup language. Simply put, markup languages like this and HTML enable you to change the appearance of text. This particular language was designed to make markup as simple as possible

In [ ]:


ICSM CODING
Breakout time!!

Variables

x

temp i

y

j out

Types

int

float list

bool

dict tuple

Operators

+

= -

**

> ==

What is a type?

- The kind of data you're using
- Python implements laaazy typing

Python name	Full name	Content
int	Integer	Whole numbers
float / double	Floating point	Other numbers
str	string	characters
bool	Boolean	True or False
list	list	Collection of items
tuple	tuple	
dict	dictionary	Key – value pairs

basic types

Integers
Floats
Strings
Booleans

int

Integers i.e. the counting numbers
0, 1, 2, 3, 4, 5 ……

128	64	32	16	8	4	2	0	
0	0	0	0	0	0	0	0	**0**
0	0	0	0	0	0	0	1	**1**
0	0	0	0	0	0	1	0	**2**
0	0	0	0	0	0	1	1	**3**
…								
1	0	1	0	1	0	1	0	**170**
1	0	1	0	1	0	1	1	**171**
…								
1	1	1	1	1	1	1	1	**255**

8 bits

Floats and Doubles

Numbers with a decimal (floating) point

$$0.299792458 * 10^{9}$$

Operand

Exponent

strings

ASCII letters

A	65	N	78
B	66	O	79
C	67	P	80
D	68	Q	81
E	69	R	82
F	70	S	83
G	71	T	84
H	72	U	85
I	73	V	86
J	74	W	87
K	75	X	88
L	76	Y	89
M	77	Z	90

Characters and symbols
a, A, B, :, -,)....

RAM	Binary	ASCII
1	01101000	"h"
2	01100101	"e"
3	01101100	"l"
4	0110110	"l"
5	01101111	"o"
6	00100000	" "
8	01001001	"I"
9	01000011	"C"
10	0101010	"S"
11	01001101	"M"

Booleans

True

1, 20, -37

"any text"

[1, 2, 3]

{ "key", "value" }

False

0

" "

[ ]

{ }

None

Pop Quiz!!

True

?

False

?

Container Types

[Lists]

(Tuples)

{ dictionaries }

Lists

[1, 2, 3, 4, 5, 6]

[7, 7.4, "A", 99, "12"]

[[0,0,0], [0,0,0], [0,0,0]]

["this", "is", "a", "list"]

Lists: Indexing

0	1	2	3	4	5	6	7
7	23	9	6	54	1	1	2
-8	-7	-6	-5	-4	-3	-2	-1

example = [7, 23, 9, 6, 54, 1, 1, 2]

- This list (*example)* contains eight elements
- example[0] == 7 (the 0th element is 7)
- example[7] == 2 (the last element is 2)
- example[-1] == 2
- example[-8] == 7

Lists: Slicing

0	1	2	3	4	5	6	7
example = [7,	*23,*	*9,*	*6,*	*54,*	*1,*	*1,*	*2]*
-8	-7	-6	-5	-4	-3	-2	-1

- Example[0 : 2] == [7, 23]
- Example[1 : 6] == [23, 9, 6, 54, 1]
- Example[-4 : -2] == [54, 1]
- Example[2 :] == [23, 9, 6, 54, 1, 1, 2]
- Example[: 3] == [7, 23, 9]

Dictionaries

{ 0 : 1, 1 : 2, 2 : 3, 3 : 4 }

{ "name" : "David", "age" : 94}

{ "pt1" : 42, "pt2" : 43, "pt3" : 40 }

{ "names" : ["Albert", "Ben"], "scores" : [100, 67] }

Dictionaries: indexing


```
Example = {  "first" : "David",
             "last": "Benson,
             "age" : 94   }
```

Key : Value

- Example["first"] == "David'
- Example["second"] == "Benson"
- Example["age"] == 94

Tuple

(1, 2, 3, 4, 5, 6)

(7, 7.4, "A", 99, "12")

("singleton",)

("this", "is", "a", "list")

Pop Quiz

example = [7, 23, 9, 6, 54, 1, 1, 2]

What is the –4th element?
What about example[3:6]?

Variables

Assignment Operator

```
my_list = [ 1, 2, 3, 4, 5, 6 ]
```

Variable name

Data

Variables: immutable types

```
>>> x = 24
>>> x = 25
>>> x = x + 20
>>> print_(x)
    45
>>> y = x
>>> x = 2 * x
>>> print_("x = ", x)
    x =  90
>>> print_("y = ", y)
    y =  45
```

Copy!!

Variables: mutable types


```
>>> my_list = [1, 2, 3, 4, 5]
>>> other_list = my_list
>>> my_list[3] = 100
>>> print_("my_list is now", my_list)
    my_list is now [1, 2, 3, 100, 5]
>>> print_("other_list is now", other_list)
    other_list is now [1, 2, 3, 100, 5]
```

2nd reference!!!

Mathematical Operators

Operator	Name	Usage
=	Assignment	Store data in a variable
+	Addition	Add numbers*
-	Subtraction	Subtract numbers
*	Multiplication	Multiply numbers*
**	Exponentiation	Raise number to a power
/	True division	Ordinary division
//	Integer division	Floored division
%	Modulo	Remainder upon division

addition


```
>>> print_(1 + 1)
    2
>>> thirty = 30
>>> print_(thirty + 90)
    120
>>>
>>> print_([1,24,6] + [9,9,9])
    [1, 24, 6, 9, 9, 9]
>>>
>>> print_(("tuple",) + ("tuple",))
    ('tuple', 'tuple')
```

subtraction and exponentiation


```
>>> i = 420
>>> j = 420 - 144
>>> print_(j)
    276
>>>
>>> print_(2 ** 4)
    16
>>> print_(729 ** (1 / 6))
    3.0
```

True and Integer division


```
>>> numerator = 44
>>> denominator = 7
>>> print_(numerator / denominator)
    6.285714285714286
>>>
>>> print_(numerator // denominator)
    6
```

Modulo


```
>>> print_(38 % 3)
    2
>>> print_(37 % 3)
    1
>>> print_(36 % 3)
    0
>>> print_(299792458 % 37)
    32
```

Comparative Operators

Operator	Meaning
==	Equal
!=	Not equal
>	Greater than
>=	^^ or equal to
<	Less than
<=	^^ or equal to

Equality


```
>>> i = 7
>>> j = 10
>>> print_(i == j)
    False
>>> print_(i != j)
    True
>>> print_(i < j)
    True
>>> print_(i >= j)
    False
```

Comparing strings and lists


```
>>> print_("Alberto" == "Alberto")
    True
>>> print_("Aaron" > "Aardvark")
    True
>>> print_("Bob" < "bob")
    True
>>>
>>> print_([1,2,3] >= [1,1,3,1])
    True
```

Logical Operators

Operator	Logical Equivalent
not	NOT
and	AND
or	OR
^	XOR

NOT

	Output
0	True
1	False

AND

		Output
0	0	False
0	1	False
1	0	False
1	1	True

OR

		Output
0	0	False
0	1	True
1	0	True
1	1	True

XOR

		Output
0	0	False
0	1	True
1	0	True
1	1	False

Logical operators


```
>>> print_(bool(1 and 1))
    True
>>> print_(bool("" or []))
    False
>>> print_(bool(not "this"))
    False
>>> print_(bool(True ^ False))
    True
```

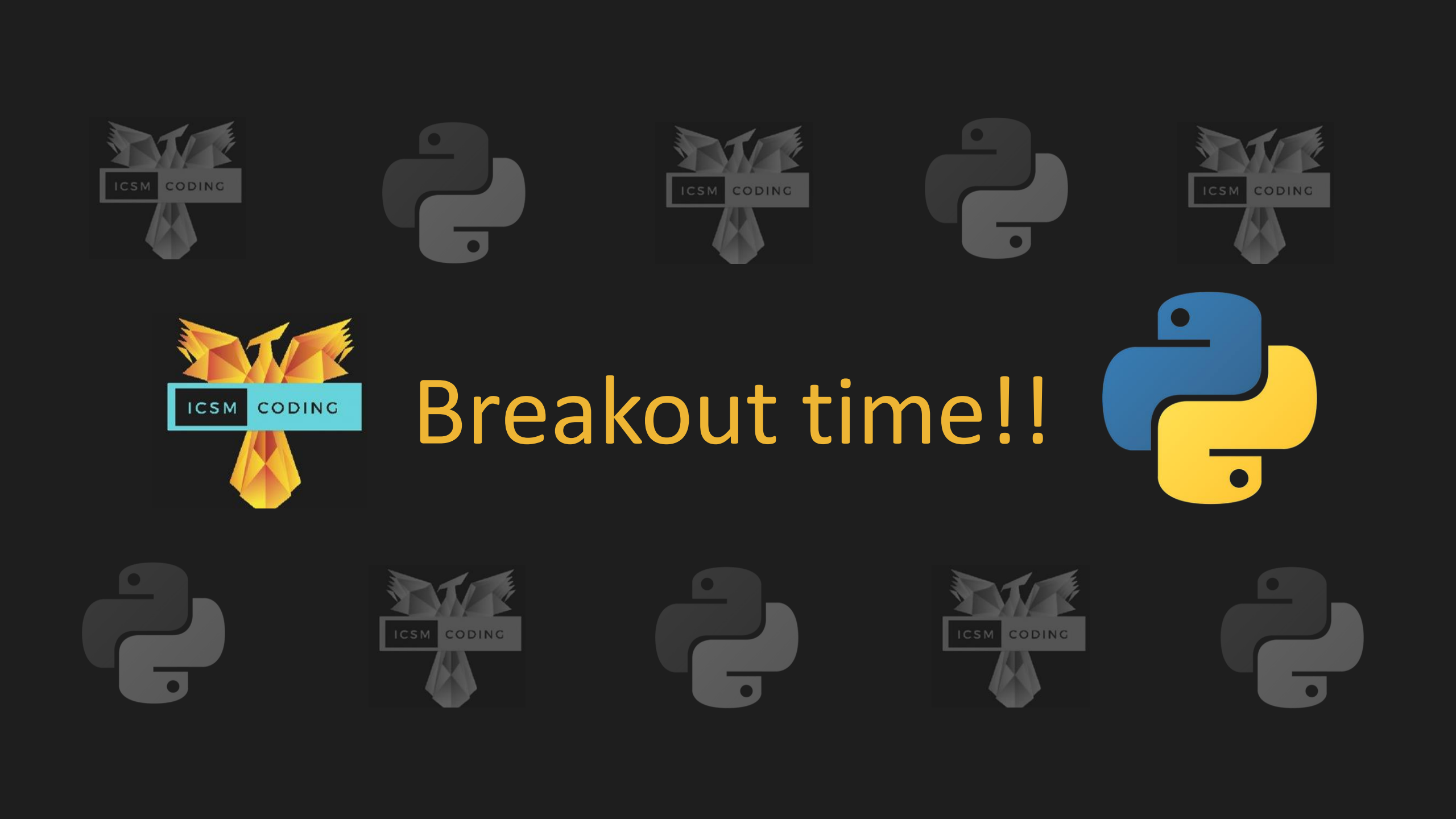
Pop Quiz!!

16 % 3

20 // 3

2 ** 6 < 64

"apples" ^ "oranges"


ICSM CODING
Breakout time!!

Loops and Flow

for loops

Variable name

Container

```
for ___ in ___ :
    .....
    .....
```

Code to be executed

for loops


```python
for i in [1, 2, 3]:
    print(i)
```

```python
name = "Chere"
for letter in name:
    print(letter)
```

```python
for element in ("subtle", "tuple"):
    print("rebutle", element)
```

```python
pressure = { sys : 120, dia : 80 }
for key in pressure:
    print(key, pressure[key])
```

range( )


```
my_list = []
for i in range(7):
    my_list = my_list + [1]

print(my_list)
```

```
[1, 1, 1, 1, 1, 1, 1]
```

```
my_img = []
for i in range(8):
    row = []
    for j in range(8):
        row = row + [j]
    my_img = my_img + [row]
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
[0, 1, 2, 3, 4, 5, 6, 7]
[0, 1, 2, 3, 4, 5, 6, 7]
[0, 1, 2, 3, 4, 5, 6, 7]
[0, 1, 2, 3, 4, 5, 6, 7]
[0, 1, 2, 3, 4, 5, 6, 7]
[0, 1, 2, 3, 4, 5, 6, 7]
[0, 1, 2, 3, 4, 5, 6, 7]
```

Generators


```python
my_list = [1 for x in range(7)]
print(my_list)
```

```
[1, 1, 1, 1, 1, 1, 1]
```

What will this output?

```python
[[0 for i in range(8)] for j in range(8)]
```

Pop Quiz!!

Which of these outputs [1, 2, 1, 2, 1, 2]?

```
for i in range(3): print([1, 2])
```

```
output = []
for i in [1, 1, 1]:
    for j in [2, 2, 2]:
        output += [i, j]
```

```
output = [2 if i % 2 else 1 for i in range(6)]
```

While Loops

Condition

```
while ___ :
    .....
    .....
```

Code to be executed

While Loops

Usage

```
print("forever")
while True:
    print("and ever")

print("this will never run")
while False:
    print("pointless")
```

```
forever
and ever
and ever
and ever
and ever
and ever
and ever
and ever
```

While Loops

Split these names up

```
names = "Vixen Prancer Dancer Comet"
split_names = []
while names:
    split_names += [""]
    while names and names[0] != " ":
        split_names[-1] += names[0]
        names = names[1:]
    names = names[1:]
```

```
['Vixen', 'Prancer', 'Dancer', 'Comet']
```

While Loops: input( )

```
prompt= ""
while prompt != "end":
    output = ""
    prompt = input("please type: ")
    for letter in prompt:
        output = letter + output
    print("answer:", output)
```

```
please type:  absolute
answer: etulosba
please type:  peng
answer: gnep
please type:  ting
answer: gnit
please type:  end
```

if .....

Condition

```
if ____ :

    .....

    .....
```

Code to be executed

if .....

Usage

```
if 10 == 5 + 5:
    print("Trivial")

if 9 + 10 == 21:
    print("I miss vine :-(")
```

if .....

Which medication?

```
while True:
    patient_age = float(input("how old? "))

    if patient_age < 55:
        print("this one needs an ACEi!")

    if patient_age >= 55:
        print("this one needs diuretics!")
```

```
how old?  27
this one needs an ACEi!
how old?  99
this one needs diuretics!
how old?
```

break

- Halts the execution of a loop

```
while True:
    patient_age = float(input("how old? "))

    if patient_age < 55:
        print("this one needs an ACEi!")

    if patient_age >= 55:
        print("this one needs diuretics!")

    if input("continue? (y / n)") == "n":
        print("exiting...")
        break
```

```
how old?  97
this one needs diuretics!
continue? (y / n) y
how old?  22
this one needs an ACEi!
continue? (y / n) n
exiting...
```

Continue

- Continues the execution of a loop

```
hr = [60, 66, 61, 58, 55, 56, 61, 65, 66]
for n in hr:
    if n >= 60:
        continue
    print(n)
```

```
58
55
56
```

if ___else___

Condition

```
if ___ :
    .....
    .....
else :
    .....
    .....
```

Executed if condition true

Executed if condition false

if ___else___

Usage

```
if True:
    print("This is true")
else:
    print("never gets here")
```

```
if 1 == 29:
    print("Does it?")
else
    print("of course not")
```

Ternary / Inline

```
patient_age = 50
print("Ultrasound" if patient_age < 45 else "Mammography")
```

Pop Quiz!!


```
number = int(input())
if number < 100:
    if number > 90:
        print("this?")
    else:
        print("or maybe this??")
else:
    if number < 1000:
        print("how about this?")
    else:
        print("nah mate this")
```

How do I get here?

if ___else___


```
text = "how many vowels/consonants there are in this sentence?"
vowels = 0
consonants = 0

for letter in text:
    if letter in "aeiou":
        vowels = vowels + 1
    else:
        consonants = consonants + 1

print("there are", vowels, "vowels and", consonants, "consonants")
```

```
there are 16 vowels and 38 consonants
```

if ___elif___else___

Condition

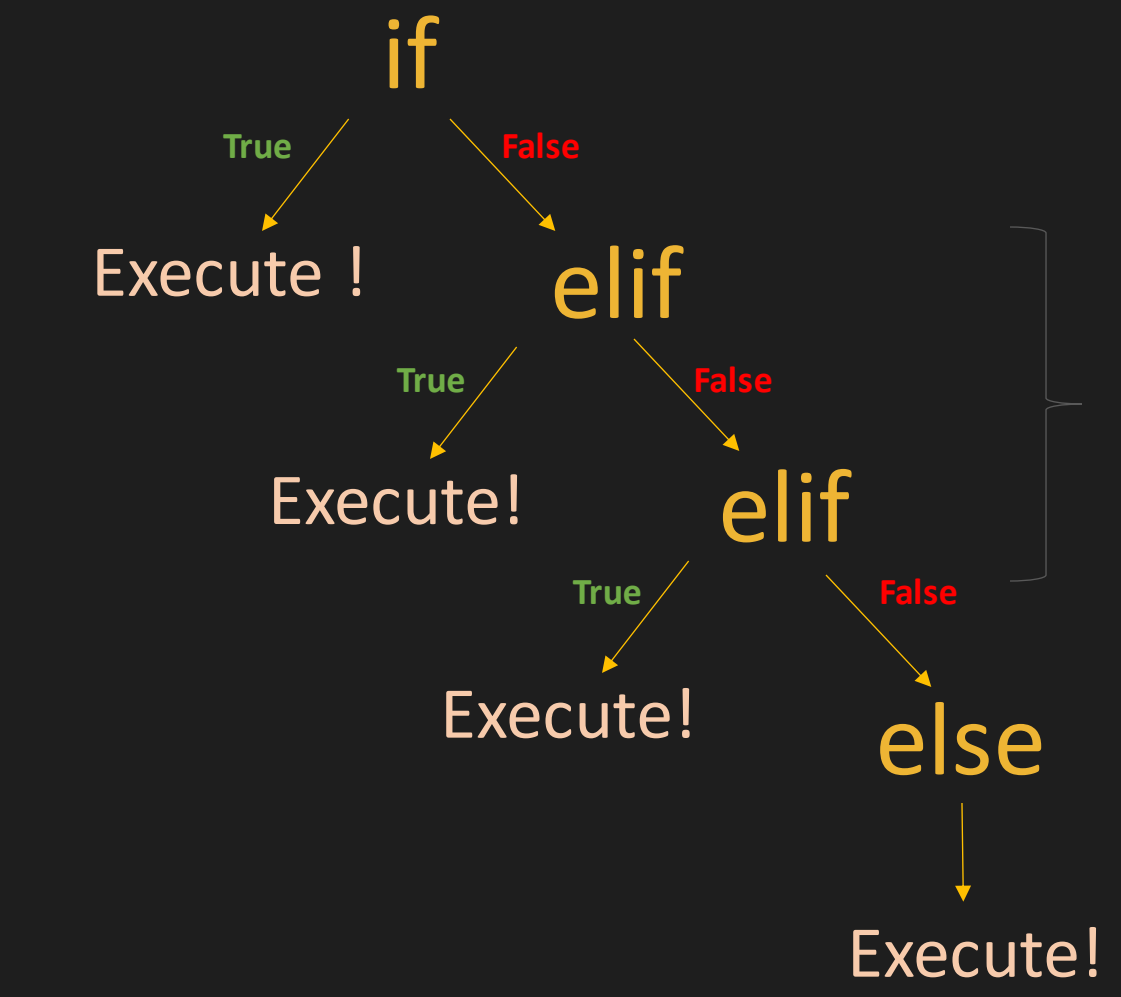
```
if ___ :
    .....
elif ___ :
    .....
else :
    .....
    .....
```

Executed if condition true

Executed if elif condition true

Executed if none are true

if___elif___else___



Can have as many of these as you like

if___elif___else___


```
sodium = float(input("what is their sodium? "))

if sodium < 120:
    print("unlucky bro")
elif sodium < 135:
    print("grab some saline")
elif sodium < 145:
    print("put the kettle on")
elif sodium < 160:
    print("this guy salty af")
else:
    print("call the coroner")
```

```
what is their sodium?  192
call the coroner
```

Pop Quiz!!

Can you describe what these do??

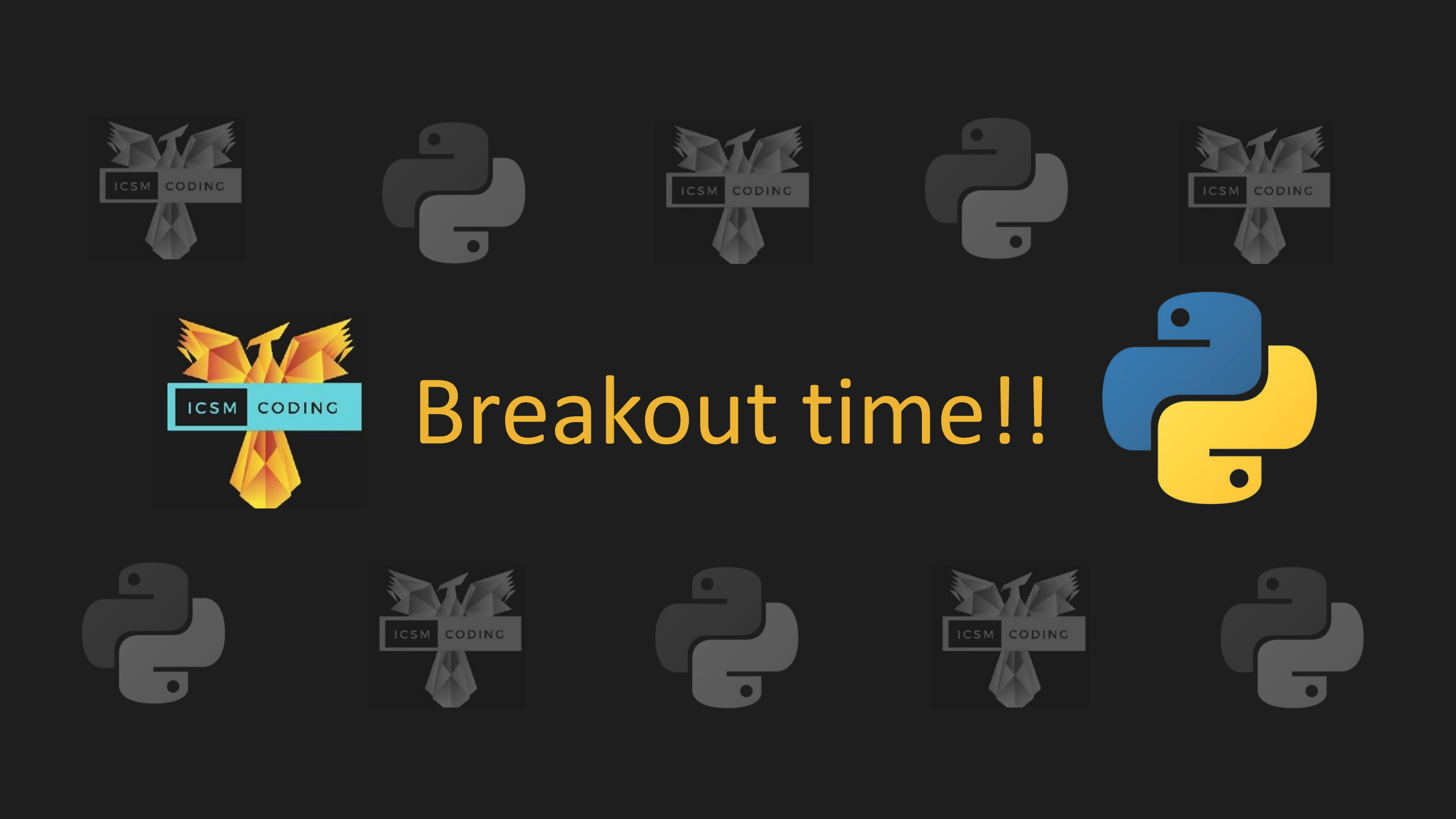
```
break
```

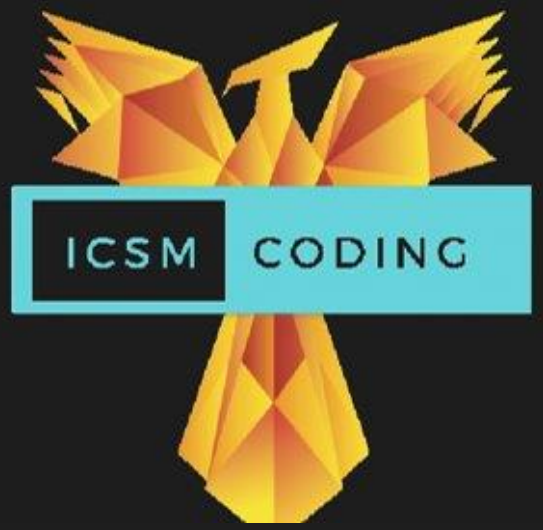
```
continue
```

Useful functions

print( )

input( )


ICSM CODING
Breakout time!!



Welcome back!!

What will we cover?

	Saturday 14/11/20
10:00 – 10:30	Intro / Getting Anaconda
15min Breakout!	Getting familiar with Jupyter
10:45 – 11:30	Variable, Types and Operators
30min Breakout!	Vs, Ts & Os worksheet
12:00 – 13:00	Lunch
13:00 – 13:45	Loops and Flow
45min Breakout!	Loops and Flow worksheet
14:30 – 14:45	Debrief / See you tomorrow

	Sunday 15/11/20
10:00 – 10:45	Welcome back / Functions
30min Breakout!	Functions Worksheet
11:15 – 12:00	Classes / Objects
30min Breakout!	Cs and Os Worksheet
12:30 – 13:30	Lunch
13:30 – 14:15	Example program
45min Breakout!	Make something!
15:00 – 15:15	Thanks

Functions

What is a function?

Mathematics

$$f(x) = 2x + 4$$

Python

```python
def f(x):
    return 2 * x + 4
```

What is a function?

Mathematics

$$f(x) = 2x + 4$$

$$f(9) = 22$$

Python

```python
def f(x):
    return 2 * x + 4
```

```python
print(f(9))
```

```
22
```

Function definitions

Function name

Arguments (optional)

```
def ___( ___ ):
    .....
    .....
    return____
```

Code to be executed

Optional return

Function definitions


```python
def greet(name):
    print("Salutations", name)

#anywhere after the function definiton
greet(input("good morning, what is your name?"))
```

```
good morning, what is your name? Albert
Salutations Albert
```

Built – in functions: len( )


```
print(len([1, 2, 3, 4, 5, 6]))
```

```
6
```

```
my_list = []
while len(my_list) < 6:
    my_list = my_list + [len(my_list) + 1]

print(my_list)
```

```
[1, 2, 3, 4, 5, 6]
```

min( ) max( )


```
def my_range(container):
    biggest = max(container)
    smallest = min(container)
    print("the range of this data is", biggest - smallest)

hr = [60, 66, 61, 58, 55, 56, 61, 65, 66]
my_range(hr)
```

```
the range of this data is 11
```

Pop Quiz!!

What will the output be??

```
min( max ( ["cannula",
            "defibrillator",
            "electorcardiogram",
            "ultrasound"]          ))
```

map( )


```python
#convert systolic/diastolic to MABP
def convert(pressure):
    MABP = (2 / 3) * pressure[0] + (1 / 3) * pressure[1]
    return int(MABP)

pressures = [(121, 82), (129, 72), (115, 97), (125, 60), (123, 77)]
list(map(convert, pressures))
```

```
[107, 110, 108, 103, 107]
```

type functions


```
_input_ = input("type anything")
print(str(_input_))
print(int(_input_))
print(float(_input_))
print(bool(_input_))
print(list(_input_))
print(tuple(_input_))
```

```
type anything 30
30
30
30.0
True
['3', '0']
('3', '0')
```

returns


```python
def var(data):
    mean = sum(data) /  len(data)
    square_diff = lambda n : (n - mean) ** 2
    return sum(map(square_diff, data))

def z_value(x, miu, stdev):
    z = (x - miu) / stdev
    return z

hr = [60, 66, 61, 58, 55, 56, 61, 65, 66]
hr_stdev = var(hr) ** 0.5
hr_mean = sum(hr) / len(hr)

print(z_value(80, hr_mean, hr_stdev))
```

Lambda*


```
def f(n):
    print(n ** 2)

f = lambda n : print(n ** 2)
```

equivalent

Pop Quiz!!

What will the output be??

```
def doubler(n):
    return 2 * n

tripler = lambda n : 3 * n

tripler( doubler( tripler( 1 )))
```

Back to generators

next( )

```
my_gen = (x + 42 for x in range(100))
print(next(my_gen))
print(next(my_gen))
print(next(my_gen))
print(next(my_gen))
print(next(my_gen))
```

```
42
43
44
45
46
```

Arguments

Positional

Key-word

```
def  func( a, b, c = 4, d = "hello" ) :
     .....
     .....
     .....
```

Positional arguments


```
def summarise(first, last, age, middle = "", home = "London"):
    name = first + " " + middle + " " + last
    print("the patient's name is", name)
    print(name, "lives in", home, "and is", age, "years old")
```

- Are position specific
- Must be provided

```
summarise("Elizabeth", "Windsor", 10000)

the patient's name is Elizabeth  Windsor
Elizabeth  Windsor lives in London and is 10000 years old
```

Key-word arguments


```
def summarise(first, last, age, middle = "", home = "London"):
    name = first + " " + middle + " " + last
    print("the patient's name is", name)
    print(name, "lives in", home, "and is", age, "years old")
```

- Come after positional in any order
- Are optional (default value provided)

```
summarise("Elizabeth", "Windsor", 10000, home = "Buckingham Palace")

the patient's name is Elizabeth  Windsor
Elizabeth  Windsor lives in Buckingham Palace and is 10000 years old
```

Accessing data


```python
a = 42
def scope_fun():
    z = 12
    a = 1

scope_fun()
print("a =", a)
print("z =", z)
```

```
a = 42
error
```

```python
a = [1,2,3]
def scope_fun():
    z = 12
    a = [9, 9, 9]

scope_fun()
print("a =", a)
print("z =", z)
```

```
a = [1, 2, 3]
error
```

```python
a = [1,2,3]
def scope_fun():
    z = 12
    a[:] = [9, 9, 9]

scope_fun()
print("a =", a)
print("z =", z)
```

```
a = [9, 9, 9]
error
```

Accessing data

- Any pre defined data may be accessed
- Try to avoid doing this
- Instead pass data as arguments...

Mutable vs Immutable arguments

value

```python
def func(n):
    n = n + " cake"
    print("n is", n)

a = "cheese"
func(a)
print("a is", a)
```

```
n is cheese cake
a is cheese
```

reference

```python
def func(n):
    n.append(1)
    print("n is", n)

a = [1,2,3]
func(a)
print("a is", a)
```

```
n is [1, 2, 3, 1]
a is [1, 2, 3, 1]
```

Pop Quiz!!

Key-Word

Vs

Positional

```python
def my_sum(container):
    s = 0
    for number in container:
        s = s + number
    return s
```

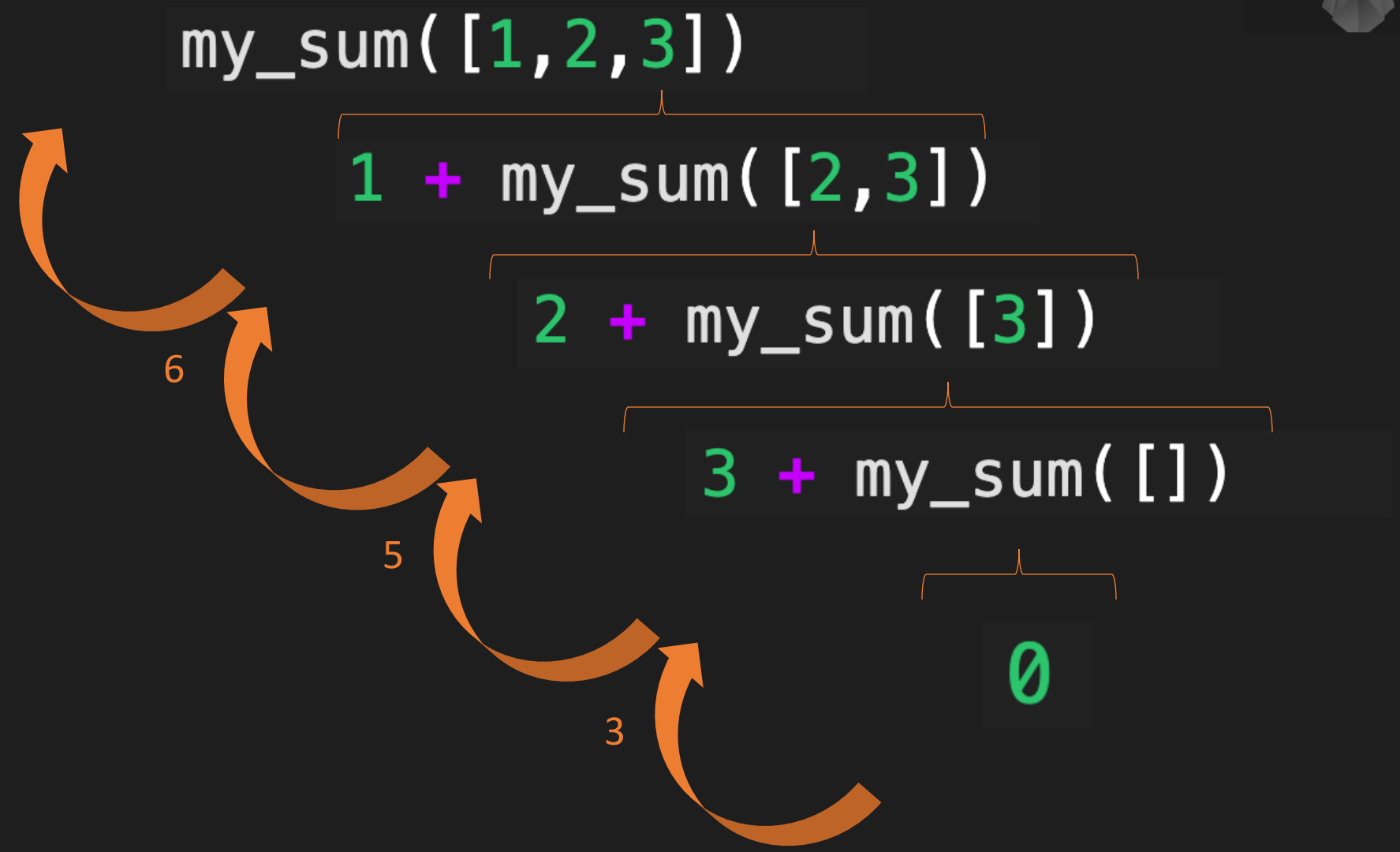
```python
def my_sum(container):
    if not container: return 0
    return container[0] + my_sum(container[1:])
```

Recursion!!

Recursion!!


```
my_sum([1,2,3])
    1 + my_sum([2,3])
        2 + my_sum([3])
            3 + my_sum([])
                0
```

Recursion!!

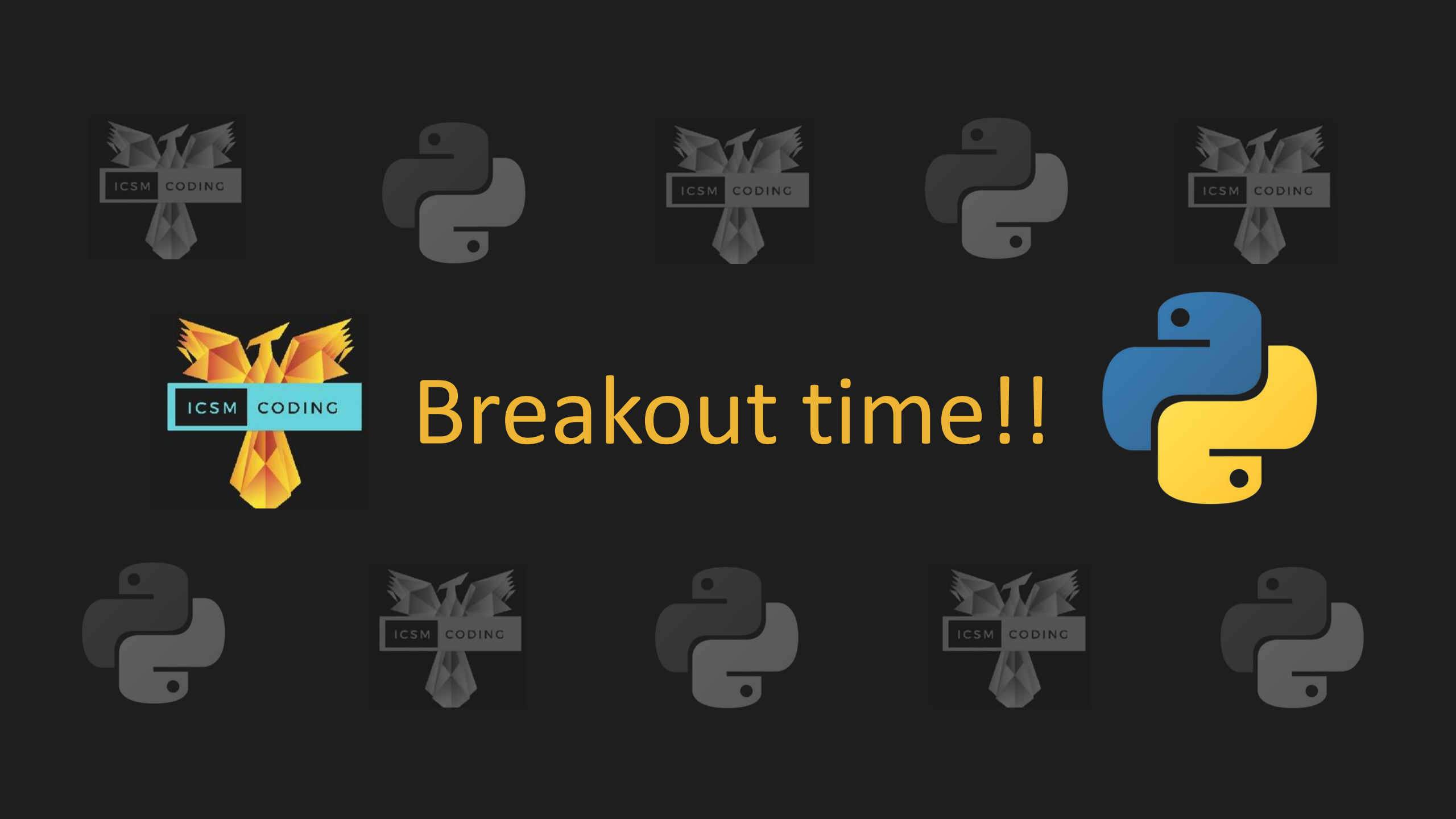


Recursion!!


```python
#calculate length of a list
def length(lst):
    if not lst: return 0
    return 1 + length(lst[1:])
```

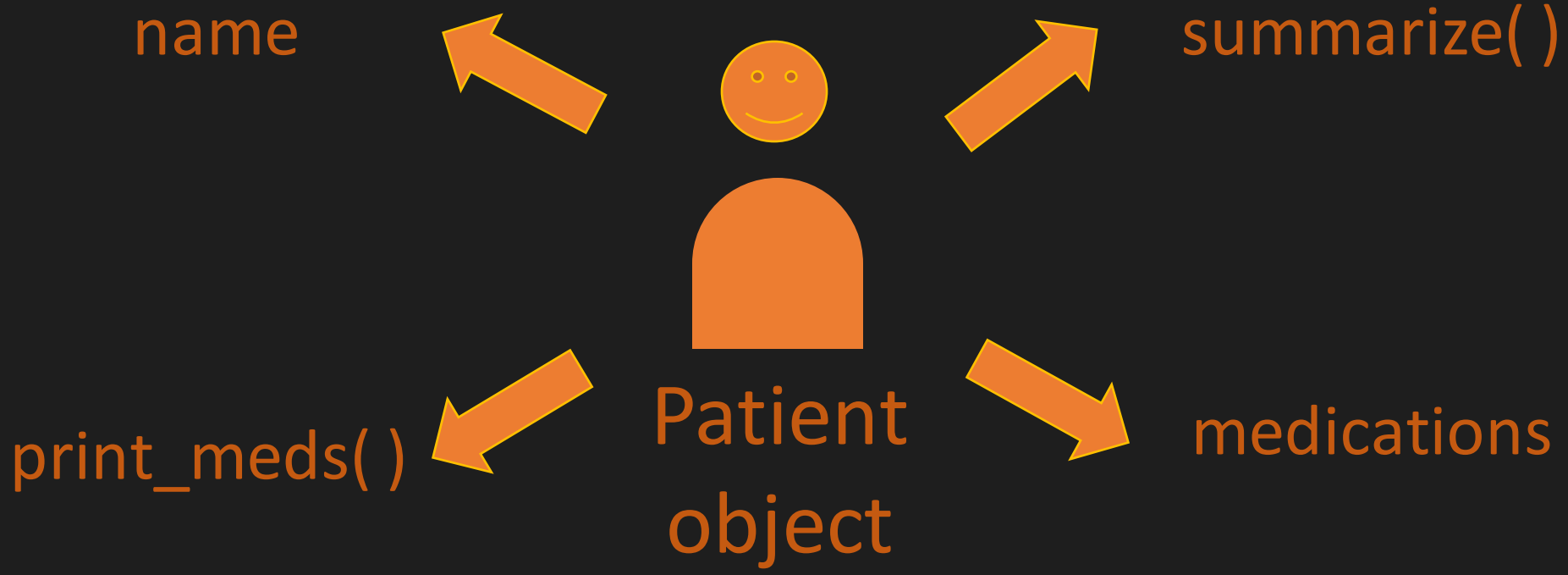
```python
#calculate factorials
def factorial(n):
    if not n: return 1
    return n * factorial(n - 1)
```

```python
#reverse a string
def reverso(text):
    if not text: return ""
    return text[-1] + reverso(text[:-1])
```


ICSM CODING
Breakout time!!

Classes and Objects

But what is an object?



String object

Example = "Hello World!"

Method / Attribute	Explanation
Example[…]	Get the letters in the string
Example.count(…)	Count occurrence of a letter
Example.find(…)	Get position of a letter
Example.strip(…)	Remove white space characters
Example.lower(…)	Convert to lowercase
Example.upper(…)	Convert to uppercase

String object


```
my_string = "    testing TESTING 123     "
print(my_string.strip())
print("the first g is at position", my_string.find("g"))
print("lowercase =", my_string.lower())
print("number of spaces is", my_string.count(" "))
```

```
testing TESTING 123
the first g is at position 10
lowercase =     testing testing 123
number of spaces is 11
```

split( ) and join( )


```
#lets do this the easy way now :-)
names = "Dancer Vixen Prancer Rudolf"

split_names = names.split(" ")
print("list of names:", split_names)

rejoined_names = ", ".join(split_names)
print("back to one string:", rejoined_names)
```

```
list of names: ['Dancer', 'Vixen', 'Prancer', 'Rudolf']
back to one string: Dancer, Vixen, Prancer, Rudolf
```

list object

Example = [2, 4, 6, 8, 10, 12]

Method / Attribute	Explanation
Example[…]	Get the elements of the list
Example.count(…)	Count occurrence of an element
Example.append(…)	Adds an element to the end
Example.pop(…)	Removes and returns an element
Example.reverse(…)	Reverses the list
Example.sort(…)	Sorts the list

list object


```python
my_list = [1, 23, 4, 5, 678, 9, 10]
new_list = []
for n in my_list:
    new_list.append(n + 1)

print(new_list)
```

```
[2, 24, 5, 6, 679, 10, 11]
```

```python
my_list = [1, 23, 4, 5, 678, 9, 10]
new_list = []
while my_list:
    new_list.append(my_list.pop(-1))

print(new_list)
```

```
[10, 9, 678, 5, 4, 23, 1]
```

Pop Quiz!!

Which method?

```
["lets", "turn", "this", "into", "that"]
```

```
"lets-turn-this-into-that"
```

dict object

Example = {"first" = "Jimmy", "second" = "Fallon"

Method / Attribute	Explanation
Example[…]	Get the elements of the dictionary
Example.items( )	Get each key value pair
Example.update(…)	Adds more key-value pairs
Example.pop(…)	Removes and returns an element
Example.clear( )	Clears the contents

dict object


```python
my_dict = {"first" : "Albert", "second" : "Einstein"}
my_dict.update( {"profession" : "Physicist"} )

for key, value in my_dict.items():
    print("key:", key, ", value:", value)
```

```
key: first , value: Albert
key: second , value: Einstein
key: profession , value: Physicist
```

Class declarations

Class name

```
Class ___ :
    def __init__ ( self, ___ ) :
        ......
        ......
    def ___ ( self, ___ ):
        ......
        ......
        ......
```

Initialization function

Class methods

__init__(self, ...)


```
def __init__ ( self,  arguments... ) :
        self.this = ....
        self.that = ....
```

__init__(self, ...)


```python
class Patient:
    def __init__(self, name, age, weight):
        self.name = name
        self.age = age
        self.weight = weight

some_guy = Patient("Maverick", 42, 242)

print(some_guy.name, "is", some_guy.age, "years old")
```

```
Maverick is 42 years old
```

Pop Quiz!!

What does the class declaration look like?

```
bob = Human(weight = 92, age = 47)
print(bob.age, bob.weight)
```

make your own methods

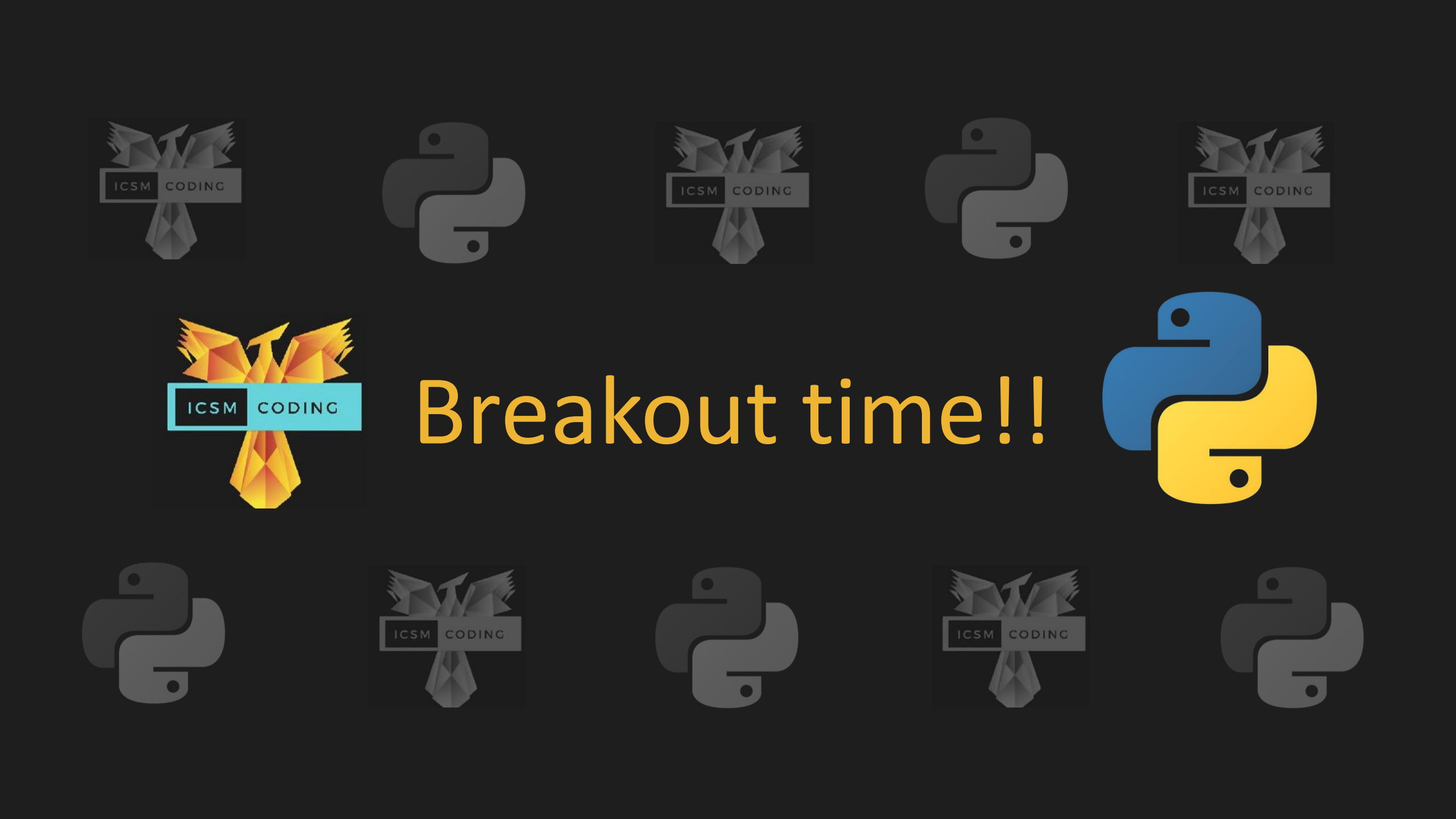

```python
class Patient:
    def __init__(self, name, age, hr_history = []):
        self.name = name
        self.age = age
        self.hr_history = hr_history

    def add_reading(self, new_hr):
        self.hr_history.append(new_hr)

    def get_average(self):
        return sum(self.hr_history) / len(self.hr_history)

some_guy = Patient("Maverick", 22, [60, 66, 75])
print(some_guy.get_average())

some_guy.add_reading(92)
print(some_guy.get_average())
```

```
67.0
73.25
```

Alternative Use


```python
class Medical_Functions:
    def BMI(weight, height):
        print( weight / (height ** 2), "kg/m2" )

    def MABP(systolic, diastolic):
        print( (2 / 3) * systolic + (1 / 3) * diastolic, "mmHg" )

    def VR(resp_rate, tidal_volume):
        print( resp_rate * tidal_volume, "dl/min" )

Medical_Functions.BMI(90, 1.87)
Medical_Functions.MABP(120, 80)
Medical_Functions.VR(12, 0.5)
```

```
25.73 kg/m2
106.67 mmHg
6.00 dl/min
```


ICSM CODING
Breakout time!!

Let solve a problem!

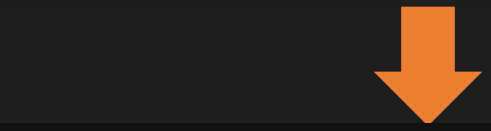

```
data = '''
Cristionna,Matthewson,2,5,119,111,101,129,153-
Bear,Ebbetts,46,8,154,110,159,152,123-
Tabby,Shouler,33, 5,159,156,159,118,166-
Mariana,Vasyutochkin,18,14,114,115,105,142,160-
Fanchette,McKeever,33,12,129,118,145,161,111-
Casey,Steinham,5,12,117,134,166,179,165-
Urbano,Petrolli,17,9,165,145,134,165,120-
Jim,Zarb,20,13,146,101,100,134,162-
Ursa,Lathleiffure,18,5,147,167,168,126,151-
Celestia,Plitz,0,6,117,120,177,130,102-
'''
```

```
Jim,Zarb,20,13,146,101,100,134,162-
```

name age glucose name

Split up individual patients


```
data = data.strip().split("-")
```



```
['Cristionna,Matthewson,2,5,119,111,101,129,153',
 'Bear,Ebbetts,46,8,154,110,159,152,123',
 'Tabby,Shouler,33, 5,159,156,159,118,166',
 'Mariana,Vasyutochkin,18,14,114,115,105,142,160',
 'Fanchette,McKeever,33,12,129,118,145,161,111',
 'Casey,Steinham,5,12,117,134,166,179,165',
 'Urbano,Petrolli,17,9,165,145,134,165,120',
 'Jim,Zarb,20,13,146,101,100,134,162',
 'Ursa,Lathleiffure,18,5,147,167,168,126,151',
 'Celestia,Plitz,0,6,117,120,177,130,102']
```

Make a list of lists


```
splitter = lambda string : string.split(",")
data = list(map(splitter, data))
```

```
[['Cristionna', 'Matthewson', '2', '5', '119', '111', '101', '129', '153'],
 ['Bear', 'Ebbetts', '46', '8', '154', '110', '159', '152', '123'],
 ['Tabby', 'Shouler', '33', ' 5', '159', '156', '159', '118', '166'],
 ['Mariana', 'Vasyutochkin', '18', '14', '114', '115', '105', '142', '160'],
 ['Fanchette', 'McKeever', '33', '12', '129', '118', '145', '161', '111'],
 ['Casey', 'Steinham', '5', '12', '117', '134', '166', '179', '165'],
 ['Urbano', 'Petrolli', '17', '9', '165', '145', '134', '165', '120'],
 ['Jim', 'Zarb', '20', '13', '146', '101', '100', '134', '162'],
 ['Ursa', 'Lathleiffure', '18', '5', '147', '167', '168', '126', '151'],
 ['Celestia', 'Plitz', '0', '6', '117', '120', '177', '130', '102']]
```

Make a list of lists


```
print(data[0])
print(data[6][7])
```

```
['Cristionna', 'Matthewson', '2', '5', '119', '111', '101', '129', '153']
165
```

Make a dictionary of patient objects


```python
class Patient:
    def __init__(self, first, second, age, bp_series, glucose = 5):
        self.first = first
        self.second = second
        self.age = age
        self.bp_series = list(map(int, bp_series))
        self.diabetic = int(glucose) < 7

patient_objects = {}
for row in data:
    name = row[0] + " " + row[1]
    new_patient = Patient(row[0], row[1], row[2], row[4:], row[3])
    patient_objects.update( { name : new_patient } )

print(patient_objects["Bear Ebbetts"].age)
```

46

Add useful functions


```python
class Patient:
    def __init__(self, first, second, age, bp_series, glucose = 5):
        self.first = first
        self.second = second
        self.age = age
        self.bp_series = list(map(int, bp_series))
        self.diabetic = int(glucose) > 7

    def update_bp(self, bp):
        self.bp_series.append(bp)

    def get_average(self):
        return sum(self.bp_series) / len(self.bp_series)

    def summarise(self):
        print(self.first, self.second, "is a ", self.age, "year old", end = " ")
        print("diabetic" if self.diabetic else "non-diabetic", end = " ")
        print("with an average BP of", self.get_average(), "mmHG")
```

Add useful functions


```
patient_objects = {}
for row in data:
    name = row[0] + " " + row[1]
    new_patient = Patient(row[0], row[1], row[2], row[4:], row[3])
    patient_objects.update( { name : new_patient } )

patient_objects["Bear Ebbetts"].summarise()
```

```
Bear Ebbetts is a  46 year old diabetic with an average BP of 139.6 mmHG
```

Over to You!!

- Getting to grips with Patient class
- Adding more functionality
- Perhaps making cohort class?

Breakout time!!

Summary

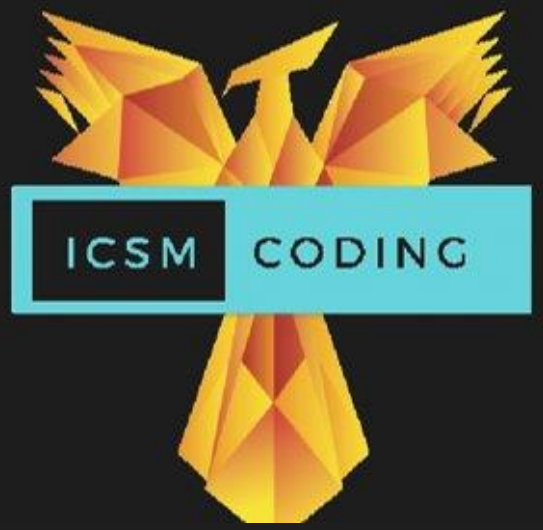
What we covered

Topic
Getting familiar with Jupyter
Variable, Types and Operators
Loops and Flow
Functions
Classes and Objects
Example Program

Plus....

- Upcoming panel event
- Python Tutorial series
- Check out Hackerrank and Codewars
- Feedback!!

- anu16@ic.a.uk



Thank you!!