# Welcome back to  !



**Learning Goals**

- Explain when conditionals are useful
- Implement conditionals
- Explain when loops are useful
- Implement for loops

# Types of data in Python

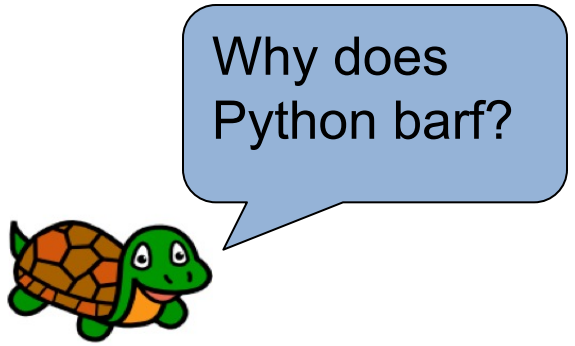
```
>>> best_food = "spam"
>>> ok_food = 'chocolate'
>>> good_num = 42
>>> pi = 3.1415926
>>> special = [2.718, 3.141, 42]
>>> 'chocolate' = ok_food
>>> 42 = good_num
```

String (single or double quotes work)

Integer (int)

"Floating point" number (float)

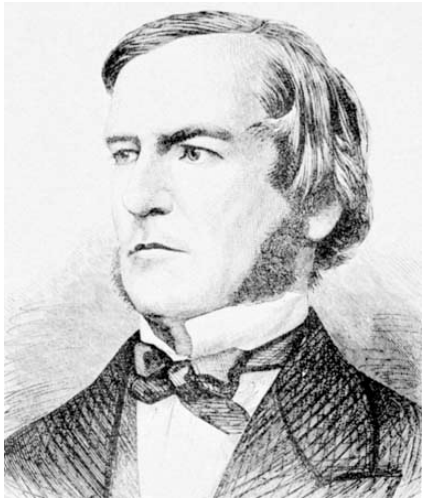
List (list)



# Booleans

```
>>> 3 == 1+2
True

>>> 1+2 == 3
True

>>> 42 == "spam"
False

>>> [1, 2, 3] == [1, 2, 3]
True

>>> [1, 2, 3] == [3, 2, 1]
False

>>> 42 > 5
True

>>> "A" != "G"
True

>>> not "A" == "G"
True
```

Notice the ==



George Boole
1815-1864

**Relational Operators**

== is equal to
!= is not equal to
> is greater than
< is less than
>= is greater than or equal to
<= is less than or equal to

# if statements

```
def special(x):
    """This function demonstrates the use
    of if statements"""
    if x == 42:
        return "Nice!"
```

the docstring

Notice the indentation here!

```
>>> special(42)
"Nice!"

>>> special(43)
```

**Syntax Cheatsheet**

```
if CONDITION:
    BODY1
```

# if, else…

```
def special(x):
    """This function demonstrates the use
    of if and else"""
    if x == 42:
        return "Nice!"
    else:
        return "Yuck!"
```

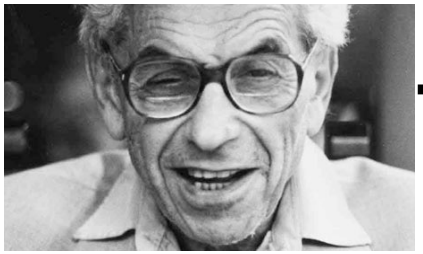
the docstring

Notice the indentation here!

```
>>> special(42)
"Nice!"

>>> special(43)
"Yuck!"
```

**Syntax Cheatsheet**

```
if CONDITION:
    BODY1
else:
    BODY2
```



Paul Erdos

# The Collatz Conjecture

the docstring

```
def collatz(n):
    """Returns n/2 if n is even and
    returns 3n+1 otherwise"""

    if n % 2 == 0:    # if n is even...
        return n/2
    else:
        return 3*n + 1
```

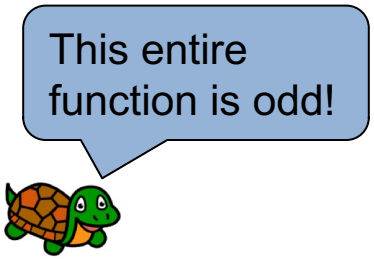
a comment

# if, elif, else…

```
def special(x):
    """This function demonstrates the use
    of if, elif, and else"""
    if x < 42:
        return "Silly little number!"
    elif x == 42:
        return "Secret to all happiness!"
    else:
        return "Wow, that's big!"
```

# Nested `if, elif, else…`

```
def special(x):
    """This function demonstrates the use
    of if, elif, and else"""
    if x < 42:
        if x % 2 == 0:
            return "Silly small even number"
        else:
            return "Silly small odd number"
    elif x == 42:
        return "Secret to all happiness!"
    else:
        if x % 2 == 0:
            return "Big & even"
        else:
            return "Big & odd"
```



# Getting things in order…

```
def schedule(hour):
    """Describes your day
    in military time"""
    if hour >= 2 and hour <= 9:
        return "Sleep"
    elif hour <= 17:
        return "In class"
    elif hour <= 20:
        return "Hang out"
    else:
        return "Do CS homework"
```

**Boolean Operators**

`and` : True if both are True

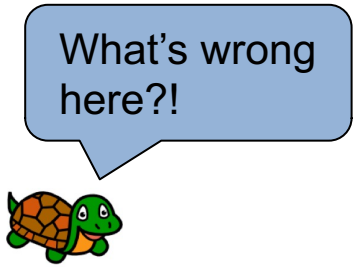
`or` : True if at least one is True

`not` : True if argument is False

# Getting things in order…

```
def schedule(hour):
    """Describes your day
    in military time"""
    if hour >= 2 and hour <= 9:
        return "Sleep"
    elif hour <= 17:
        return "In class"
    elif hour <= 20:
        return "Hang out"
    else:
        return "Do CS homework"
```

```
def schedule(hour):
    """Describes your day
    in military time"""
    if hour >= 2 and hour <= 9:
        return "Sleep"
    elif hour <= 20:
        return "Hang out"
    elif hour <= 17:
        return "In class"
    else:
        return "Do CS homework"
```



# A Cheesy Example…


```
def special(x):
    if x == 42:
        dinner = "spam"
    else:
        dinner = "mac & cheese"
    return dinner
```

`return` is strong stuff! When Python sees it, the function is done!

## Alternatively??

Option 1

```
def special2(x):
    if x == 42:
        dinner = "spam"
    dinner = "mac & cheese"
    return dinner
```

Option 2

```
def special3(x):
    if x == 42:
        dinner = "spam"
        return dinner
    dinner = "mac & cheese"
    return dinner
```

**Worksheet:** Which of these two alternatives does the same thing as the special function above? Why?

# Python Gets Loopy!

The Python,
Busch Gardens Florida

`loopy` is a function, but it does not expect any input!

We choose the name of a variable…

… and we provide a list

```
def loopy():
    sum = 0
    for my_number in [20, 17, 5]:
        sum = sum + my_number
    return sum
```

```
>>> loopy()
42
```

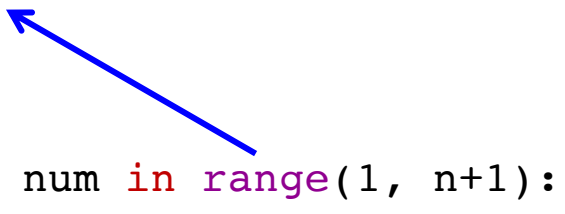
# Carl Friedrich Gauss
# 1777-1855

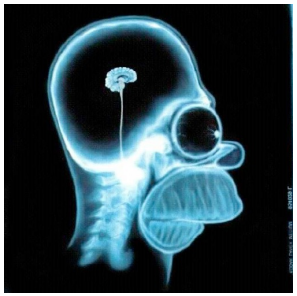
# If Gauss had Python…

**Goal: Add up 1 + 2 + … + n**

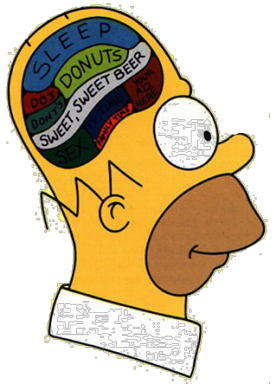
```
def gauss(n):
    """Returns 1 + 2 + … + n"""
    sum = 0
    for num in list(range(1, n+1)):
        sum = sum + num
    return sum
```

This is the list `[1,2,3,…,n]`





# The Anatomy of a for loop




```
for my_variable in list:
    Do all the stuff that is
    indented beneath the for loop

Stuff at this level of
indentation is done afterwards!
```

# Factorial…


**Goal: Take an integer n as input and return n!**

```
def factorial(n):
    """Returns the factorial of n"""
```

Try this (and the next two problems) in your **notes.**

In your notes

# Approximating e…


**Goal: Take a value n as input and return**
**1 + 1/1! + 1/2! + … + 1/n!**

```
def e(n):
    """Returns 1 + 1/1! + 1/2! + … 1/n!
    Assume that factorial(n) is available"""
```

In your notes

# A mystery…

Q

```
def mystery(n):
    for d in range(2, n):
        if n % d == 0:
            return False
    return True
```

For example, `7 % 2` is `1` (the remainder when the integer 7 is divided by the integer 2)

What is this function saying about its input `n`?

In your notes

# A perfect worksheet problem!

$6 = 1 + 2 + 3$ $28 = 1 + 2 + 4 + 7 + 14$

6, 28, 496, 8128, 33,550,336,

Known to ancient Greeks Nicomachus, 100 CE! 1456 CE

8,589,869,056, 137,438,691,328,

Ismail ibn Fallūs (1194-1252 CE) Pietro Cataldi, 1588 CE

# Achieving perfection!

Q

```
>>> perfect(6)
True

>>> perfect(7)
False

def perfect(n):
    """Returns True if n is perfect,
    False otherwise"""
    sod = 0
    for d in range(1, n):
        if n % d == 0:
            sod = sod + d
```

What is this doing? A few more lines of code ***in your notes*** and we are done!

# Example: Do pesticides affect bumble bees?

Concern about imidacloprid crop seed treatments potentially harming bumble bees

GROUP 4A INSECTICIDE

**Gaucho® 600 SC Insecticide**

For uses in pest management, suppression of insect vectored diseases and maintenance of plant health.

ACTIVE INGREDIENT:

Imidacloprid, 1-[(6-Chloro-3-pyridinyl)methyl]-*N*-nitro-2-imidazolidinimine ..... 48.7%

OTHER INGREDIENTS: ..... 51.3%

100.0%

EPA Reg. No. 264-828 EPA Est. No. 3125-MO-001

Contains 5 pounds of imidacloprid per gallon. SHAKE WELL BEFORE USING

Weighed bumble bees as they exited/entered nest

Photo credit: Richard Gill

Bumble bee collecting pollen

Photo credit: Dave Goulson

Feltham et al. (2014) Field realistic doses of pesticide imidacloprid reduce bumblebee pollen foraging efficiency. *Ecotoxicology*

```
massCollectedPest = [49,60,38,51,47]

massCollectedContrl = [40,36,37,35,32]

def mean(numList):
      """Returns the mean of
      a list of  numbers"""
      sum = 0
      count = 0
```

Q

Feltham et al. (2014) Field realistic doses of pesticide imidacloprid reduce bumblebee pollen foraging efficiency. *Ecotoxicology*

Name _____


**Worksheet:** Which of these two alternatives does the same thing as the original special function? Why?

Option 1

```
def special2(x):
    if x == 42:
        dinner = "spam"
    dinner = "mac & cheese"
    return dinner
```

Option 2

```
def special3(x):
    if x == 42:
        dinner = "spam"
        return dinner
    dinner = "mac & cheese"
    return dinner
```

---

```
def perfect(n):
    """Returns True if n is perfect,
    False otherwise"""
    sod = 0
    for d in range(1, n):
        if n % d == 0:
            sod = sod + d
```

What is this doing? A few more lines of code ***in your notes*** and we are done!