Watch by Thursday, October 8, 2020 | **Lesson #2**

# Math, Variables, and Strings

OCEAN 215 | Autumn 2020

Ethan Campbell and **Katy Christensen**

# What we'll cover in this lesson

1. Mathematical operations

2. Variables

3. Strings

# What we'll cover in this lesson

1. **Mathematical operations**
2. Variables
3. Strings

# Python can do math

Arithmetic Operators

| Operation | | Examples | |
|---|---|---|---|
| + | Addition | 2+2 | 4 |
| - | Subtraction | 4−2 | 2 |
| * | Multiplication | 4*2 | 8 |
| / | Division | 8/2 | 4 |
| ** | Exponential | 2**4 | 16 |
| % | Remainder | 16%5 | 1 |
| // | Floor | 16//5 | 3 |

**Just like a calculator!**

# Python can do math

Arithmetic Operators

| Operation | |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| ** | Exponential |
| % | Remainder |
| // | Floor |

**Example: Use these operations to do unit conversions**

Inches $\xrightarrow{\text{2.54 cm/in.}}$ Centimeters

(# of in.) * 2.54

^ Sorry this was wrong in the Panopto video! It should be multiplication, not division.

Kilometers $\xrightarrow{\text{10**5 cm/km}}$ Centimeters

(# of km) * (10**5)

# A note about parentheses...

Python follows the operation order:

**P**arentheses

**E**xponents

**M**ultiplication/**D**ivision + Remainder and Floor

**A**ddition/**S**ubtraction

| | |
|---|---|
| 4 + 4 ** 4 + 4 | 264 |
| (4 + 4) ** 4 + 4 | 4100 |
| 4 + 4 ** (4 + 4) | 65540 |
| (4 + 4) ** (4 + 4) | 16777216 |

**If in doubt, put more parentheses around an operation!**

Make sure to close your parentheses: ((4 + 4) ** (4 + 4)

```
SyntaxError: unexpected EOF while parsing
```

# What we'll cover in this lesson

1. Mathematical operations
2. **Variables**
3. Strings

# Use variables to keep information

A variable is a name attached to an object that can be called on later.

Example:

```
2+2
```

```
4
```

**Guide to naming a variable:**

1) Contains only alphanumeric characters (A-Z, 0-9) or underscore ( _ )

2) No spaces

3) Cannot start with a number

4) Variables (and Python in general) are case sensitive

5) Avoid "Camel Case" : numberOfStudents

6) Using informative names can prevent confusion

(only use single letters if the meaning is clear)

**Variable name (left)** **Desired information (right)**

```
my_var = 2 + 2
print(my_var)

my_var2 = my_var*2
print(my_var2)
```

```
4
8
```

# Types of objects

1. Numbers
2. Booleans
3. Strings
4. Structures

# Types of objects

1. **Numbers**
2. Booleans
3. Strings
4. Structures

Integer (int):

a whole number, without decimals

```
my_int_sml = 1
my_int_med = 492
my_int_lrg = 12349876
```

Floating Point Number (float):

a number containing at least one decimal

```
my_float_sml = 1.0
my_float_med = 567.51234
my_float_lrg = 12e15
```

$12 \times 10^{15}$

Complex Number (complex):

a number containing an imaginary part

```
my_complex_sml = 1 + 1j
my_complex_med = 32.5 + 15.2j
```

# Types of objects

1. **Numbers**
2. Booleans
3. Strings
4. Structures

Arithmetic operators can be applied to all variables that are numbers.

| Operation | |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| ** | Exponential |
| % | Remainder |
| // | Floor |

```
# Create variables with numbers
my_number1 = 53124
my_number2 = 97568

# Add and subtract the variables
print( my_number1 + my_number2 )
print( my_number1 - my_number2 )
```

```
150692
-44444
```

# Types of objects

1. **Numbers**
2. Booleans
3. Strings
4. Structures

The value of a variable can be altered using assignment operators.

| Operation | |
|---|---|
| += | Addition |
| -= | Subtraction |
| *= | Multiplication |
| /= | Division |
| **= | Exponential |
| %= | Remainder |
| //= | Floor |

```
my_number3 = 492

# Add 10 to the number
my_number3 += 10

print(my_number3)
```

```
502
```

**my_number3 = my_number3 + 10**

This overwrites the original number and saves the new one in its place

# Types of objects

1. **Numbers**
2. Booleans
3. Strings
4. Structures

Numbers are essential to data and our understanding of the world.

| Oceanographic Numbers | | | |
|---|---|---|---|
| **Time** (s) | **Populations** (count) | **Distances** (km) | **And so much more!!!** |
| **Temperature** (˚C) | **Current Speeds** (m/s) | **Fish Length** (cm) | |
| **Salinity** | **Density** (kg/m$^3$) | **Oxygen Levels** (mol) | |
| **Chemical Composition** (g/kg) | **Chlorophyll Concentration** (µg/L) | **Lat/Lon** (˚) | |

# Types of objects

1. Numbers
2. **Booleans**
3. Strings
4. Structures

# Types of objects

1. Numbers
2. **Booleans**
3. Strings
4. Structures

Booleans (bool) are objects with values of True or False.

```
t_bool = True
f_bool = False

print(t_bool,f_bool)
```

**Notice that these are capitalized**

```
True False
```

**Arithmetic operators can be used on a boolean, but it changes into an integer**

True = 1 False = 0

```
bool_math = (t_bool * 4) + f_bool

print(bool_math)
```

```
4
```

# Types of objects

1. Numbers
2. **Booleans**
3. Strings
4. Structures

Booleans (bool) are objects with values of True or False.

Comparison operators

| Operation | | Examples | |
|---|---|---|---|
| `==` | Equal | `5 == 5` | `True` |
| `!=` | Not Equal | `5 != 5` | `False` |
| `>` | Greater than | `4 > 10` | `False` |
| `>=` | Greater than or equal to | `14 >= 10` | `True` |
| `<` | Less than | `4 < 10` | `True` |
| `<=` | Less than or equal to | `10 <= 10` | `True` |

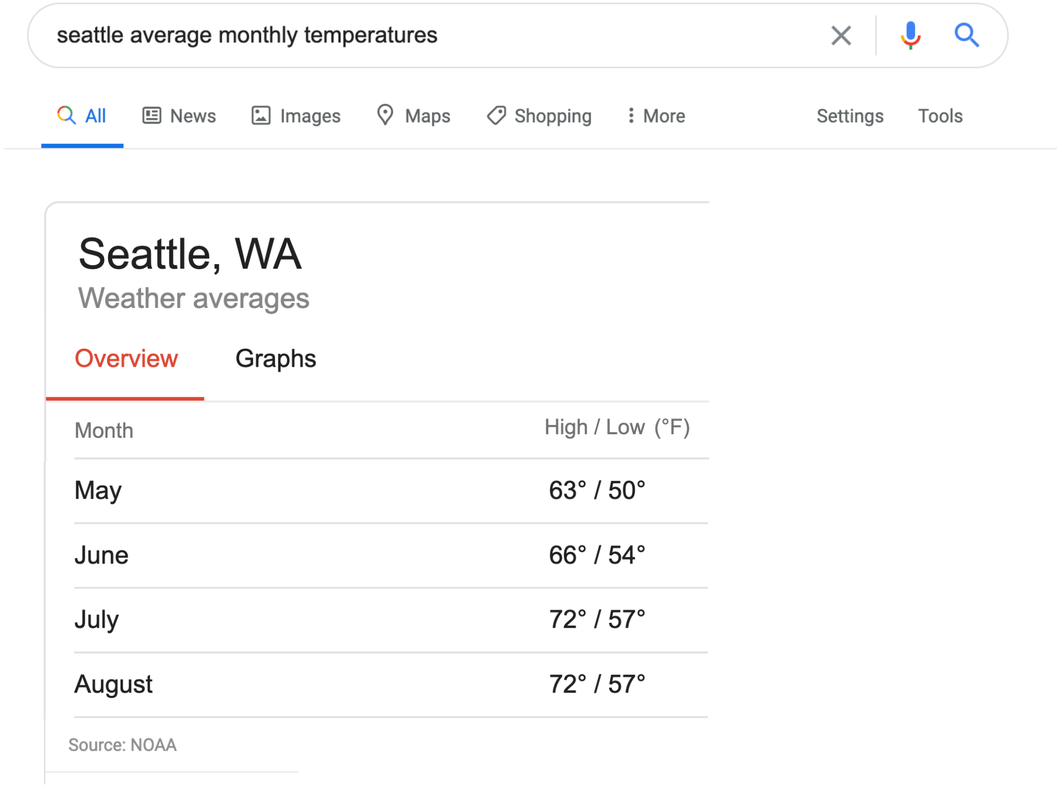
# Types of objects

1. Numbers
2. **Booleans**
3. Strings
4. Structures

Booleans (bool) are objects with values of True or False.

Comparison operators

| Operation | |
|---|---|
| == | Equal |
| != | Not Equal |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |



# Types of objects

1. Numbers
2. **Booleans**
3. Strings
4. Structures

Booleans (bool) are objects with values of True or False.

Comparison operators

| Operation | |
|---|---|
| == | Equal |
| != | Not Equal |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |

```
# Average high temperature in Seattle (˚F)
T_may = 63
T_jun = 66
T_jul = 72
T_aug = 72


print(T_may >= T_jun)
print(T_jun < T_aug)
print(T_jul == T_aug)
```

```
False
True
True
```

# What we'll cover in this lesson

1. Mathematical operations

2. Variables

3. **Strings**

# Types of objects

1. Numbers
2. Booleans
3. **Strings**
4. Structures

Strings (str) contain text information.

```
string_hws = 'Hello world!'
string_hwd =  "Hello world!"

print(string_hws)
print(string_hwd)
```

**Single quotes**

**Double quotes**

```
Hello world!
Hello world!
```

You need the same kind of quote on the beginning and end of the string

```
string_bad1 = 'Hello world!"
string_bad2 = 'Hello world!
```

```
SyntaxError: EOL while scanning string literal
```

# Types of objects

1. Numbers
2. Booleans
3. **Strings**
4. Structures

Numbers can be strings too, but you cannot do arithmetic with them.

```
string_number = '32'

print(string_number)
```

```
32
```

```
print(string_number - 2)
```

```
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

# Types of objects

1. Numbers
2. Booleans
3. **Strings**
4. Structures

**Concatenate:**
**combining strings**

**+**

```
hi = 'Hello'
wld = 'world!'
spc = ' '

print(hi+spc+wld)
```

```
Hello world!
```

**A space between quotes**

**Duplicate:**
**repeating strings**

*****

```
hi = 'Hello'

print(hi*4)
```

```
HelloHelloHelloHello
```

# Types of objects

1. Numbers
2. Booleans
3. **Strings**
4. Structures

To put certain characters in a string, an escape sequence ( \ ) is needed.

| What you want | What you type |
|---|---|
| \ | \\ |
| “ | \” |
| ‘ | \’ |

**These can be avoided by using different quotes than your string identifying quotes**

```
print('Double "quotes" inside single quotes')
print("Single 'quotes' inside double quotes")
print()
print('Single \'quotes\' inside single quotes')
print("Double \"quotes\" inside double quotes")
```

```
Double "quotes" inside single quotes
Single 'quotes' inside double quotes

Single 'quotes' inside single quotes
Double "quotes" inside double quotes
```

# String indexing and slicing

A string can contain any number of characters, as long as there are quotes around it.

Strings can be empty...

```
empty_string = ''

print(empty_string)
```

Or strings can be long. This means that strings have a dimension to them: length.

Use the **len()** function to find out how many characters are in a string!

```
test_string = 'the quick brown fox jumped over the lazy dog.'

# Get the length of the string
str_len = len(test_string)
print(str_len)
```

```
45
```

**Spaces are counted as characters**

# String indexing and slicing

How python counts characters (indexing):

| string = | P | y | t | h | o | n | | i | s | | f | u | n | ! |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

**Length = 14 characters**

| Individual characters | Character Groups | Whole string |
|---|---|---|
| string[0] | string[10:13] | string[:] |
| P | fun | Python is fun! |

You can select certain parts of a string by slicing it.

# String indexing and slicing

Example:

```
# This is the scientific name for the humpback whale
sci_name = 'Megaptera novaeangliae'
           0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

# Separate the string into genus and species names
genus = sci_name[0:9]
species = sci_name[10:]

print(genus)
print(species)

# Get just the first letter of the genus and the species
initials = genus[0] + species[0]
print(initials)
```

This is the same as sci_name[ 10 : 22 ]

```
Megaptera
novaeangliae
Mn
```

Image Source: https://www.washington.edu/news/2019/10/21/humpback-whale-population-on-the-rise-after-near-miss-with-extinction/

# String membership

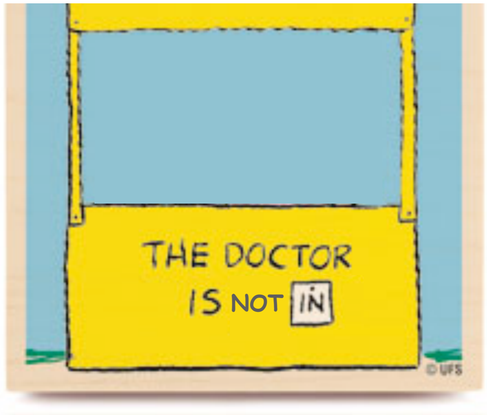
You can check if specific characters are in a string using the membership operators.


**in**

**True**

**not in**

**False**




**in**

**False**

**not in**

**True**

Image source: https://www.writermag.com/improve-your-writing/nonfiction/interview-like-psychiatrist/

# String membership

You can check if specific characters are in a string using the membership operators.



```
compound =

sulf_test = 'sulf' in compound
phos_test = 'phos' in compound
carb_test = 'carb' not in compound


print(sulf_test,phos_test,carb_test)
```

Image source: https://www.writermag.com/improve-your-writing/nonfiction/interview-like-psychiatrist/

# String membership

You can check if specific characters are in a string using the membership operators.



```
compound =

sulf_test = 'sulf' in compound
phos_test = 'phos' in compound
carb_test = 'carb' not in compound


print(sulf_test,phos_test,carb_test)
```

```
True False True
```

Image source: https://www.writermag.com/improve-your-writing/nonfiction/interview-like-psychiatrist/

# String membership

You can check if specific characters are in a string using the membership operators.



Example: DMSP $(CH_3)_2S^+CH_2CH_2COO^-$

```
compound = 'Dimethylsulfoniopropionate'

sulf_test = 'sulf' in compound
phos_test = 'phos' in compound
carb_test = 'carb' not in compound


print(sulf_test,phos_test,carb_test)
```

```
True False True
```

Image source: https://www.writermag.com/improve-your-writing/nonfiction/interview-like-psychiatrist/

# String functions

```
my_string = '        Apples and Bananas!!!!!!!!!!!!!'
```

| | | | |
|---|---|---|---|
| **lstrip** | Removes characters from the left side of the string (default: remove spaces) | `# Remove the spaces on the left side`<br>`my_string = my_string.lstrip()` | 'Apples and Bananas!!!!!!!!!!!!!' |
| **rstrip** | Removes characters from the right side of the string (default: remove spaces) | `# Remove the ! on the right side`<br>`my_string = my_string.rstrip('!')` | 'Apples and Bananas' |
| **upper** | Makes all letters in the string upper case | `# Capitalize the whole string`<br>`my_string_caps = my_string.upper()` | 'APPLES AND BANANAS' |
| **lower** | Makes all letters in the string lower case | `# Now make the whole string lower case`<br>`my_string_lows = my_string.lower()` | 'apples and bananas' |
| **count** | Counts the number of times a given character is in the string | `# Find how many a's are in the string`<br>`a_num = my_string_lows.count('a')` | 5 |
| **replace** | Replaces a given character with a different character | `# Replace all the a's with o's`<br>`my_string_o = my_string_lows.replace('a','o')` | 'opples ond bononos' |

# Resources used to create this lesson...

1. Python Operators: w3schools.com
2. Seattle average monthly temperatures: Google search
3. Megaptera Novaeangliae: A guide to the pronunciation and meaning of cetacean taxonomic names
4. Dimethylsulfoniopropionate (DMSP): Smithsonian Marine Microbes