```
print()

int/int = float

Booleans:
True
False
not True
not False

Equality Operators:
==
!=

if condition :
if condition and condition :
if condition or condition :
elif
else

while condition :

counter (i) from 0 to b - 1
for i in range (b)

counter (i) from a to b - 1
for i in range (a, b):

anything to string
str(anything)

repetition (#) times with anything
print(anything * #)

length of string
len(string)

finding the type of something
type(something)

indexing
[#]

slicing, making a substring from length a to b - 1
[a:b]

find "a" in a string and replace it with "b"
string.replace("a", "b")

to include ' in strings
"what\'re you doing?"

line break
\n

automatically include line breaks in a string
"""string"""

don't end line
\

appending to list (end of list, one value at a time)
list.append(element)

lists:
[“a”, “b”, “c”, “d”, “e”]

tuples: faster lists, but values can't change
(a, b, c, d, e)
```

```
tuple to list
list(tuple)

list to tuple
tuple(list)

empty list
let(list) == 0

function
def name(parameters):

dictionary: unsorted lists
numbers = {"one", "un", "two": "deus"}
numbers[3] = "trois"
numbers[True] = "quatre"

{'one', 'un', 'two': 'deus', 3: 'trois', True: 'quatre'}

    indexing
    numbers["one"]
    'un'

    returns keys
    numbers.keys()
    dict_keys(['one', 'two', 3, True])

    checks for a key:
    "one" in numbers
    True

    updating:
    a = {1, "un}
    dictionary.update(a)

    remove a value:
    dictionary.pop("key name")

Object Constructor:
class Car:
    wheels = 4
    def _init_(self, color):
        self.color = color
        self.running = False
    def start_engine(self):
        self.running = True
        print("Vroom")

    car_one = Car("red")
    print (car_one.color)

    "red"

Class Animal:
    def _init_(self, type):
        self.type = type

    inheriting a class
    Class Bear(Animal):
        def _init_(self, breed):
            Animal._init_(self, "mammal")
            self.breed = breed

import datetime
birthday = datetime.date(1998, 12, 16)

from datetime import date, time
from datetime import date as d
birthday = d(1998, 12, 16)
```

modules.py

f-string:
```
print(f“{2} new messages”)
2 new messages
```

```
new_message = 2
print(f”{new_messages}”)
2 new messages
```

```
-=
+=
```

removing all letters from a string:
```
import re
strEdit = re.sub('[abcdefghijklmnopqrstuvwxyz]', '', string)
```

removing all punctuations from a string:
```
import re
new_string = re.sub("[^a-zA-Z0-9]+", " ",string)
```

making a list from a string with split (from spaces)
```
list = string.split(' ')
```

adding an element to a specific index in a list (one element at a time)
```
list = [“a”, “b”, “c”, “d”]
list.insert(index, new_element)
```

if index 0 is chosen
```
[new_element, “a”, “b”, “c”, “d”]
```

deleting an element of a list:
```
list.pop(index)
```

```
grades = [1, 2, 3, 4, 5]
for score in grades:
    print(scores)
1
2
3
4
5
```

getting the greatest/lowest value in a list
```
max(list)
min(list)
```

sorting a list and changing its value to be sorted
```
list.sort()
```

sum of list
```
sum(list)
```

combining lists
```
lists = list1 + list2
```

finding how many times an element appears on a list
```
list.count(element)
```

finding if an element is in a list
```
element in list
```

function names should start with verbs (an action): create, get, compute, calculate

function names for boolean outputs: is, has, can

local scope: variables created inside functions (def)

global variables: variables not created inside functions (def)

reversing a string
string[::-1]
    Isabella Rocha
    ahcoR allebasI

```
print (True & True)
```

immutable: attempting to modify them results in an error

tuples are immutable, they can’t be updated, deleted, or add new value

dictionary variable team, with “team 1” as the key and [“chandler”, “joey”] as values. You can’t repeat keys

```
team = {
    “team 1”: [“chandler”, “joey”]
}
```

sets don’t allow duplicates

```
set = {a, b, c, d}
```

    adding to a set
    set.add(element)

    removing from a set
    set.remove(element)

converting list to set
set(list)

subset: when all elements of a set are in another set
we can check this
set1.issubset(set2)m

you can join sets and remove duplicates
set1.union(set2)

join sets only with their duplicate values
set1.intersect(set2)

getting the differences between sets, what set1 has that set2 doesn’t have
set1.difference(set2)

```
a = [10, 20, 34]
b = [number/2 for number in a]
5, 10, 17
```

```
class Name:
    variable = value
```

```
instance = Name()
instance.variable
```

classes can have methods (def) and they need to be passed in the parameter (self)

constructors: __init__()

```
class Dog:
    color = “pink”
    def __init__(self):
        self.color = “brown”
    def print_color(self):
        print(self.color)
```

```
rocky = Dog()
rocky.print_color()
brown
print(rocky.color)
pink
```

modules: import math, you can get more information with them with: help(math)

you can import more than one module with ‘,’
```
import statistics, math
```

import specific parts of a module
```
from math import pi
```

aliasing: modify the names if modules
```
import statistics as stats
```