Hello World

Today we're learning Python for beginner data analysts.

Five key topics that we need to know.

- variable
- data type
- data structure
- control flow
- function

```
print("hello world")
```

```
hello world
```

```
print("run me!")
```

```
run me!
```

```
print(1+1)
```

```
2
```

List

```
movies = ["The Dark Knight", "Superman", "Beekeper"]

for movie in movies:
    print(movie)
```

```
    The Dark Knight
    Superman
    Beekeper
```

```
1 ## same results as append
2 movies = movies + ["Titanic", "Avatar"]
3 print(movies)
```

```
    ['The Dark Knight', 'Superman', 'Beekeper', 'Titanic', 'Avatar']
```

```
1 int("150") * 2
```

```
    300
```

```
1 # python dynamic typed language
2 x = True
3 type(x)
```

```
    bool
```

```
1 # python type hint
2 name: str = "toy"
3 gpa: float = 3.45
4 age: int = 36
5 movie_lover: bool = True
6
7 print(name, gpa, age, movie_lover)
```

```
    toy 3.45 36 True
```

```
1 name:str = 100
2 print(name)
```

```
    100
```

```
1
```

Tuple

Immutable object

```
1 friends = ("toy", "john", "doe", "toy", "toy")
2 type(friends)
```

```
tuple
```

```
1 for name in friends:
2     print("hi! " + name)
```

```
hi! toy
hi! john
hi! doe
```

```
1 # cannot update value in tuple
2 friends[0] = "TOY"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-c1d0af9a30ec> in <cell line: 2>()
      1 # cannot update value in tuple
----> 2 friends[0] = "TOY"

TypeError: 'tuple' object does not support item assignment
```

SEARCH STACK OVERFLOW

```
1 friends.count("toy")
```

```
    3
```

```
## tuple unpacking
age_friends = (35, 30, 28)

toy, john, doe = age_friends

print(toy, john, doe)
```

```
    35 30 28
```

```
## def new function that returns more than one value
def add_two_nums(x, y):
    print("greeting")
    print("hello")
    return x+y

## test function
result = add_two_nums(10, 15)

print(result)
```

```
    greeting
    hello
    25
```

```
def create_books():
    return ("BSM", "MMS", "Coach Builder", "Freak")

b1, b2, _, b4 = create_books()

print(b1, b2, b4)
```

```
    BSM MMS Freak
```

```
1
```

Dictionary

key-value pair

```
customer01 = {
    "name": "toy",
    "age": 36,
    "city": "Bangkok"
}
```

```
## dict is mutable object
customer01["name"] = "John Wick"
customer01["age"] = 49
customer01["city"] = "New York"

customer01
```

```
{'name': 'John Wick', 'age': 49, 'city': 'New York'}
```

```
## add new key
customer01["country"] = "United States"
customer01
```

```
{'name': 'John Wick',
 'age': 49,
 'city': 'New York',
 'country': 'United States'}
```

```
for item in customer01.items():
    print(item[0], item[1])
```

```
name John Wick
age 49
```

```
    city New York
    country United States
```

```
1 customer01.get("name")
```

```
    'John Wick'
```

```
1 # delete key
2 customer01.pop("country")
3 customer01
```

```
    {'name': 'John Wick', 'age': 49, 'city': 'New York'}
```

```
1 # delete
2 del customer01["name"]
3
4 customer01
```

```
    ---------------------------------------------------------------------------
    KeyError                                  Traceback (most recent call last)
    <ipython-input-71-7d678f0eb0c4> in <cell line: 2>()
          1 # delete
    ----> 2 del customer01["name"]
          3
          4 customer01

    KeyError: 'name'
```

SEARCH STACK OVERFLOW

```
1
```

Set

```
## set in Python is unique
numbers = [1,1,2,3,4,5,5,6,9,9]

set(numbers)
```

```
{1, 2, 3, 4, 5, 6, 9}
```

```
package = ["post", "pre", "pre", "post", "terminate"]

set(package)
```

```
{'post', 'pre', 'terminate'}
```

```
set_a = {"banana", "apple"}
set_b = {"banana", "orange"}

# union
set_a | set_b

# intersect
set_a & set_b
```

```
{'banana'}
```

```
1
```

Control Flow

- if
- for
- while

```
# if else function
def grading(score):
    if score >= 80:
        return "A"
    elif score >= 70:
        return "B"
    elif score >= 60:
        return "C"
    else:
        return "Failed"
```

```
grade = grading(85)
print(grade)
```

```
A
```

```
score = 85

print("passed") if score >= 80 else print("failed")

if score >= 80:
    print("passed")
else:
    print("failed")
```

```
passed
passed
```

```
artists = ["Taylor Swift",
           "Spice Girls",
           "Backstreet Boys"]

# list comprehension (google)
upper_artists = [artist.upper() for artist in artists]

print(upper_artists)
```

```
['TAYLOR SWIFT', 'SPICE GIRLS', 'BACKSTREET BOYS']
```

```
# for loop
artists = ["Taylor Swift",
           "Spice Girls",
           "Backstreet Boys"]

for artist in artists:
    first_name = artist.split(" ")[0].upper()
    if first_name == "TAYLOR":
        print("Greeeeeed! I'm going to see you in Japan.")
    else:
        print(first_name)

```

```
Greeeeeed! I'm going to see you in Japan.
SPICE
BACKSTREET
```

```
# cost of program: time
nums = list(range(1, 10001)) # n-1
```

```
def sum_manual(nums):
    result = 0
    for num in nums:
        result += num
    return result

sum_manual(nums)
```

```
50005000
```

```
def sum_shortcut(nums):
    return (nums[0] + nums[-1]) * nums[-1] / 2

sum_shortcut(nums)
```

```
50005000.0
```

while loop

```
n = 1 # counter

while n < 5:
    print("hi!")
    n += 1 # update counter
```

```
hi!
hi!
hi!
hi!
```

```
def game():
    print("This is a test game!")
    while True:
        user_input = input("Do you want to continue or stop? ")
        if user_input == "stop":
            print("Game stop!")
            break
        else:
            print("Let's continue.")
```

```
game()
```

```
This is a test game!
Do you want to continue or stop? go
Let's continue.
```

```
Do you want to continue or stop? nope
Let's continue.
Do you want to continue or stop? fun
Let's continue.
Do you want to continue or stop? stop
Game stop!
```

Object Oriented Programming

```
class Book:
    def __init__(self, name, year, author):
        self.name = name
        self.year = year
        self.author = author
```

```
book1 = Book("Think lik a freak", 2010, "Dubner")
book2 = Book("Business Made Simple", 2018, "Donald Miller")
book3 = Book("Data Science for Business", 2015, "Wiley")
```

```
# dot notation
book3.year
```

```
2015
```

```
# OOP = Object Oriented Programming
# class Dog

class Dog:
    def __init__(self, name, age, specie):
        self.name = name
        self.age = age
        self.specie = specie

    def sitting(self):
        print("I'm sitting on the bed.")

    def get_older(self, year):
        self.age += year
        print(f"I'm getting older {year} year.")

dog1 = Dog("andy", 3, "chihuahua")
print(dog1.name, dog1.age, dog1.specie)
```

```
andy 3 chihuahua
```

```
dog1.sitting()
```

```
I'm sitting on the bed.
```

```
# Dog method
print(dog1.age)
dog1.get_older(2)
print(dog1.age)
```

```
3
I'm getting older 2 year.
5
```

```
import pandas as pd
```

```
## Homework 2
```

```
2 class DataAnalyst:
3     pass
4
5 ## 3 attributes
6 ## 3 functions (methods)
```