



Class Resumes at 02:35 PM

INTRO TO FUNCTIONS



BASIC FUNCTIONS

Out of the box, Python offers a bunch of built-in functions to make your life as a data scientist easier. You already know two such functions: `print()` and `type()`.

You've also used the functions `str()`, `int()`, `bool()` and `float()` to switch between data types. These are built-in functions as well

FAMILIAR FUNCTIONS - EXAMPLE

- Calling a function is easy. To get the type of 3.0 and store the output as a new variable, result, you can use the following:

```
result = type(3.0)
```

```
print(result)
```

- You also use combination of print(), function and type() function to get type of 3.0. you can use the following:

```
print(type(3.0))
```

TASKS

Use `print()` in combination with `type()` to print out the type of `var1` and `var2`. To complete this task use the code below.

```
#create variables var1 and var2  
var1 =[1,2,3,4]  
var2=True
```

ANSWER

```
#create variables var1 and var2
```

```
var1 =[1,2,3,4]
```

```
var2=True
```

```
#Print type of var1
```

```
print(type(var1))
```

```
#print type of var 2
```

```
print(type(var2))
```

TASKS

Use `print()` in combination with `len()` to print out the length of the list `var l`.

To complete this task use the code below.

```
#create variables var l
```

```
var l =[1,2,3,4]
```

ANSWER

#create a list var l

var l =[1,2,3,4]

#print out length of var l list

print(len(var l))

TYPE CONVERSION TO INTEGER

In Python, you can convert one data type to another data type , for example you can convert a boolean data type to to an integer data type. This can be done by the int() function.

```
#create a float variable
```

```
Test = True
```

```
#convert this into an integer data type by:
```

```
intvar=int(Test)
```

```
#check the data type of intvar by:
```

```
print(type(intvar))
```

Out: int

TASKS

Use `int()` to convert `var2` (a Boolean variable) to an integer data type. Store the output as `var3`. Use the following code to create a Boolean variable `var2`:

```
#create a Boolean variable var2  
var2=True
```

ANSWER

#create a float variable var2

var2=True

#convert float data type to integer data type using int () function

var3=int(var2)

#print out var3 to check data type

print(type(var3))

HELP!

Maybe you already know the name of a Python function, but you still have to figure out how to use it. Ironically, you have to ask for information about a function with another function: `help()`.

- To get help you can write `help(function name)`.
- Or you can also use “?” before the function name.

For example, on the `max()` function you can use one of these calls:

- `help(max)`
- `?max`

```
▶ help(max)
```

```
Help on built-in function max in module builtins:
```

```
max(...)
```

```
max(iterable, *[, default=obj, key=func]) -> value
```

```
max(arg1, arg2, *args, *[, key=func]) -> value
```

```
With a single iterable argument, return its biggest item. The  
default keyword-only argument specifies an object to return if  
the provided iterable is empty.
```

```
With two or more arguments, return the largest argument.
```

STRING METHODS

Strings come with a bunch of methods. If you want to discover them in more detail, you can always type `help(str)`.

- `upper()` will return the string in UPPER CASE.
- `lower()` will return the string in lower case.
- `count()` will search the sub-string in a given string & returns how many times the sub-string is present in the given string

EXAMPLE - UPPER()

Use of upper() function

Let's convert the following sentence "Python is used for Data Science" in UPPER CASE.

#to add a string:

```
String = "Python is used for Data Science"
```

#save in upperString

```
upperString = string.upper()
```

#print result

```
Print(upperString)
```

#Out:

PYTHON IS USED FOR DATA SCIENCE

EXAMPLE - COUNT()

Use of count() function

Let's count the number of times the letter "a" occurs in the following sentence "Python is used for Data Science"

#to add a string:

```
String = "Python is used for Data Science"
```

#add a sub-string

```
Substring = "a"
```

#count the number of letter "a"

```
number = string.count(substring)
```

#print result

```
Print(number)
```

#Out:

2

TASKS

Create a String place

```
place = "poolhouse"
```

Use the upper() method on place and store the result in place_up.

Print out place and place_up.

ANSWER

#Create a string place with value “poolhouse”

```
place="poolhouse"
```

#use upper() function

```
place_up =place.upper()
```

#print out place_up

```
print(place_up)
```

TASKS

Create a String place

```
place = "poolhouse"
```

Print out the number of letter o's on the variable 'place' by calling count() on 'place' and passing the letter 'o' as an input to the method.

ANSWER

#Create a string place with value “poolhouse”

```
place="poolhouse"
```

#count number of o's occur on place

```
numerofo= place.count('o')
```

print out number variable check occurrence of o's

```
print (number)
```

LIST METHODS

Strings are not the only Python types that have methods associated with them.

Lists, floats, integers and booleans are also types that come packaged with a bunch of useful methods.

In this section, you'll be experimenting with the `index()` & `count()` method within list.

- `index()`, to get the index of the first element of a list that matches its input
- `count()`, to get the number of times an element appears in a list.

EXAMPLE – LIST INDEX()

Use of index() function

Let's find the position of “cherry” in the Fruit List ['apple', 'banana', 'cherry']?

#create a list

```
fruits = ['apple', 'banana', 'cherry']
```

#find index value of “cherry”

```
position=fruits.index('cherry')
```

#print result

```
Print(position)
```

#Out:

2

EXAMPLE – LIST COUNT()

Use of count() function

Let's find the number of times “cherry” occurs in the Fruit List ['apple', 'banana', 'cherry']?

#create a list

```
Fruits=['apple','banana','cherry']
```

#count number of times “cherry” occurs in the list

```
number=fruits.count('cherry')
```

#print result

```
Print(number)
```

#Out:

|

TASKS

Use the `index()` method to get the index of the element in `areas` list that is equal to 20.0 and print out this index.

Use the list with the area of different parts of a house:

```
areas=[11.25,18.0,20.0,10.75,9.50]
```

ANSWER

```
#create an areas list
```

```
areas=[11.25,18.0,20.0,10.75,9.50]
```

```
#get index number of 20.0 in the areas list
```

```
index=areas.index(20.0)
```

```
#Print index
```

```
print (index)
```


TASKS

Use `count()` on the `areas` list to find out how many times `9.50` appears and print out this number.

Use the list with the area of different parts of a house:

```
areas=[11.25,18.0,20.0,10.75,9.50]
```

ANSWER

```
#create areas list
```

```
areas=[11.25,18.0,20.0,10.75,9.50]
```

```
#count the times 9.50 occurs in the areas list
```

```
number= areas.count(9.50)
```

```
#print out the occurrence of 9.50
```

```
print (number)
```

FURTHER LIST METHODS

Now you know how to use the `count()` & `index()` methods. Let's move on to further list methods:

- `append()`, it adds an element to a list
- `remove()`, it removes the element of a list that matches the input
- `reverse()`, it reverses the order of the elements in the list

EXAMPLE – LIST APPEND()

Use the append() function

Let's add “orange” in the Fruit List ['apple', 'banana', 'cherry']?

#create a list

```
fruits = ['apple', 'banana', 'cherry']
```

#add element in fruits list

```
appendfruits=fruits.append('orange')
```

#print result

```
Print(appendfruits)
```

#Out:

```
['apple', 'banana', 'cherry', 'orange']
```

EXAMPLE – LIST REMOVE()

Use the remove() function

Let's now remove “banana” in the Fruit List ['apple', 'banana', 'cherry', 'orange']?

#fruit list

```
fruits = ['apple', 'banana', 'cherry', 'orange']
```

#remove element in fruits list

```
removefruits=fruits.remove('banana')
```

#print result

```
Print(appendfruits)
```

#Out:

```
['apple', 'cherry', 'orange']
```

EXAMPLE – LIST REVERSE()

Use the reverse() function

Let's reverse the order of the Fruit List ['apple', 'cherry', 'orange']

#fruit list

```
fruits = ['apple', 'cherry', 'orange']
```

#reverse the order of the fruits list

```
reversefruits=fruits.reverse()
```

#print result

```
Print(reversefruits)
```

#Out:

```
['orange', 'cherry', 'apple']
```

TASKS

Add 24.5 and 15.45, respectively to areas. Make sure to add them in this order. Print out areas.

Use this list with the area of different parts of a house:

```
areas=[11.25,18.0,20.0,10.75,9.50]
```

ANSWER

```
#create areas list
```

```
areas=[11.25,18.0,20.0,10.75,9.50]
```

```
#Use append() twice to add the size of the poolhouse and garage
```

```
areas.append(24.5)
```

```
areas.append(15.45)
```

```
#print areas list
```

```
print(areas)
```


TASKS

Reverse the order of the elements in areas and print out areas.

ANSWER

```
#create areas list
```

```
areas=[11.25,18.0,20.0,10.75,9.50]
```

```
#print areas
```

```
print(areas)
```

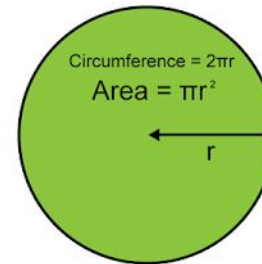
```
#reverse order of areas using reverse() function & print
```

```
areas.reverse()
```

```
print(areas)
```

USING MATH FUNCTIONS IN PYTHON

As a data scientist, some notions of geometry never hurt. Let's refresh some of the basics. You may remember some of these formulas from math class



So when you want to find the circumference, C , and area, A , of a circle when the radius of the circle is r , you can calculate C and A as:

$$C = 2\pi r$$

$$A = \pi r^2$$

IMPORT MATH PACKAGE

To calculate the area & circumference of a circle you'll require the constant pi, for that you'll need the math package.

```
#import the math package.  
import math
```

Now you can access the constant pi with math.pi.

EXAMPLE - USE OF THE MATH PACKAGE

Using the math function

So, radius of circle $r=0.43$. Let's calculate the circumference(C)

```
#import math library
```

```
import math
```

```
#define radius of the circle as r
```

```
r= 0.43
```

```
# Calculate the circumference of the circle and store it in C.
```

```
C=2*math.pi*r
```

```
#print circumference of the circle
```

```
print (C)
```

```
#Out
```

```
2.701769
```

TASKS

Calculate the area of a circle with r (radius) = 0.43 and store it in A and print the result.

$$A = \pi r^2$$

ANSWER

```
#import math library
```

```
import math
```

```
#define radius of the circle as r
```

```
r= 0.43
```

```
# Calculate the area of the circle and store it in A.
```

```
A=math.pi *r ** 2
```

```
#print Area of the circle
```

```
print (A)
```

SELECTIVE IMPORT

General imports, like `import math`, make all functionality from the `math` package available to you.

However, if you decide to only use a specific part of a package, you can always make your import more selective. So, you can individually import `pi` from the `math` package.

```
#import pi package from math  
from math import pi
```

Now you can access the constant `pi`.

EXAMPLE – SELECTIVE IMPORT FROM THE MATH PACKAGE

Using the math function

So, for radius of circle $r=2.10$. Let's calculate the circumference(C)

```
#import pi from the math package
```

```
from math import pi
```

```
#define radius of the circle as r
```

```
r= 2.10
```

```
# Calculate the circumference of the circle and store it in C.
```

```
C=2*pi*r
```

```
#print circumference of the circle
```

```
print (C)
```

```
#Out
```

```
13.1946
```

SELECTIVE IMPORT - EXAMPLE

Let's say the Moon's orbit around planet Earth is a perfect circle, with a radius $r=192500$ (in km).

You can perform a selective import from the math package where you only import the radians function.

```
#import radians function of math package  
from math import radians
```

- Let's try to calculate the distance travelled by the Moon over 13 degrees of its orbit.
- You can calculate this as $r * \text{phi}$, where r is the radius and phi is the angle in radians.
- To convert an angle in degrees to an angle in radians, we'll use the `radians()` function, which you just imported.

SELECTIVE IMPORT - EXAMPLE

Distance travelled by the Moon over 13 degrees of its orbit

```
#Definition of radius
```

```
r=192500
```

```
#import radians function of math package
```

```
from math import radians
```

```
#travel distance of Moon over 12 degree .store in dist.
```

```
dist=r*radians(13)
```

```
#Print out dist.
```

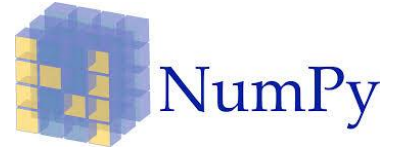
```
print(dist)
```

Out

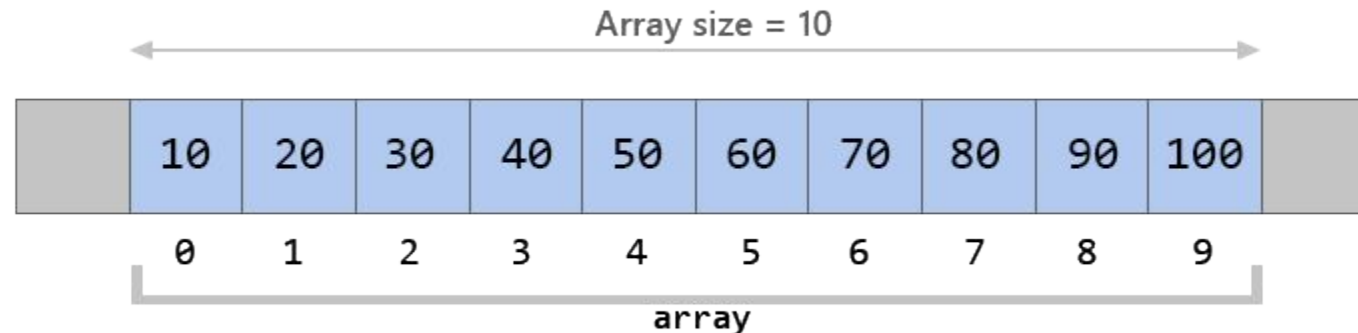
43676.8645



NUMPY



- NumPy is the core library for scientific computing in Python.
- It contains a collection of tools and techniques that can be used to solve mathematical models in Science and Engineering.
- NumPy provides vectorization of mathematical operations on arrays and matrices which are helpful features for any Data Scientist.
- A NumPy array is a grid of values, all of the same type, and is indexed.



IMPORT NUMPY PACKAGE

Let's import the numpy package as np.

```
#import the numpy package as np.
```

```
import numpy as np
```

Now you can access the numpy functions as np.

EXAMPLE - USE OF THE NUMPY PACKAGE

Using the numpy package to create a numpy array

```
#create a list
```

```
Testlist=[1,2,3,4,5,6,7,8,9]
```

```
#print out the type for TestList
```

```
print(type(Testlist))
```

```
#Out
```

```
<class 'list'>
```

```
#import numpy as np
```

```
import numpy as np
```

```
#create numpy array for Testlist as np_list
```

```
np_list=np.array(Testlist)
```

```
#print type of np_list
```

```
print(type(np_list))
```

```
#Out
```

```
<class 'numpy.ndarray'>
```

NUMPY ARRAYS & MLB

Let's dive into the world of baseball. Along the way, you'll get comfortable with the basics of numpy, a powerful package to do data science.

Let's create a Baseball list in Python script, representing the height of some baseball players in centimeters.

```
baseball=[180,215,210,210,188,176,209,200]
```



TASKS

Use `np.array()` to create a numpy array from `baseball` list. Name this array `np_baseball`. Print out the type of `np_baseball`.

Use this list to create a NumPy array

```
baseball=[180,215,210,210,188,176,209,200]
```


ANSWER

```
#create list baseball
```

```
baseball=[180,215,210,210,188,176,209,200]
```

```
#import the numpy package as np
```

```
import numpy as np
```

```
#create a numpy array from baseball as np_baseball
```

```
np_baseball=np.array (baseball)
```

```
#print out type of np_baseball
```

```
print(type(np_baseball))
```

BASEBALL PLAYERS' HEIGHT – CASE STUDY

- After making your first NumPy array, let's now dive into some more statistics on the height of the MLB players.
- MLB has passed along the height data on more than a thousand players.

Note: The height is expressed in inches.



TASKS

Create a numpy array from the list `height_in` & name this new array `np_height_in`. Print `np_height_in`.

Use this list to create a NumPy array

```
height_in=[74, 74, 72, 72, 73, 69, 69, 71, 76, 71, 73, 73, 74, 74, 69, 70, 73, 75, 78, 79, 76, 74, 76, 72, 71, 75, 77, 74, 73, 74, 78, 73, 75, 73, 75, 75, 74, 69, 71, 74, 73, 73, 76, 74, 74, 70, 72, 77, 74, 70, 73, 75, 76, 76, 78, 74, 74, 76, 77, 81, 78, 75, 77, 75, 76, 74, 72, 72, 75, 73, 73, 73, 70, 70, 70, 76, 68, 71, 72, 75, 75, 75, 75, 68, 74, 78, 71, 73, 76, 74, 74, 79, 75, 73, 76, 74, 74, 73, 72, 74, 73, 74, 72, 73, 69, 72, 73, 75, 75, 73, 72, 72, 76, 74, 72, 77, 74, 77, 75, 76, 80, 74, 74, 75, 78, 73, 73, 74, 75, 76, 71, 73, 74, 76, 76, 74, 73, 74, 70, 72, 73, 73, 73, 73, 71, 74, 74, 72, 74, 71, 74, 73, 75, 75, 79, 73, 75, 76, 74, 76, 78, 74, 76, 72, 74, 76, 74, 75, 78, 75, 72, 74, 72, 74, 70, 71, 70, 75, 71, 71, 73, 72, 71, 73, 72, 75, 74, 74, 75, 73, 77, 73, 76, 75, 74, 76, 75, 73, 71, 76, 75, 72, 71, 77, 73, 74, 71, 72, 74, 75, 73, 72, 75, 75, 74, 72, 74, 71, 70, 74, 77, 77, 75, 75, 78, 75, 76, 73, 75, 75, 79, 77, 76, 71, 75, 74, 69, 71, 76, 72, 72, 70, 72, 73, 71, 72, 71, 73, 72, 73, 74, 74, 72, 75, 74, 74, 77, 75, 73, 72, 71, 74, 77, 75, 75, 75, 78, 78, 74, 76, 78, 76, 70, 72, 80, 74, 74, 71, 70, 72, 71, 74, 71, 72, 71, 74, 69, 76, 75, 75, 76, 73, 76, 73, 77, 73, 72, 72, 77, 77, 71, 74, 74, 73, 78, 75, 73, 70, 74, 72, 73, 73, 75, 75, 74, 76, 73, 74, 75, 75, 72, 73, 73, 72, 74, 78, 76, 73, 74, 75, 70, 75, 71, 72, 78, 75, 73, 73, 71, 75, 77, 72, 69, 73, 74, 72, 70, 75, 70, 72, 72, 74, 73, 74, 76, 75, 80, 72, 75, 73, 74, 74, 73, 75, 75, 71, 73, 75, 74, 74, 72, 74, 74, 74, 73, 76, 75, 72, 73, 73, 73, 72, 72, 72, 72, 71, 75, 75, 74, 73, 75, 79, 74, 76, 73, 74, 74, 72, 74, 74, 75, 78, 74, 74, 74, 77, 70, 73, 74, 73, 71, 75, 71, 72, 77, 74, 70, 77, 73, 72, 76, 71, 76, 78, 75, 73, 78, 74, 79, 75, 76, 72, 75, 75, 70, 72, 70, 74, 71, 76, 73, 76, 71, 69, 72, 72, 69, 73, 69, 73, 74, 74, 72, 71, 72, 72, 76, 76, 76, 74, 76, 75, 71, 72, 71, 73, 75, 76, 75, 71, 75, 74, 72, 73, 73, 73, 73, 76, 72, 76, 73, 73, 73, 75, 75, 77, 73, 72, 75, 70, 74, 72, 80, 71, 71, 74, 74, 73, 75, 76, 73, 77, 72, 73, 77, 76, 71, 75, 73, 74, 77, 71, 72, 73, 69, 73, 70, 74, 76, 73, 73, 75, 73, 79, 74, 73, 74, 77, 75, 74, 73, 77, 73, 77, 74, 74, 73, 77, 74, 77, 75, 77, 75, 71, 74, 70, 79, 72, 72, 70, 74, 74, 72, 73, 72, 74, 74, 76, 82, 74, 74, 70, 73, 73, 74, 77, 72, 76, 73, 73, 72, 74, 74, 71, 72, 75, 74, 74, 77, 70, 71, 73, 76, 71, 75, 74, 72, 76, 79, 76, 73, 76, 78, 75, 76, 72, 72, 73, 73, 75, 71, 76, 70, 75, 74, 75, 73, 71, 71, 72, 73, 73, 72, 69, 73, 78, 71, 73, 75, 76, 70, 74, 77, 75, 79, 72, 77, 73, 75, 75, 75, 73, 73, 76, 77, 75, 70, 71, 71, 75, 74, 69, 70, 75, 72, 75, 73, 72, 72, 72, 76, 75, 74, 69, 73, 72, 72, 75, 77, 76, 80, 77, 76, 79, 71, 75, 73, 76, 77, 73, 76, 70, 75, 73, 75, 70, 69, 71, 72, 72, 73, 70, 70, 73, 76, 75, 72, 73, 79, 71, 72, 74, 74, 74, 72, 76, 76, 72, 72, 71, 72, 72, 70, 77, 74, 72, 76, 71, 76, 71, 73, 70, 73, 73, 72, 71, 71, 71, 72, 72, 74, 74, 74, 71, 72, 75, 72, 71, 72, 72, 72, 72, 74, 74, 77, 75, 73, 75, 73, 76, 72, 77, 75, 72, 71, 71, 75, 72, 73, 73, 71, 70, 75, 71, 76, 73, 68, 71, 72, 74, 77, 72, 76, 78, 81, 72, 73, 76, 72, 72, 74, 76, 73, 76, 75, 70, 71, 74, 72, 73, 76, 76, 73, 71, 68, 71, 71, 74, 77, 69, 72, 76, 75, 76, 75, 76, 72, 74, 76, 74, 72, 75, 78, 77, 70, 72, 79, 74, 71, 68, 77, 75, 71, 72, 70, 72, 72, 73, 72, 74, 72, 72, 75, 72, 73, 74, 72, 78, 75, 72, 74, 75, 75, 76, 74, 74, 73, 74, 71, 74, 75, 76, 74, 76, 76, 73, 75, 75, 74, 68, 72, 75, 71, 70, 72, 73, 72, 75, 74, 70, 76, 71, 82, 72, 73, 74, 71, 75, 77, 72, 74, 72, 73, 78, 77, 73, 73, 73, 73, 73, 76, 75, 70, 73, 72, 73, 75, 74, 73, 73, 76, 73, 75, 70, 77, 72, 77, 74, 75, 75, 75, 75, 72, 74, 71, 76, 71, 75, 76, 83, 75, 74, 76, 72, 72, 75, 75, 72, 77, 73, 72, 70, 74, 72, 74, 72, 71, 70, 71, 76, 74, 76, 74, 74, 74, 75, 75, 71, 71, 74, 77, 71, 74, 75, 77, 76, 74, 76, 72, 71, 72, 75, 73, 68, 72, 69, 73, 73, 75, 70, 70, 74, 75, 74, 74, 73, 74, 75, 77, 73, 74, 76, 74, 75, 73, 76, 78, 75, 73, 77, 74, 72, 74, 72, 71, 73, 75, 73, 67, 67, 76, 74, 73, 70, 75, 70, 72, 77, 79, 78]
```

ANSWER

creating a regular list height_in

```
height_in=[74,74,72,72,73,69,69,71,76,71,73,73,74,74,69,70,73,75,78,79,76,74,76,72,71,75,77,74,73,74,78,73,75,73,75,75,74,69,71,74,73,73,76,74,74,70,72,77,74,70,73,75,76,76,78,74,74,76,77,81,78,75,77,75,76,74,72,72,75,73,73,73,70,70,70,76,68,71,72,75,75,75,75,68,74,78,71,73,76,74,74,79,75,73,76,74,74,73,72,74,73,74,72,73,69,72,73,75,75,73,72,72,76,74,72,77,74,77,75,76,80,74,74,75,78,73,73,74,75,76,71,73,74,76,76,74,73,74,70,72,73,73,73,71,74,74,72,74,71,74,73,75,75,79,73,75,76,74,76,78,74,76,72,74,76,74,75,78,75,72,74,72,74,70,71,70,75,71,71,73,72,71,73,72,75,74,74,75,73,77,73,76,75,74,76,75,73,71,76,75,72,71,77,73,74,71,72,74,75,73,72,75,74,72,74,71,70,74,77,77,75,75,78,75,76,73,75,75,79,77,76,71,75,74,69,71,76,72,72,70,72,73,71,72,71,73,72,73,74,74,72,75,74,74,77,75,73,72,71,74,77,75,75,75,78,78,74,76,78,76,70,72,80,74,74,71,70,72,71,74,71,72,71,74,69,76,75,75,76,73,76,73,77,73,72,72,77,77,71,74,74,73,78,75,73,70,74,72,73,73,75,75,74,76,73,74,75,75,72,73,72,74,78,76,73,74,75,70,75,71,72,78,75,73,73,71,75,77,72,69,73,74,72,70,75,70,72,72,74,73,74,76,75,80,72,75,73,74,74,73,75,75,71,73,75,74,74,72,74,74,74,73,76,75,72,73,73,72,72,72,71,75,75,74,73,75,79,74,76,73,74,74,72,74,74,75,78,74,74,77,70,73,74,73,71,75,71,72,77,74,70,77,73,72,76,71,76,78,75,73,78,74,79,75,76,72,75,75,70,72,70,74,71,76,73,76,71,69,72,72,69,73,69,73,74,74,72,71,72,72,76,76,74,76,75,71,72,71,73,75,76,75,71,75,74,72,73,73,73,76,72,76,73,73,73,75,75,77,73,72,75,70,74,72,80,71,71,74,74,73,75,76,73,77,72,73,77,76,71,75,73,74,77,71,72,73,69,73,70,74,76,73,73,75,73,79,74,73,74,77,75,74,73,77,73,77,74,74,73,77,74,77,75,77,75,71,74,70,79,72,72,70,74,74,72,73,72,74,74,76,82,74,74,70,73,73,74,77,72,76,73,73,72,74,74,71,72,75,74,74,77,70,71,73,76,71,75,74,72,76,79,76,73,76,78,75,76,72,72,73,73,75,71,76,70,75,74,75,73,71,71,72,73,73,72,69,73,78,71,73,75,76,70,74,77,75,79,72,77,73,75,75,73,73,76,77,75,70,71,71,75,74,69,70,75,72,75,73,72,72,72,76,75,74,69,73,72,72,75,77,76,80,77,76,79,71,75,73,76,77,73,76,70,75,73,75,70,69,71,72,72,73,70,70,73,76,75,72,73,79,71,72,74,74,74,72,76,76,72,72,71,72,72,70,77,74,72,76,71,76,71,73,70,73,73,72,71,71,71,72,72,74,74,74,71,72,75,72,71,72,72,72,74,74,77,75,73,75,73,76,72,77,75,72,71,71,75,72,73,73,71,70,75,71,76,73,68,71,72,74,77,72,76,78,81,72,73,76,72,72,74,76,73,76,75,70,71,74,72,73,76,76,73,71,68,71,71,74,77,69,72,76,75,76,75,76,72,74,76,74,72,75,78,77,70,72,79,74,71,68,77,75,71,72,70,72,72,73,72,74,72,72,75,72,73,74,72,78,75,72,74,75,75,76,74,74,73,74,71,74,75,76,74,76,76,73,75,75,74,68,72,75,71,70,72,73,72,75,74,70,76,71,82,72,73,74,71,75,77,72,74,72,73,78,77,73,73,73,73,73,76,75,70,73,72,73,75,74,73,73,76,73,75,70,77,72,77,74,75,75,75,72,74,71,76,71,75,76,83,75,74,76,72,72,75,75,72,77,73,72,70,74,72,74,72,71,70,71,76,74,76,74,74,74,75,75,71,71,74,77,71,74,75,77,76,74,76,72,71,72,75,73,68,72,69,73,73,75,70,70,74,75,74,74,73,74,75,77,73,74,76,74,75,73,76,78,75,73,77,74,72,74,72,71,73,75,73,67,67,76,74,73,70,75,70,72,77,79,78]
```

#import numpy as np

import numpy as np

#create a numpy array from height_in as np_height_in

np_height_in=np.array(height_in)

#print out np_height_in

print(np_height_in)

TASKS

Convert all height measurements from inches to meters by multiplying `np_height_in` (created in the last task) with 0.0254 & store the new values in a new array, `np_height_m`.

Print out `np_height_m`.

ANSWER

```
#import numpy as np
```

```
import numpy as np
```

```
#convert height from inches to meters by multiply all array 0.0254
```

```
np_height_m=np_height_in*0.0245
```

```
#print out np_height_m
```

```
print(np_height_m)
```

MLB PLAYERS' WEIGHT

- MLB has also passed along the weight data for the same players, the weight is expressed in pounds.



TASKS

Create a numpy array from the `weight_lb` list with the correct units. Multiply by 0.453592 to go from pounds to kilograms. Store the resulting numpy array as `np_weight_kg`.

Use this list to create a NumPy array

```
weight_lb=[180, 215, 210, 210, 188, 176, 209, 200, 231, 180, 188, 180, 185, 160, 180, 185, 189, 185, 219, 230, 205, 230, 195, 180, 192, 225, 203, 195, 182, 188, 200, 180, 200, 200, 245, 240, 215, 185, 175, 199, 200, 215, 200, 205, 206, 186, 188, 220, 210, 195, 200, 200, 212, 224, 210, 205, 220, 195, 200, 260, 228, 270, 200, 210, 190, 220, 180, 205, 210, 220, 211, 200, 180, 190, 170, 230, 155, 185, 185, 200, 225, 225, 220, 160, 205, 235, 250, 210, 190, 160, 200, 205, 222, 195, 205, 220, 220, 170, 185, 195, 220, 230, 180, 220, 180, 180, 170, 210, 215, 200, 213, 180, 192, 235, 185, 235, 210, 222, 210, 230, 220, 180, 190, 200, 210, 194, 180, 190, 240, 200, 198, 200, 195, 210, 220, 190, 210, 225, 180, 185, 170, 185, 185, 180, 178, 175, 200, 204, 211, 190, 210, 190, 190, 185, 290, 175, 185, 200, 220, 170, 220, 190, 220, 205, 200, 250, 225, 215, 210, 215, 195, 200, 194, 220, 180, 180, 170, 195, 180, 170, 206, 205, 200, 225, 201, 225, 233, 180, 225, 180, 220, 180, 237, 215, 190, 235, 190, 180, 165, 195, 200, 190, 190, 185, 185, 205, 190, 205, 206, 220, 208, 170, 195, 210, 190, 211, 230, 170, 185, 185, 241, 225, 210, 175, 230, 200, 215, 198, 226, 278, 215, 230, 240, 184, 219, 170, 218, 190, 225, 220, 176, 190, 197, 204, 167, 180, 195, 220, 215, 185, 190, 205, 205, 200, 210, 215, 200, 205, 211, 190, 208, 200, 210, 232, 230, 210, 220, 210, 202, 212, 225, 170, 190, 200, 237, 220, 170, 193, 190, 150, 220, 200, 190, 185, 185, 200, 172, 220, 225, 190, 195, 219, 190, 197, 200, 195, 210, 177, 220, 235, 180, 195, 195, 190, 230, 190, 200, 190, 190, 200, 200, 184, 200, 180, 219, 187, 200, 220, 205, 190, 170, 160, 215, 175, 205, 200, 214, 200, 190, 180, 205, 220, 190, 215, 235, 191, 200, 181, 200, 210, 240, 185, 165, 190, 185, 175, 155, 210, 170, 175, 220, 210, 205, 200, 205, 195, 240, 150, 200, 215, 202, 200, 190, 205, 190, 160, 215, 185, 200, 190, 210, 185, 220, 190, 202, 205, 220, 175, 160, 190, 200, 229, 206, 220, 180, 195, 175, 188, 230, 190, 200, 190, 219, 235, 180, 180, 180, 200, 234, 185, 220, 223, 200, 210, 200, 210, 190, 177, 227, 180, 195, 199, 175, 185, 240, 210, 180, 194, 225, 180, 205, 193, 230, 230, 220, 200, 249, 190, 208, 245, 250, 160, 192, 220, 170, 197, 155, 190, 200, 220, 210, 228, 190, 160, 184, 180, 180, 200, 176, 160, 222, 211, 195, 200, 175, 206, 240, 185, 260, 185, 221, 205, 200, 170, 201, 205, 185, 205, 245, 220, 210, 220, 185, 175, 170, 180, 200, 210, 175, 220, 206, 180, 210, 195, 200, 200, 164, 180, 220, 195, 205, 170, 240, 210, 195, 200, 205, 192, 190, 170, 240, 200, 205, 175, 250, 220, 224, 210, 195, 180, 245, 175, 180, 215, 175, 180, 195, 230, 230, 205, 215, 195, 180, 205, 180, 190, 180, 190, 190, 220, 210, 255, 190, 230, 200, 205, 210, 225, 215, 220, 205, 200, 220, 197, 225, 187, 245, 185, 185, 175, 200, 180, 188, 225, 200, 210, 245, 213, 231, 165, 228, 210, 250, 191, 190, 200, 215, 254, 232, 180, 215, 220, 180, 200, 170, 195, 210, 200, 220, 165, 180, 200, 200, 170, 224, 220, 180, 198, 240, 239, 185, 210, 220, 200, 195, 220, 230, 170, 220, 230, 165, 205, 192, 210, 205, 200, 210, 185, 195, 202, 205, 195, 180, 200, 185, 240, 185, 220, 205, 205, 180, 201, 190, 208, 240, 180, 230, 195, 215, 190, 195, 215, 215, 220, 220, 230, 195, 190, 195, 209, 204, 170, 185, 205, 175, 210, 190, 180, 180, 160, 235, 200, 210, 180, 190, 197, 203, 205, 170, 200, 250, 200, 220, 200, 190, 170, 190, 220, 215, 206, 215, 185, 235, 188, 230, 195, 168, 190, 160, 200, 200, 189, 180, 190, 200, 220, 187, 240, 190, 180, 185, 210, 220, 219, 190, 193, 175, 180, 215, 210, 200, 190, 185, 220, 170, 195, 205, 195, 210, 190, 190, 180, 220, 190, 186, 185, 190, 180, 190, 170, 210, 240, 220, 180, 210, 210, 195, 160, 180, 205, 200, 185, 245, 190, 210, 200, 200, 222, 215, 240, 170, 220, 156, 190, 202, 221, 200, 190, 210, 190, 200, 165, 190, 185, 230, 208, 209, 175, 180, 200, 205, 200, 250, 210, 230, 244, 202, 240, 200, 215, 177, 210, 170, 215, 217, 198, 200, 220, 170, 200, 230, 231, 183, 192, 167, 190, 180, 180, 215, 160, 205, 223, 175, 170, 190, 240, 175, 230, 223, 196, 167, 195, 190, 250, 190, 190, 190, 170, 160, 150, 225, 220, 209, 210, 176, 260, 195, 190, 184, 180, 195, 195, 219, 225, 212, 202, 185, 200, 209, 200, 195, 228, 210, 190, 212, 190, 218, 220, 190, 235, 210, 200, 188, 210, 235, 188, 215, 216, 220, 180, 185, 200, 210, 220, 185, 231, 210, 195, 200, 205, 200, 190, 250, 185, 180, 170, 180, 208, 235, 215, 244, 220, 185, 230, 190, 200, 180, 190, 196, 180, 230, 224, 160, 178, 205, 185, 210, 180, 190, 200, 257, 190, 220, 165, 205, 200, 208, 185, 215, 170, 235, 210, 170, 180, 170, 190, 150, 230, 203, 260, 246, 186, 210, 198, 210, 215, 180, 200, 245, 200, 192, 192, 200, 192, 205, 190, 186, 170, 197, 219, 200, 220, 207, 225, 207, 212, 225, 170, 190, 210, 230, 210, 200, 238, 234, 222, 200, 190, 170, 220, 223, 210, 215, 196, 175, 175, 189, 205, 210, 180, 180, 197, 220, 228, 190, 204, 165, 216, 220, 208, 210, 215, 195, 200, 215, 229, 240, 207, 205, 208, 185, 190, 170, 208, 225, 190, 225, 185, 180, 165, 240, 220, 212, 163, 215, 175, 205, 210, 205, 208]
```


ANSWER

```
# regular list weight_lb
```

```
weight_lb=[180,215,210,210,188,176,209,200,231,180,188,180,185,160,180,185,189,185,219,230,205,230,195,180,192,225,203,195,182,188,200,180,200,200,245,240,215,185,175,199,200,215,200,205,206,186,188,220,210,195,200,200,212,224,210,205,220,195,200,260,228,270,200,210,190,220,180,205,210,220,211,200,180,190,170,230,155,185,185,200,225,225,220,160,205,235,250,210,190,160,200,205,222,195,205,220,220,170,185,195,220,230,180,220,180,180,170,210,215,200,213,180,192,235,185,235,210,222,210,230,220,180,190,200,210,194,180,190,240,200,198,200,195,210,220,190,210,225,180,185,170,185,185,180,178,175,200,204,211,190,210,190,190,185,290,175,185,200,220,170,220,190,220,205,200,250,225,215,210,215,195,200,194,220,180,180,170,195,180,170,206,205,200,225,201,225,233,180,225,180,220,180,237,215,190,235,190,180,165,195,200,190,190,185,185,205,190,205,206,220,208,170,195,210,190,211,230,170,185,185,241,225,210,175,230,200,215,198,226,278,215,230,240,184,219,170,218,190,225,220,176,190,197,204,167,180,195,220,215,185,190,205,205,200,210,215,200,205,211,190,208,200,210,232,230,210,220,210,202,212,225,170,190,200,237,220,170,193,190,150,220,200,190,185,185,200,172,220,225,190,195,219,190,197,200,195,210,177,220,235,180,195,195,190,230,190,200,190,190,200,200,184,200,180,219,187,200,220,205,190,170,160,215,175,205,200,214,200,190,180,205,220,190,215,235,191,200,181,200,210,240,185,165,190,185,175,155,210,170,175,220,210,205,200,205,195,240,150,200,215,202,200,190,205,190,160,215,185,200,190,210,185,220,190,202,205,220,175,160,190,200,229,206,220,180,195,175,188,230,190,200,190,219,235,180,180,180,200,234,185,220,223,200,210,200,210,190,177,227,180,195,199,175,185,240,210,180,194,225,180,205,193,230,230,220,200,249,190,208,245,250,160,192,220,170,197,155,190,200,220,210,228,190,160,184,180,180,200,176,160,222,211,195,200,175,206,240,185,260,185,221,205,200,170,201,205,185,205,245,220,210,220,185,175,170,180,200,210,175,220,206,180,210,195,200,200,164,180,220,195,205,170,240,210,195,200,205,192,190,170,240,200,205,175,250,220,224,210,195,180,245,175,180,215,175,180,195,230,230,205,215,195,180,205,180,190,180,190,190,220,210,255,190,230,200,205,210,225,215,220,205,200,220,197,225,187,245,185,185,175,200,180,188,225,200,210,245,213,231,165,228,210,250,191,190,200,215,254,232,180,215,220,180,200,170,195,210,200,220,165,180,200,200,170,224,220,180,198,240,239,185,210,220,200,195,220,230,170,220,230,165,205,192,210,205,200,210,185,195,202,205,195,180,200,185,240,185,220,205,205,180,201,190,208,240,180,230,195,215,190,195,215,215,220,220,230,195,190,195,209,204,170,185,205,175,210,190,180,180,160,235,200,210,180,190,197,203,205,170,200,250,200,220,200,190,170,190,220,215,206,215,185,235,188,230,195,168,190,160,200,200,189,180,190,200,220,187,240,190,180,185,210,220,219,190,193,175,180,215,210,200,190,185,220,170,195,205,195,210,190,190,180,220,190,186,185,190,180,190,170,210,240,220,180,210,210,195,160,180,205,200,185,245,190,210,200,200,222,215,240,170,220,156,190,202,221,200,190,210,190,200,165,190,185,230,208,209,175,180,200,205,200,250,210,230,244,202,240,200,215,177,210,170,215,217,198,200,220,170,200,230,231,183,192,167,190,180,180,215,160,205,223,175,170,190,240,175,230,223,196,167,195,190,250,190,190,190,170,160,150,225,220,209,210,176,260,195,190,184,180,195,195,219,225,212,202,185,200,209,200,195,228,210,190,212,190,218,220,190,235,210,200,188,210,235,188,215,216,220,180,185,200,210,220,185,231,210,195,200,205,200,190,250,185,180,170,180,208,235,215,244,220,185,230,190,200,180,190,196,180,230,224,160,178,205,185,210,180,190,200,257,190,220,165,205,200,208,185,215,170,235,210,170,180,170,190,150,230,203,260,246,186,210,198,210,215,180,200,245,200,192,192,200,192,205,190,186,170,197,219,200,220,207,225,207,212,225,170,190,210,230,210,200,238,234,222,200,190,170,220,223,210,215,196,175,175,189,205,210,180,180,197,220,228,190,204,165,216,220,208,210,215,195,200,215,229,240,207,205,208,185,190,170,208,225,190,225,185,180,165,240,220,212,163,215,175,205,210,205,208]
```

```
#import numpy as np
```

```
import numpy as np
```

```
#create a numpy array from weight_lb as np_weight_lb
```

```
np_weight_lb=np.array(weight_lb)
```

```
#print out np_weight_lb
```

```
print(np_weight_lb)
```

```
#convert weight from pounds to kilograms by multiply all array 0.453592
```

```
np_weight_kg= np_weight_lb* 0.453592
```

```
#print out np_weight_kg
```

```
print(np_weight_kg)
```

MLB PLAYERS' BMI

- Now we can use our NumPy arrays `np_height_m` and `np_weight_kg` to calculate the players' BMI.
- BMI or Body Mass Index is one way of measuring whether a person's weight is healthy. It takes into account both their weight & height.

$$\text{BMI} = \frac{\text{weight(kg)}}{\text{height(m)}^2}$$



TASKS

Use `np_height_m` and `np_weight_kg` to calculate the BMI of each player. Use the following equation:

$$\text{BMI} = \frac{\text{weight(kg)}}{\text{height(m)}^2}$$

Save the resulting numpy array as `bmi` & print out `bmi`.

Note: The height should be squared as per the formula of BMI

ANSWER

```
#import numpy as np
import numpy as np
#Calculate the BMI as bmi
bmi=np_weight_kg / np_height_m ** 2
#print out bmi
print (bmi)
```

NUMPY: BOOLEAN INDEXING

- A boolean array is a NumPy array with boolean values such as `[False True True False False]`
- Such an array can be obtained by applying a logical operator to another NumPy array.
- NumPy will automatically create a boolean array when comparisons are made.

NUMPY: BOOLEAN INDEXING - EXAMPLE

Using logical operators on a list to check which values in a list are greater than 5

```
#create a list & import NumPy
```

```
x = [4 , 9 , 6, 3, 1]
```

```
import numpy as np
```

```
#create a NumPy array
```

```
y = np.array(x)
```

```
#using logical operator to create a Boolean array
```

```
high = y > 5
```

```
#print high
```

```
Print(high)
```

```
Out:
```

```
[False True True False False]
```

```
#returning values which are greater the 5
```

```
print(y[high])
```

```
Out:
```

```
[9,6]
```

TASKS

Use the BMI calculation to create a boolean numpy array: the element of the array should be True if the corresponding baseball player's BMI is below 21. You can use the `<` operator for this. Name the array `'np_light.'`

Print the array light.

```
# Import numpy
import numpy as np

# Calculate the BMI: bmi
np_height_m = np.array(height_in) * 0.0254
np_weight_kg = np.array(weight_lb) * 0.453592
bmi = np_weight_kg / np_height_m ** 2
```

ANSWER

```
#import numpy as np
import numpy as np
#Calculate the BMI as bmi
bmi=np_weight_kg / np_height_m ** 2
#print out bmi
print (bmi)
#Create the light array
lightArray =bmi<21
#print out lightArray
print(lightArray)
```


TASKS

Create a boolean numpy array: the element of the array should be True if the corresponding baseball player's BMI is greater than 21. You can use the `>` operator for this. Name the array `np_heavy`.

Print `np_heavy`

ANSWER

```
#import numpy as np
```

```
import numpy as np
```

```
#Calculate the BMI as bmi
```

```
bmi=np_weight_kg / np_height_m ** 2
```

```
#print out bmi
```

```
print (bmi)
```

```
#Create the light array
```

```
heavyArray =bmi >21
```

```
#print out heavyArray
```

```
print(heavyArray)
```

SUBSETTING NUMPY ARRAYS

- Subsetting is a useful feature for NumPy arrays.
- Subsetting can be used to acquire a single element from an array by using the index number in square bracket notation `[]`.
- Subsetting can also be helpful to acquire a range of element form an array.
- To get a range of element form an array simply use the start and end index numbers in bracket notation `[start:end]`.

SUBSETTING NUMPY ARRAYS – EXAMPLE SINGLE ELEMENT

Acquiring a single element from an array

```
#create an array
```

```
x = ["a", "b", "c"]
```

```
#subsetting to acquire a single element
```

```
x[1]
```

```
#Out
```

```
b
```

SUBSETTING NUMPY ARRAYS – EXAMPLE RANGE OF ELEMENTS

Acquiring a range of elements from an array

Example:

```
#create an array
```

```
x = ["a", "b", "c", "d", "e", "f", "g", "h", "i"]
```

```
#create a NumPy array from x
```

```
np_x = np.array(x)
```

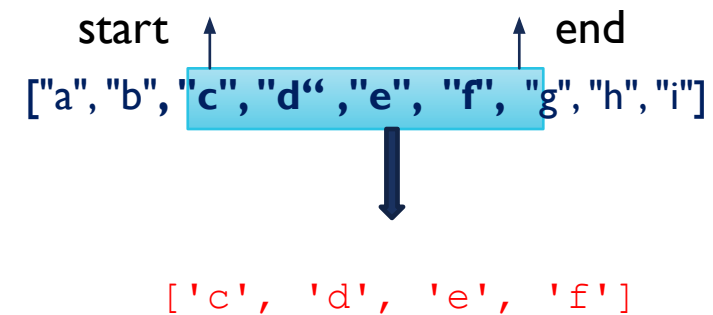
```
#subsetting to acquire the range of elements from 2 to 6
```

```
print(np_x[2:6])
```

Out:

```
['c', 'd', 'e', 'f']
```

Note: The output array has four elements start from the 2nd & ending on the 5th



TASKS

Using the MLB data subset and print out the element at index 50 from `np_weight_lb`.

Use the code written below.

```
#import numpy as np
```

```
import numpy as np
```

```
#store weight as numpy array
```

```
np_weight_lb=np.array(weight_lb)
```

ANSWER

```
#import numpy as np
import numpy as np
#store weight lists as numpy array
np_weight_lb = np.array(weight_lb)
#printing out the element at index 50
print(np_weight_lb[50])
```

TASKS

Print out a sub-array of `np_height_in` that contains the elements at index 100 up to and including 110.

Use the code written below.

```
#import numpy as np
import numpy as np
#store height as numpy array
np_height_in=np.array(height_in)
```


ANSWER

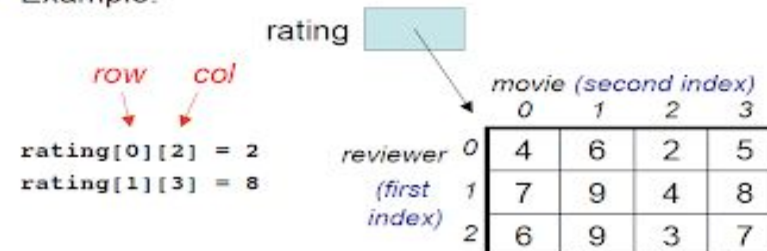
```
#import numpy as np
import numpy as np
#store height lists as numpy array
np_height_in=np.array(height_in)
#Print out a sub-array of np_height_in : index 100 up to and including index 110.
print(np_height_in[100:111])
```

2D NUMPY ARRAY

- A two dimensional array is an array within an array, it is an array of arrays.
- In this type of array the position of an data element is referred by two indices instead of one.
- So it represents a table with rows and columns of data

Two-Dimensional Arrays

- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.
- Example:



row col

rating

reviewer (first index)

movie (second index)

	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

`rating[0][2] = 2`
`rating[1][3] = 8`

2D NUMPY ARRAY - EXAMPLE

Before working on the actual MLB data, let's try to create a 2D numpy array from a small list of lists.

The list contains 4 elements. Each of these elements is a list containing the height and the weight of 4 baseball players, in this order. [height, weight]

```
#creating a baseball list of lists
```

```
baseball= [[180, 78.4],  
           [215, 102.7],  
           [210, 98.5],  
           [188, 75.2]]
```

```
#import numpy
```

```
import numpy as np
```

```
#create 2D numpy array from baseball to np_baseball
```

```
np_baseball=np.array(baseball)
```

```
#print out the shape of np_baseball
```

```
print(np_baseball.shape)
```

Out:

(4,2)

The Shape function will tell you about the dimensions of a Numpy array. it will give you how many rows and column does an array have

BASEBALL DATA IN 2D FORM (TASK BACKGROUND)

- Let's restructure the MLB data in a 2D numpy array so it can make more sense.
- This array should have 1015 rows, corresponding to the 1015 baseball players you have information on, and 2 columns (for height and weight).
- In this list of lists, each sublist represents the height and weight of a single baseball player.

TASKS

Use `np.array()` to create a 2D numpy array from `baseball` list. Name it `np_baseball`.

Print out the shape attribute of `np_baseball`. Use `np_baseball.shape`.

baseball																																																																																																			
195]	[73, 182]	[74, 188]	[78, 200]	[73, 180]	[75, 200]	[73, 200]	[75, 245]	[75, 240]	[74, 215]	[69, 185]	[71, 175]	[74, 199]	[73, 200]	[73, 215]	[76, 200]	[74, 205]	[74, 206]	[70, 186]	[72, 188]	[77, 220]	[74, 210]	[70, 195]	[73, 200]	[75, 200]	[76, 212]	[76, 224]	[78, 210]	[74, 205]	[74, 220]	[76,																																																																					
195]	[77, 200]	[81, 260]	[78, 228]	[75, 270]	[77, 200]	[75, 210]	[76, 190]	[74, 220]	[72, 180]	[72, 205]	[75, 210]	[73, 220]	[73, 211]	[73, 200]	[70, 180]	[70, 190]	[70, 170]	[76, 230]	[68, 155]	[71, 185]	[72, 185]	[75, 200]	[75, 225]	[75, 225]	[75, 220]	[68, 160]	[74, 205]	[78, 235]	[71, 250]	[73,																																																																					
210]	[76, 190]	[74, 160]	[74, 200]	[79, 205]	[75, 222]	[73, 195]	[76, 205]	[74, 220]	[74, 220]	[73, 170]	[72, 185]	[74, 195]	[73, 220]	[74, 230]	[72, 180]	[73, 190]	[69, 180]	[72, 180]	[73, 170]	[75, 210]	[75, 215]	[73, 200]	[72, 213]	[72, 180]	[76, 192]	[74, 235]	[72, 185]	[77, 235]	[74, 210]	[77,																																																																					
222]	[75, 210]	[76, 230]	[80, 220]	[74, 180]	[74, 190]	[75, 200]	[78, 210]	[73, 194]	[73, 180]	[74, 190]	[75, 240]	[76, 200]	[71, 198]	[73, 200]	[74, 195]	[76, 210]	[76, 220]	[74, 190]	[73, 210]	[74, 225]	[70, 180]	[72, 185]	[73, 170]	[73, 185]	[73, 185]	[73, 180]	[71, 178]	[74, 175]	[74, 200]	[72,																																																																					
204]	[74, 211]	[71, 190]	[74, 210]	[73, 190]	[75, 190]	[75, 185]	[79, 290]	[73, 175]	[75, 185]	[76, 200]	[74, 220]	[76, 170]	[78, 220]	[74, 190]	[76, 220]	[72, 205]	[74, 200]	[76, 250]	[74, 225]	[75, 215]	[78, 210]	[72, 195]	[72, 195]	[74, 200]	[72, 194]	[74, 220]	[70, 180]	[71, 180]	[70, 170]	[75,																																																																					
195]	[71, 180]	[71, 170]	[73, 206]	[72, 205]	[71, 200]	[73, 225]	[72, 201]	[75, 225]	[74, 233]	[74, 180]	[75, 225]	[73, 180]	[77, 220]	[73, 180]	[76, 237]	[75, 215]	[74, 190]	[76, 235]	[75, 190]	[73, 180]	[71, 165]	[76, 195]	[75, 200]	[72, 190]	[71, 190]	[77, 185]	[73, 185]	[74, 205]	[71, 190]	[72,																																																																					
205]	[74, 206]	[75, 220]	[73, 208]	[72, 170]	[75, 195]	[75, 210]	[74, 190]	[72, 211]	[74, 230]	[71, 170]	[70, 185]	[74, 185]	[77, 241]	[77, 225]	[75, 210]	[75, 175]	[78, 230]	[75, 200]	[76, 215]	[73, 198]	[75, 226]	[75, 278]	[79, 215]	[77, 230]	[76, 240]	[71, 184]	[75, 219]	[74, 170]	[69, 181]	[71,																																																																					
190]	[76, 225]	[72, 220]	[72, 176]	[70, 190]	[72, 197]	[73, 204]	[71, 167]	[72, 180]	[71, 195]	[73, 220]	[72, 215]	[73, 185]	[74, 190]	[74, 205]	[72, 205]	[75, 200]	[74, 210]	[74, 215]	[77, 200]	[75, 205]	[73, 211]	[72, 190]	[71, 208]	[74, 200]	[77, 210]	[75, 232]	[75, 230]	[75, 210]	[78, 220]	[78,																																																																					
210]	[74, 202]	[76, 212]	[78, 225]	[76, 170]	[70, 190]	[72, 200]	[80, 237]	[74, 220]	[74, 170]	[71, 193]	[70, 190]	[72, 150]	[71, 220]	[74, 200]	[71, 190]	[72, 185]	[74, 185]	[74, 200]	[69, 172]	[76, 220]	[75, 225]	[75, 190]	[76, 195]	[73, 219]	[76, 190]	[73, 197]	[77, 200]	[73, 195]	[72, 210]	[72,																																																																					
177]	[77, 220]	[77, 235]	[71, 180]	[74, 195]	[74, 195]	[73, 190]	[78, 230]	[75, 190]	[73, 200]	[70, 190]	[74, 190]	[72, 200]	[73, 200]	[73, 184]	[75, 200]	[75, 180]	[74, 219]	[76, 187]	[73, 200]	[74, 220]	[75, 205]	[75, 190]	[72, 170]	[73, 160]	[73, 215]	[72, 175]	[74, 205]	[78, 200]	[76, 214]	[73,																																																																					
200]	[74, 190]	[75, 180]	[70, 205]	[75, 220]	[71, 190]	[72, 215]	[78, 235]	[75, 191]	[73, 200]	[73, 181]	[71, 200]	[75, 210]	[77, 240]	[72, 185]	[69, 165]	[73, 190]	[74, 185]	[72, 175]	[70, 155]	[75, 210]	[70, 170]	[72, 175]	[72, 220]	[74, 210]	[73, 205]	[74, 200]	[76, 205]	[75, 195]	[80, 240]	[72,																																																																					
150]	[75, 200]	[73, 215]	[74, 2																																																																																																

ANSWER

#height and weight data as baseball list of lists

```
baseball = [[74, 180], [74, 215], [72, 210], [72, 210], [73, 188], [69, 176], [69, 209], [71, 200], [76, 231], [71, 180], [73, 188], [73, 180], [74, 185], [74, 160], [69, 180], [70, 185], [73, 189], [75, 185], [78, 219], [79, 230], [76, 205], [74, 230], [76, 195], [72, 180], [71, 192], [75, 225], [77, 203], [74, 195], [73, 182], [74, 188], [78, 200], [73, 180], [75, 200], [73, 200], [75, 245], [75, 240], [74, 215], [69, 185], [71, 175], [74, 199], [73, 200], [73, 215], [76, 200], [74, 205], [74, 206], [70, 186], [72, 188], [77, 220], [74, 210], [70, 195], [73, 200], [75, 200], [76, 212], [76, 224], [78, 210], [74, 205], [74, 220], [76, 195], [77, 200], [81, 260], [78, 228], [75, 270], [77, 200], [75, 210], [76, 190], [74, 220], [72, 180], [72, 205], [75, 210], [73, 220], [73, 211], [73, 200], [70, 180], [70, 190], [70, 170], [76, 230], [68, 155], [71, 185], [72, 185], [75, 200], [75, 225], [75, 225], [75, 220], [68, 160], [74, 205], [78, 235], [71, 250], [73, 210], [76, 190], [74, 160], [74, 200], [79, 205], [75, 222], [73, 195], [76, 205], [74, 220], [73, 170], [72, 185], [74, 195], [73, 220], [74, 230], [72, 180], [73, 220], [69, 180], [72, 180], [73, 170], [75, 210], [75, 215], [73, 200], [72, 213], [72, 180], [76, 192], [74, 235], [72, 185], [77, 235], [74, 210], [77, 222], [75, 210], [76, 230], [80, 220], [74, 180], [74, 190], [75, 200], [78, 210], [73, 194], [73, 180], [74, 190], [75, 240], [76, 200], [71, 198], [73, 200], [74, 195], [76, 210], [76, 220], [74, 190], [73, 210], [74, 225], [70, 180], [72, 185], [73, 170], [73, 185], [73, 185], [73, 180], [71, 178], [74, 175], [74, 200], [72, 204], [74, 211], [71, 190], [74, 210], [73, 190], [75, 190], [75, 185], [79, 290], [73, 175], [75, 185], [76, 200], [74, 220], [76, 170], [78, 220], [74, 190], [76, 220], [72, 205], [74, 200], [76, 250], [74, 225], [75, 215], [78, 210], [75, 215], [72, 195], [74, 200], [72, 194], [74, 220], [70, 180], [71, 180], [70, 170], [75, 195], [71, 180], [71, 170], [73, 206], [72, 205], [71, 200], [73, 225], [72, 201], [75, 225], [74, 233], [74, 180], [75, 225], [73, 180], [77, 220], [73, 180], [76, 237], [75, 215], [74, 190], [76, 235], [75, 190], [73, 180], [71, 165], [76, 195], [75, 200], [72, 190], [71, 190], [77, 185], [73, 185], [74, 205], [71, 190], [72, 205], [74, 206], [75, 220], [73, 208], [72, 170], [75, 195], [75, 210], [74, 190], [72, 211], [74, 230], [71, 170], [70, 185], [74, 185], [77, 241], [77, 225], [75, 210], [75, 175], [78, 230], [75, 200], [76, 215], [73, 198], [75, 226], [75, 278], [79, 215], [77, 230], [76, 240], [71, 184], [75, 219], [74, 170], [69, 218], [71, 190], [76, 225], [72, 220], [72, 176], [70, 190], [72, 197], [73, 204], [71, 167], [72, 180], [71, 195], [73, 220], [72, 215], [73, 185], [74, 190], [74, 205], [72, 205], [75, 200], [74, 210], [74, 215], [77, 200], [75, 205], [73, 211], [72, 190], [71, 208], [74, 200], [77, 210], [75, 232], [75, 230], [75, 210], [78, 220], [78, 210], [74, 202], [76, 212], [78, 225], [76, 170], [70, 190], [72, 200], [80, 237], [74, 220], [74, 170], [71, 193], [70, 190], [72, 150], [71, 220], [74, 200], [71, 190], [72, 185], [71, 185], [74, 200], [69, 172], [76, 220], [75, 225], [75, 190], [76, 195], [73, 219], [76, 190], [73, 197], [77, 200], [73, 195], [72, 210], [72, 177], [77, 220], [77, 235], [71, 180], [74, 195], [73, 190], [78, 230], [75, 190], [73, 200], [70, 190], [72, 190], [72, 200], [73, 200], [73, 184], [75, 200], [75, 180], [74, 219], [76, 187], [73, 200], [74, 220], [75, 205], [75, 190], [72, 170], [73, 160], [73, 215], [72, 175], [74, 205], [78, 200], [76, 214], [73, 200], [74, 190], [75, 180], [70, 205], [75, 220], [71, 190], [72, 215], [78, 235], [75, 191], [73, 200], [73, 181], [71, 200], [75, 210], [77, 240], [72, 185], [69, 165], [73, 190], [74, 185], [72, 175], [70, 155], [75, 210], [70, 170], [72, 175], [72, 220], [74, 210], [73, 205], [74, 200], [76, 205], [75, 195], [80, 240], [72, 150], [75, 200], [73, 215], [74, 202], [74, 200], [73, 190], [75, 205], [75, 190], [71, 160], [73, 215], [75, 185], [74, 200], [74, 190], [72, 210], [74, 185], [74, 220], [74, 190], [73, 202], [76, 205], [75, 220], [72, 175], [73, 160], [73, 190], [73, 200], [72, 229], [72, 206], [72, 220], [72, 180], [71, 195], [75, 175], [75, 188], [74, 230], [73, 190], [75, 200], [79, 190], [74, 219], [76, 235], [73, 180], [74, 180], [74, 180], [72, 200], [74, 234], [74, 185], [75, 220], [78, 223], [74, 200], [74, 210], [74, 200], [77, 210], [70, 190], [73, 177], [74, 227], [73, 180], [71, 195], [75, 199], [71, 175], [72, 185], [77, 240], [74, 210], [70, 180], [77, 194], [73, 225], [72, 180], [76, 205], [71, 193], [76, 230], [78, 230], [75, 220], [73, 200], [78, 249], [74, 190], [79, 208], [75, 245], [76, 250], [72, 160], [75, 192], [75, 220], [70, 170], [72, 197], [70, 155], [74, 190], [71, 200], [76, 220], [73, 210], [76, 228], [71, 190], [69, 160], [72, 184], [72, 180], [69, 180], [73, 200], [69, 176], [73, 160], [74, 222], [74, 211], [72, 195], [71, 200], [72, 175], [72, 206], [76, 240], [76, 185], [76, 260], [74, 185], [76, 221], [75, 205], [71, 200], [72, 170], [71, 201], [73, 205], [75, 185], [76, 205], [75, 245], [71, 220], [75, 210], [74, 220], [72, 185], [73, 175], [73, 170], [73, 180], [73, 200], [76, 210], [72, 175], [76, 220], [73, 206], [73, 180], [73, 210], [75, 195], [75, 200], [77, 200], [73, 164], [72, 180], [75, 220], [70, 195], [74, 205], [72, 170], [80, 240], [71, 210], [71, 195], [74, 200], [74, 205], [73, 192], [75, 190], [76, 170], [73, 240], [77, 200], [72, 205], [73, 175], [77, 250], [76, 220], [71, 224], [75, 210], [73, 195], [74, 180], [77, 245], [71, 175], [72, 180], [73, 215], [69, 175], [73, 180], [70, 195], [74, 230], [76, 230], [73, 205], [73, 215], [75, 195], [73, 180], [79, 205], [74, 180], [73, 190], [74, 180], [77, 190], [75, 190], [74, 220], [73, 210], [77, 255], [73, 190], [77, 230], [74, 200], [74, 205], [73, 210], [77, 225], [74, 215], [77, 220], [75, 205], [77, 200], [75, 220], [71, 197], [74, 225], [70, 187], [79, 245], [72, 185], [72, 185], [70, 175], [74, 200], [74, 180], [72, 188], [73, 225], [72, 200], [74, 210], [74, 245], [76, 213], [82, 231], [74, 165], [74, 228], [70, 210], [73, 250], [73, 191], [74, 190], [77, 200], [72, 215], [76, 254], [73, 232], [73, 180], [72, 215], [74, 220], [74, 180], [71, 200], [72, 170], [75, 195], [74, 210], [74, 200], [77, 220], [70, 165], [76, 205], [70, 192], [75, 210], [74, 205], [75, 200], [73, 210], [71, 185], [71, 195], [72, 202], [73, 205], [73, 195], [72, 180], [69, 200], [73, 185], [78, 240], [71, 185], [73, 220], [75, 205], [76, 205], [70, 180], [74, 201], [77, 190], [75, 208], [79, 240], [72, 180], [77, 230], [73, 195], [75, 215], [75, 190], [75, 195], [73, 215], [73, 215], [76, 220], [77, 220], [75, 230], [70, 195], [71, 190], [71, 195], [75, 209], [74, 204], [69, 170], [70, 185], [75, 205], [72, 175], [75, 210], [73, 190], [72, 180], [72, 180], [72, 160], [76, 235], [75, 200], [74, 210], [69, 180], [73, 190], [72, 197], [72, 203], [75, 205], [77, 170], [76, 200], [80, 250], [77, 200], [76, 220], [79, 200], [71, 190], [75, 170], [73, 190], [76, 220], [77, 215], [73, 206], [76, 215], [70, 185], [75, 235], [73, 188], [75, 230], [70, 195], [69, 168], [71, 190], [72, 160], [72, 200], [73, 200], [70, 189], [70, 180], [73, 190], [76, 200], [75, 220], [72, 187], [73, 240], [79, 190], [71, 180], [72, 185], [74, 210], [74, 220], [74, 219], [72, 190], [76, 193], [76, 175], [72, 180], [72, 215], [71, 210], [72, 200], [72, 190], [70, 185], [77, 220], [74, 170], [72, 195], [76, 205], [71, 195], [76, 210], [71, 190], [73, 190], [70, 180], [73, 220], [73, 190], [72, 186], [71, 185], [71, 190], [71, 180], [72, 190], [72, 170], [74, 210], [74, 240], [74, 220], [71, 180], [72, 210], [75, 210], [72, 195], [71, 160], [72, 180], [72, 205], [72, 200], [72, 185], [74, 245], [74, 190], [77, 210], [75, 200], [73, 200], [75, 222], [73, 215], [76, 240], [72, 170], [77, 220], [75, 156], [72, 190], [71, 202], [71, 221], [75, 200], [72, 190], [73, 210], [73, 190], [71, 200], [70, 165], [75, 190], [71, 185], [76, 230], [73, 208], [68, 209], [71, 175], [72, 180], [74, 200], [77, 205], [72, 200], [76, 250], [78, 210], [81, 230], [72, 244], [73, 202], [76, 240], [72, 200], [72, 215], [74, 177], [76, 210], [73, 170], [76, 215], [75, 217], [70, 198], [71, 200], [74, 220], [72, 170], [73, 200], [76, 230], [76, 231], [73, 183], [71, 192], [68, 167], [71, 190], [71, 180], [74, 180], [77, 215], [69, 160], [72, 205], [76, 223], [75, 175], [76, 170], [75, 190], [76, 240], [72, 175], [74, 230], [76, 223], [74, 196], [72, 167], [75, 195], [78, 190], [77, 250], [70, 190], [72, 190], [79, 190], [74, 170], [71, 160], [68, 150], [77, 225], [75, 220], [71, 209], [72, 210], [70, 176], [72, 260], [72, 195], [73, 190], [72, 184], [74, 180], [72, 195], [72, 195], [75, 219], [72, 225], [73, 212], [74, 202], [72, 185], [78, 200], [75, 209], [72, 200], [74, 195], [75, 228], [75, 210], [76, 190], [74, 212], [74, 190], [73, 218], [74, 220], [71, 190], [74, 235], [75, 210], [76, 200], [74, 188], [76, 210], [76, 235], [73, 188], [75, 215], [75, 216], [74, 220], [68, 180], [72, 185], [75, 200], [71, 210], [70, 220], [72, 185], [73, 231], [72, 210], [75, 195], [74, 200], [70, 205], [76, 200], [71, 190], [82, 250], [72, 185], [73, 180], [74, 170], [71, 180], [75, 208], [77, 235], [72, 215], [74, 244], [72, 220], [73, 185], [78, 230], [77, 190], [73, 200], [73, 180], [73, 190], [73, 196], [73, 180], [76, 230], [75, 224], [70, 160], [73, 178], [72, 205], [73, 185], [75, 210], [74, 180], [73, 190], [73, 200], [76, 257], [73, 190], [75, 220], [70, 165], [77, 205], [72, 200], [77, 208], [74, 185], [75, 215], [75, 170], [75, 235], [75, 210], [72, 170], [74, 180], [71, 170], [76, 190], [71, 150], [75, 230], [76, 203], [83, 260], [75, 246], [74, 186], [76, 210], [72, 198], [72, 210], [75, 215], [75, 180], [72, 200], [77, 245], [73, 200], [72, 192], [70, 192], [74, 200], [72, 192], [74, 205], [72, 190], [71, 186], [70, 170], [71, 197], [76, 219], [74, 200], [76, 220], [74, 207], [74, 225], [74, 207], [75, 212], [75, 225], [71, 170], [71, 190], [74, 210], [77, 230], [71, 210], [74, 200], [75, 238], [77, 234], [76, 222], [74, 200], [76, 190], [72, 170], [71, 220], [72, 223], [75, 210], [73, 215], [68, 196], [72, 175], [69, 175], [73, 189], [73, 205], [75, 210], [70, 180], [70, 180], [74, 197], [75, 220], [74, 228], [74, 190], [73, 204], [74, 165], [75, 216], [77, 220], [73, 208], [74, 210], [76, 215], [74, 195], [75, 200], [73, 215], [76, 229], [78, 240], [75, 207], [73, 205], [77, 208], [74, 185], [72, 190], [74, 170], [72, 208], [71, 225], [73, 190], [75, 225], [73, 185], [67, 180], [67, 165], [76, 240], [74, 220], [73, 212], [70, 163], [75, 215], [70, 175], [72, 205], [77, 210], [79, 205], [78, 208]]
```

#import numpy as np

import numpy as np

#create 2D numpy array from baseball as np_baseball

np_baseball=np.array(baseball)

Print out the shape attribute of np_baseball. Use np_baseball.shape

print(np_baseball.shape)

SUBSETTING 2D NUMPY ARRAYS

- For regular Python lists, subsetting is a real pain. In 2D numpy arrays, however, it's pretty intuitive!
- Subsetting can also be helpful to acquire a range of element from an array.
- To get a range of element from an array simply use the start and end index numbers in bracket notation [start:end].
- The indexes before the comma refer to the rows.
- You can select a single row by passing the index number before the comma [1,:]
- The indexes after the comma refer to the columns
- Similarly you can select a single column by passing the index number after the comma[:,1]
- To select a combination of rows & columns you can use the row / column index number in square brackets [0:,1]

SUBSETTING 2D NUMPY ARRAYS - EXAMPLE

Selecting First row & First column in a NumPy Array.

```
#create an array
```

```
x = [["a", "b"], ["c", "d"]]
```

```
#import numpy as np
```

```
import numpy as np
```

```
#create numpy array
```

```
np_x = np.array(x)
```

```
#select a single row
```

```
np_x[1,:]
```

```
Out:['c', 'd']
```

```
#select a single column
```

```
np_x[:,1]
```

```
Out:['b', 'd']
```

```
#combination of rows & columns
```

```
np_x[0:,1]
```

```
Out:['b']
```

Np_x =

a	b
c	d

TASKS

Using the 2D array you created earlier, print out the 50th row of `np_baseball`.

ANSWER

#height as weight are available as np_baseball numpy 2D array

#import numpy as np

import numpy as np

#Print out the 50th row of np_baseball.

print(np_baseball[49,:])

TASKS

Make a new variable, `np_weight_lb_n`, containing the entire second column of `np_baseball`. Print out `np_weight_lb_n`

ANSWER

#height as weight are available as np_baseball numpy 2D array

#import numpy as np

import numpy as np

#create np_weight_lb, containing the entire second column of np_baseball

np_weight_lb=np_baseball[:,1]

print(np_weight_lb)

TASKS

Select the height (first column) of the 124th baseball player in `np_baseball` and print it out.

ANSWER

#height as weight are available as np_baseball numpy 2D array

#import numpy as np

import numpy as np

#print out height of 124th baseball player

print(np_baseball[123,:1])

2D ARITHMETIC

- Remember how you calculated the Body Mass Index for all baseball players?
- Numpy was able to perform all calculations element-wise (i.e. element by element).
- For 2D numpy arrays this isn't any different!
- You can combine matrices with single numbers, with vectors, and with other matrices.

2D ARITHMETIC – EXAMPLE (+)

Using arithmetic operators on 2D Arrays – Addition in an array

```
#creating an array
test_array = [[1, 2], [3, 4], [5, 6]]
#import numpy as np
import numpy as np
#create numpy array from test_array
np_test_array = np.array(test_array)
#addition to np_test_array
np_test_array = np_test_array + np.array([10, 10])
#print np_test_array
Print(np_test_array)
```

Out:

```
[[11 12] [13 14] [15 16]]
```


2D ARITHMETIC – EXAMPLE (X)

Using arithmetic operators on 2D Arrays - Multiplication

```
#creating an array
test_array = [[1, 2], [3, 4], [5, 6]]
#import numpy as np
import numpy as np
#create numpy array from test_array
np_test_array = np.array(test_array)
#multiplying np_test_array with 2
np_test_array=np_test_array * 2
#print np_test_array
Print(np_test_array)
```

Out:

```
[[ 2  4] [ 6  8] [10 12]]
```

TASKS

Create a Numpy Array `price_2020x` using the below matrix and add 100 to it.

Price 2019 \$	Price 2018 \$
690	199.00
199	192.00
959	913.00
683	129.00
188	510.00
592	207.00
245	507.00

ANSWER

```
#creating an array
```

```
price_2020x = [[690, 199], [199, 192], [959, 913], [683, 129], [188, 510], [592, 207], [245, 507]]
```

```
#import numpy as np
```

```
import numpy as np
```

```
#create numpy array from price_2020x
```

```
np_price_2020x = np.array(price_2020x)
```

```
#addition to np_price_2020x
```

```
np_price_2020x = np_price_2020x+ 100
```

```
#print np_price_2020x
```

```
print(np_price_2020x)
```

TASKS

Create a Numpy Array `price_2020x`. Multiply the Price in 2019 to the Inflation Rate (7 %) and Store the new price as `price2020x_adjusted`

Price 2019 \$	Price 2018 \$
690	199.00
199	192.00
959	913.00
683	129.00
188	510.00
592	207.00
245	507.00

IMPORTING MLB DATA FROM GITHUB

We have data with 3 columns representing height (in inches), weight (in pounds) and age (in years). It's stored in GitHub as a .CSV file. We will import this dataset using the read function from the Panda's Library; We will create a 2d Numpy array using this CSV File.

```
#importing libraries  
import pandas as pd  
import numpy as np
```

```
#importing MLB data using the read function from Panda's  
baseball = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/mlbData.csv")
```

```
#creating a NumPy array  
np_baseball = np.array(baseball)
```

TASKS

You want to convert the units of height and weight to metric (meters and kilograms respectively). As a first step, create a numpy array with three values: (0.0254, 0.453592 and 1). Name this array `np_conversion`.

Multiply `np_baseball` with `np_conversion` and print out the result.

Use below code to get dataset from github

```
#importing libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
#importing MLB data using the read function from Panda's
```

```
baseball = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/mlbData.csv")
```

```
#creating a NumPy array
```

```
np_baseball = np.array(baseball)
```

ANSWER

#importing libraries

```
import pandas as pd
```

```
import numpy as np
```

#importing MLB data using the read function from Panda's

```
baseball = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/mlbData.csv")
```

#creating a NumPy array

```
np_baseball = np.array(baseball)
```

Create numpy array: conversion

```
conversion=np.array([0.0254,0.453592,1])
```

Print out product of np_baseball and conversion

```
print(np_baseball*conversion)
```

STATISTICS – AVERAGE VS MEDIAN

You now know how to use numpy functions to get a better feeling for your data. It basically comes down to importing numpy and then calling several simple functions on the numpy arrays:

Let's try the mean & median functions:

```
import numpy as np
```

```
x = [1, 4, 8, 10, 12]
```

```
np.mean(x)
```

Out: 7

```
np.median(x)
```

Out: 8

MLB DATA SUMMARY STATISTICS - TASK BACKGROUND

The baseball data is available as a 2D numpy array with 3 columns (height, weight, age) and 1015 rows.

The name of this numpy array is `np_baseball`. After restructuring the data, however, you notice that some height values are abnormally high.

Let's try using some summary statistics on the MLB Dataset.

TASKS

Print out the mean and median of players height from `np_baseball_conversion` numpy array.

ANSWER

```
# np_baseball is available
# Import numpy
import numpy as np
# Create np_height_in from np_baseball
np_height_in=np.array(np_baseball[:,0])
# Print out the mean of np_height_in
print(np.mean(np_height_in))
# Print out the median of np_height_in
print(np.median(np_height_in))
```

STATISTICS – STANDARD DEVIATION

Standard deviation is a measure of the amount of variation or dispersion of a set of values.

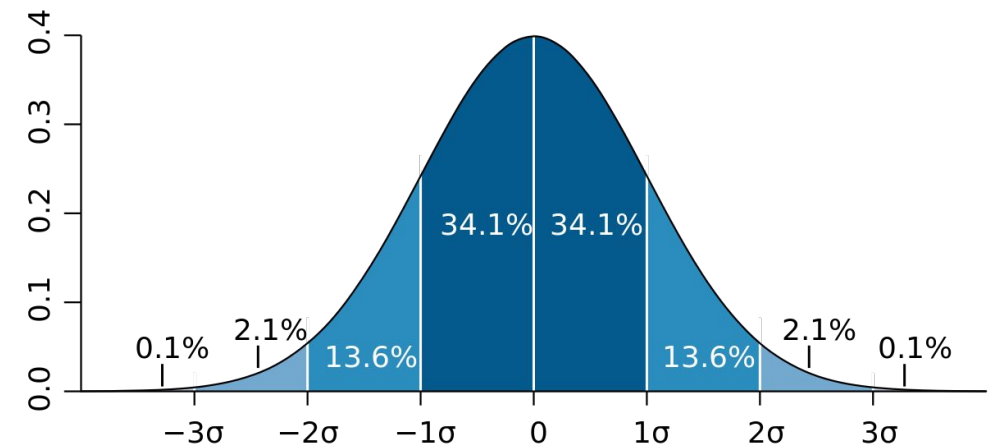
Let's try the standard deviation function:

```
import numpy as np
```

```
x = [1, 4, 8, 10, 12]
```

```
np.std(x)
```

Out: 4



TASKS

Use `np.std()` on the first column (height) of `np_baseball_conversion` to calculate standard deviation.

Print out standard deviation using `print()`

ANSWER

```
# np_baseball is available
# Import numpy
import numpy as np
# Print out the standard deviation on height. Replace 'None'
stddev = np.std(np_baseball[:,0])
print(stddev)
```

STATISTICS – CORRELATION

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related.

For example, height and weight are related; taller people tend to be heavier than shorter people.

Let's try the correlation function:

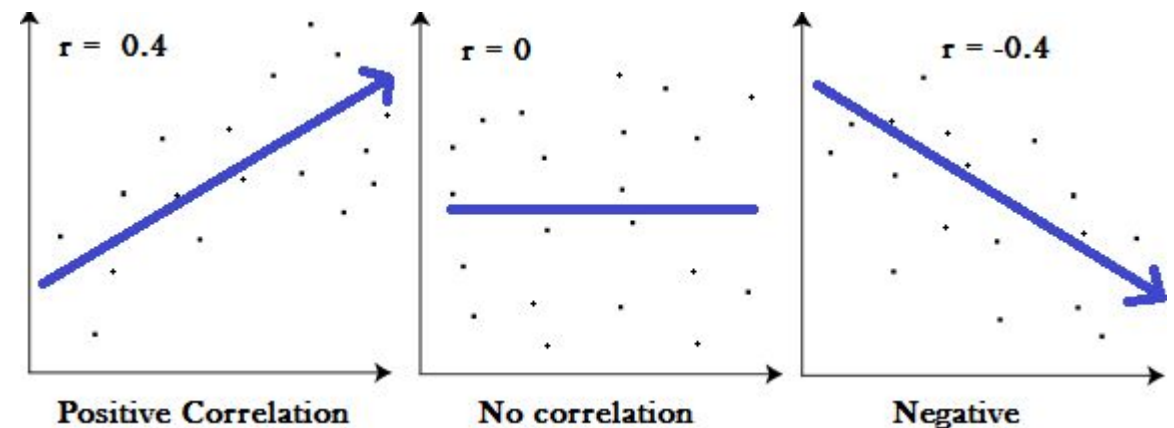
```
import numpy as np
```

```
x = [1, 4, 8, 10, 12]
```

```
y = [3, 5, 2, 15, 20]
```

```
np.corrcoef(x,y)
```

Out: **0.79350562**



TASKS

Do tall players tend to be heavier?

Use `np.corrcoef()` to find the correlation between the height (first column) and weight (second column) of `np_baseball_conversion`.

Store the Value of the Correlation in `corr`

ANSWER

```
# np_baseball is available
```

```
# Import numpy
```

```
import numpy as np
```

```
# Print out correlation between first and second column. Replace 'None'
```

```
corr = np.corrcoef(np_baseball[:,0],np_baseball[:,1])
```

```
Print(corr)
```

BLEND IT TOGETHER

In the last few exercises you've learned everything there is to know about heights and weights of baseball players. Now it's time to dive into another sport: soccer.

You've contacted FIFA for some data and they handed you two lists. The lists are the following:

```
positions = ['GK', 'M', 'A', 'D', ...]
```

```
heights = [191, 184, 185, 180, ...]
```

BLEND IT TOGETHER – TASK BACKGROUND

Each element in the lists corresponds to a player. The first list, `positions`, contains strings representing each player's position. The possible positions are: 'GK' (goalkeeper), 'M' (midfield), 'A' (attack) and 'D' (defense). The second list, `heights`, contains integers representing the height of the player in cm. The first player in the lists is a goalkeeper and is pretty tall (191 cm).

You're fairly confident that the median height of goalkeepers is higher than that of other players on the soccer field. Some of your friends don't believe you, so you are determined to show them using the data you received from FIFA and your newly acquired Python skills.

TASKS

Use the code below to create heights and positions from FIFA dataset from github.

```
import pandas as pd
```

```
fifa = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/fifapos.csv")
```

- Convert heights and positions, which are regular lists, to numpy arrays.
- Call them np_heights and np_positions.
- Print out both np_heights and np_positions.

ANSWER

```
import pandas as pd
import numpy as np
#importing Fifa dataset
fifa = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/fifapos.csv")
# Convert positions and heights to numpy arrays: np_positions, np_heights
np_positions=np.array(fifa[:0])
np_heights=np.array(fifa[:1])
#print out both np_position and np_heights
print(np_heights)
```

TASKS

Extract all the heights of the goalkeepers.

You can use a little trick here: use `np_fifa[index] == 'GK' .`

Assign the result to `gk_heights`.

ANSWER

```
# np_positions and np_positions are available numpy array
# Import numpy
import numpy as np
# Heights of the goalkeepers:gk_heights
gk_heights=np_heights[np_positions=='GK']
#print out goalkeepers positions
print(gk_heights)
```

TASKS

Extract all the heights of the Attack Place .

You can use a little trick here: use `np_fifa[index] == 'A'` .

Assign the result to `attack_heights`.

TASKS

Extract all the heights of all the other players.

This time use `np_positions != 'GK'`.

Assign the result to `other_heights`.

ANSWER

```
# np_positions and np_positions are available numpy array
```

```
# Import numpy
```

```
import numpy as np
```

```
# Heights of the other players: other_heights
```

```
other_heights=np_heights[np_positions!='GK']
```

```
#print out other players height
```

```
print(other_heights)
```

TASKS

Print out the median height of the goalkeepers using `np.median()`.

ANSWER

```
# np_positions and np_positions are available numpy array
# Import numpy
import numpy as np
# Heights of the goalkeepers: gk_heights
gk_heights=np_heights[np_positions=='GK']
#Print out the median height of the goalkeepers using np.median()
median_gk=np.median(gk_heights)
#print out goalkeepers median height
print(median_gk)
```

TASKS

Do the same for the other players. Print out their median height of other players except goalkeepers.

ANSWER

```
# np_positions and np_positions are available numpy array
# Import numpy
import numpy as np
# Heights of the other players: other_heights
other_heights=np_heights[np_positions!='GK']
# median height of the other players using np.median()
median_other_heights=np.median(other_heights)
#print out other players median height
print(median_other_heights)
```



Q&A



python
THANK YOU