Functions

Familiar Functions

print(), type(),str(), int(), bool() and float()

In [1]:

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

# Print out type of var1
print(type(var1))

# Print out length of var1
print(len(var1))

# Convert var2 to an integer: out2
out2 = int(var2)
```

```
<class 'list'>
4
```

you have to ask for information about a function with another function: help(). In IPython specifically, you can also use ? before the function name.

In [2]:

```
help(max)
?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.
```

Multiple Arguments

In [3]:

```
# Create lists first and second
first = [11.25, 18.0, 20.0]
second = [10.75, 9.50]

# Paste together first and second: full
full = first + second

# Sort full in descending order: full_sorted
full_sorted = sorted(full,reverse = True)

# Print out full_sorted
print(full_sorted)
```

```
[20.0, 18.0, 11.25, 10.75, 9.5]
```

Methods

Methods: call functions on objects

In [4]:

```
# string to experiment with: place
place = "poolhouse"

# Use upper() on place: place_up
place_up = place.upper()

# Print out place and place_up
print(place + place_up)

# Print out the number of o's in place
print(place.count('o'))
```

```
poolhousePOOLHOUSE
3
```

In [5]:

```
# Create list areas
areas = [11.25, 18.0, 20.0, 10.75, 9.50]

# Print out the index of the element 20.0
print(areas.index(20.0))

# Print out how often 9.50 appears in areas
print(areas.count(9.5))
```

```
2
1
```

In [6]:

```
# Use append twice to add poolhouse and garage size
areas.append(24.5)
areas.append(15.45)

# Print out areas
print(areas)

# Reverse the orders of the elements in areas
areas.reverse()

# Print out areas
print(areas)
```

```
[11.25, 18.0, 20.0, 10.75, 9.5, 24.5, 15.45]
[15.45, 24.5, 9.5, 10.75, 20.0, 18.0, 11.25]
```

Packages

In [7]:

```
# Definition of radius
r = 0.43

# Import the math package
import math

# Calculate C
C = 2*math.pi * r

# Calculate A
A = (C*r)/2

# Build printout
print("Circumference: " + str(C))
print("Area: " + str(A))
```

```
Circumference: 2.701769682087222
Area: 0.5808804816487527
```

In [8]:

```
# Definition of radius
r = 192500

# Import radians function of math package
from math import radians

# Travel distance of Moon over 12 degrees. Store in dist.
dist = r * radians(12)

# Print out dist
print(dist)
```

```
40317.10572106901
```

Suppose you want to use the function inv(), which is in the linalg subpackage of the scipy package. You want to be able to use this function as follows:

my_inv([[1,2], [3,4]])

Which import statement will you need in order to run the above code without an error?

- import scipy
- import scipy.linalg
- from scipy.linalg import my_inv
- from scipy.linalg import inv as my_inv

NumPy

In [10]:

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

# Print out type of np_baseball
print(type(np_baseball))
```

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

In [15]:

```
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, 7
3, 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, 74, 75, 75, 78, 76, 75, 69, 75,
72, 75, 73, 74, 75, 75, 73]
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, 20
0, 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, 20
5, 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, 18
0, 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, 22
5, 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, 19
4, 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, 20
```

```
5, 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, 19
7, 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, 19
3, 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, 20
0, 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, 21
0, 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, 20
6, 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, 19
4, 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, 19
5, 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, 18
0, 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, 18
0, 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, 21
3, 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, 22
0, 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, 19
5, 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, 22
0, 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, 18
0, 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, 24
5, 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, 24
0, 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, 19
0, 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, 22
0, 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, 18
5, 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, 23
0, 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, 20
0, 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, 20
7, 205, 208, 185, 190, 170, 208, 225, 190, 225, 185, 180, 165, 240, 220, 212, 163, 215,
175, 205, 210, 205, 208, 215, 180, 200, 230, 211, 230, 190, 220, 180, 205, 190, 180, 20
5, 190, 195]
```

In [12]:

```
# Import numpy
import numpy as np

# Create a numpy array from height_in: np_height_in
np_height_in = np.array(height_in)

# Print out np_height_in
print(np_height_in)

# Convert np_height_in to m: np_height_m
np_height_m = 0.0254 * np_height_in

# Print np_height_m
print(np_height_m)
```

```
[74 74 72 ... 75 75 73]
[1.8796 1.8796 1.8288 ... 1.905  1.905  1.8542]
```

In [16]:

```
# Import numpy
import numpy as np

# Create array from height_in with metric units: np_height_m
np_height_m = np.array(height_in) * 0.0254

# Create array from weight_lb with metric units: np_weight_kg
np_weight_kg = np.array(weight_lb) * 0.453592

# Calculate the BMI: bmi
bmi = np_weight_kg/np_height_m ** 2

# Print out bmi
print(bmi)
```

```
[23.11037639 27.60406069 28.48080465 ... 25.62295933 23.74810865
 25.72686361]
```

In [17]:

```
# 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

# Create the light array
light  =  np.array(bmi) < 21

# Print out light
print(light)

# Print out BMIs of all baseball players whose BMI is below 21
print(bmi[light])
```

```
[False False False ... False False False]
[20.54255679 20.54255679 20.69282047 20.69282047 20.34343189 20.34343189
 20.69282047 20.15883472 19.4984471  20.69282047 20.9205219 ]
```

Have a look at this line of code:

np.array([True, 1, 2]) + np.array([3, 4, False]) Can you tell which code chunk builds the exact same Python object?

- np.array([True, 1, 2, 3, 4, False])
- np.array([4, 3, 0]) + np.array([0, 2, 2])
- np.array([1, 1, 2]) + np.array([3, 4, -1])
- np.array([0, 1, 2, 3, 4, 5])

Subsetting NumPy Arrays

In [18]:

```
# Import numpy
import numpy as np

# Store weight and height lists as numpy arrays
np_weight_lb = np.array(weight_lb)
np_height_in = np.array(height_in)

# Print out the weight at index 50
print(np_weight_lb[50])

# Print out sub-array of np_height_in: index 100 up to and including index 110
print(np_height_in[100:111])
```

```
200
[73 74 72 73 69 72 73 75 75 73 72]
```

2D NumPy Arrays

In [19]:

```
# Create baseball, a list of lists
baseball = [[180, 78.4],
            [215, 102.7],
            [210, 98.5],
            [188, 75.2]]

# Import numpy
import numpy as np

# Create a 2D numpy array from baseball: np_baseball
np_baseball=np.array(baseball)

# Print out the type of np_baseball
print(type(np_baseball))

# Print out the shape of np_baseball
print(np_baseball.shape)
```

```
<class 'numpy.ndarray'>
(4, 2)
```

In [20]:

```
# Create a 2D numpy array from baseball: np_baseball
np_baseball = np.array(baseball)

# Print out the shape of np_baseball
print(np_baseball.shape)
```

```
(4, 2)
```

In [21]:

```
# Print out the 3rd row of np_baseball
print(np_baseball[2,:])

# Select the entire second column of np_baseball: np_weight_lb
np_weight_lb = np_baseball[:,1]
```

```
[210.   98.5]
```

In [22]:

```
# 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))
```

```
73.6896551724138
74.0
```

In [23]:

```
# Print mean height (first column)
avg = np.mean(np_baseball[:,0])
print("Average: " + str(avg))

# Print median height. Replace 'None'
med = np.median(np_baseball[:,0])
print("Median: " + str(med))

# Print out the standard deviation on height. Replace 'None'
stddev =np.std(np_baseball[:,0])
print("Standard Deviation: " + str(stddev))

# Print out correlation between first and second column. Replace 'None'
corr = np.corrcoef(np_baseball[:,0],np_baseball[:,1])
print("Correlation: " + str(corr))
```

```
Average: 198.25
Median: 199.0
Standard Deviation: 14.635146053251399
Correlation: [[1.         0.95865738]
 [0.95865738 1.        ]]
```

In [ ]:

```

```