TerryZhou-Module1

March 27, 2022

```
In [2]: print("Hello World")
```

```
Hello World
```

The previous code prints out the statement "Hello World", without quotation marks.

1 Getting used to Jupyter

```
In [12]: a = 1
         type(a)
```

```
Out[12]: int
```

```
In [4]: b = 1.1
        type(b)
```

```
Out[4]: float
```

```
In [5]: c = "c"
        type(c)
```

```
Out[5]: str
```

```
In [13]: a = float(format(a, '.2f'))
         print(a)
         type(a)
```

```
1.0
```

```
Out[13]: float
```

It is not possible to change variable "a" into a 2 digit decimal.

```
In [17]: a = 1
         a = str(a)
         type(a)
```

```
Out[17]: str

In [18]: b = int(b)
         type(b)

Out[18]: int

In [19]: b = str(b)
         type(b)

Out[19]: str

In [20]: d = [1, 2, 3, 4]
         type(d)

Out[20]: list

In [21]: e = ("a", "b", "c", "d", "e")
         type(e)

Out[21]: tuple

In [23]: f = {"a": "a", "b": "b", "c": 1, "d": 2, "e": 3}
         type(f)

Out[23]: dict

In [24]: d = list(d)
         type(d)

Out[24]: list

In [85]: class School:
             def __init__(self, State = "NY", name = "University of Rochester"):
                 self.Name = name
                 self.State = State

             def printname(self):
                 return self.Name

             def printstate(self):
                 return self.State

Out[85]: 'University of Rochester'

In [89]: class Student:
             def __init__(self, Name, ID, state = "NY", name = "University of Rochester"):
                 self.Name = Name
                 self.ID = ID
                 school = School(state, name)
```

```
            self.School = school

        student1 = Student("Terry", 31375672)
        type(student1)

        student2 = Student("Nani", 31323893)
        type(student2)
```

Out[89]: __main__.Student

```
In [88]: class Undergrad:
             def __init__(self, major, year, name, ID, state = "NY", school = "University of R
                 self.Major = major
                 self.Year = year
                 self.Student = Student(name, ID)
                 self.School = School(state, school)

         undergrad1 = Undergrad("Computational Biology", 2022, "Terry", 31375672)
         type(undergrad1)

         undergrad2 = Undergrad("Molecular Genetics", 2022, "Nani", 31323893)
         type(undergrad1)
```

Out[88]: __main__.Undergrad

```
In [90]: def listlength(l):
             length = 0
             for i in l:
                 length = length + 1
             return length
```

```
In [105]: def sum(int1, int2, int3):
              tot = int1 + int2 + int3
              return tot
```

```
In [100]: def multiply(int1, int2, int3):
              product = int1 * int2 * int3
              return product
```

```
In [109]: def sum_and_multiply(int1, int2, int3):
              s = sum(int1, int2, int3)
              m = multiply(int1, int2, int3)
              return s, m
```

```
In [107]: def sum_or_multiply(int1, int2, int3, boolean):
              if boolean == True:
                  x = sum(int1, int2, int3)
                  return x
              elif boolean == False:
                  y = multiply(int1, int2, int3)
                  return y
```

```
In [111]: def sumlist(lis):
              tot = 0
              for i in lis:
                  tot = tot + i
              return tot
```

```
In [116]: def dictionary_value(dictionary):
              while True:
                  x = dictionary.items()
                  return x

          di = {"a": "apple", "b": "banana", "c": "cherry", "d": "dragonfruit"}
          dictionary_value(di)
```

```
Out[116]: dict_items([('a', 'apple'), ('b', 'banana'), ('c', 'cherry'), ('d', 'dragonfruit')])
```

```
In [122]: def five_three(int):
              nums = list(range(0, int + 1))
              div_5_3 = []
              for i in nums:
                  if (i % 5 == 0) | (i % 3 == 0):
                      div_5_3.append(i)
              div_15 = []
              for j in div_5_3:
                  if j % 15 != 0:
                      div_15.append(j)
              return div_15
```

```
In [133]: def pos(nums):
              for i in nums:
                  if i < 0:
                      return False
                  else:
                      return True
          def sumlist(nums):
              if len(nums) == 0:
                  return 0
              elif pos(nums) == False:
                  raise AssertionError("There is at least one negative value in this list.")
              else:
                  return nums[0] + sumlist(nums[1:])
```

The base case is when `len(nums) == 0`.

```
In [143]: def harmonic_sum(nums):
              if len(nums) == 0:
                  return 0
              else:
                  for i in nums[0:19]:
                      return (1 / nums[0]) + harmonic_sum(nums[1:20])
```

The base case is when `len(nums) == 0`.

```
In [149]: def Fibonacci(n):
              if n <= 1:
                  return n
              else:
                  return Fibonacci(n - 1) + Fibonacci(n - 2)


          [Fibonacci(n) for n in range(20)]
```

```
Out[149]: [0,
           1,
           1,
           2,
           3,
           5,
           8,
           13,
           21,
           34,
           55,
           89,
           144,
           233,
           377,
           610,
           987,
           1597,
           2584,
           4181]
```

The code drew out a dandelion. While each section is being drawn, the line and the cursor are red, but once they are done being drawn they turn blue.

```
In [174]: import pandas as pd
          iris = pd.read_csv("iris.data")
          print(iris.loc[0:4,])
          iris.info()
```

```
   sepal length in cm  sepal width in cm  petal length in cm  \
0                 5.1                3.5                 1.4
1                 4.9                3.0                 1.4
2                 4.7                3.2                 1.3
3                 4.6                3.1                 1.5
4                 5.0                3.6                 1.4

   petal width in cm        class
0                0.2  Iris-setosa
```

```
1                0.2  Iris-setosa
2                0.2  Iris-setosa
3                0.2  Iris-setosa
4                0.2  Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   sepal length in cm  150 non-null    float64
 1   sepal width in cm   150 non-null    float64
 2   petal length in cm  150 non-null    float64
 3   petal width in cm   150 non-null    float64
 4   class               150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

There are 5 attributes and 150 observations in the iris dataset.

```
In [170]: import numpy
          petal_length = iris[["petal length in cm"]].to_numpy()
          print(petal_length)
          len(petal_length)
```

```
[[1.4]
 [1.4]
 [1.3]
 [1.5]
 [1.4]
 [1.7]
 [1.4]
 [1.5]
 [1.4]
 [1.5]
 [1.5]
 [1.6]
 [1.4]
 [1.1]
 [1.2]
 [1.5]
 [1.3]
 [1.4]
 [1.7]
 [1.5]
 [1.7]
 [1.5]
 [1. ]
```

[1.7]
[1.9]
[1.6]
[1.6]
[1.5]
[1.4]
[1.6]
[1.6]
[1.5]
[1.5]
[1.4]
[1.5]
[1.2]
[1.3]
[1.5]
[1.3]
[1.5]
[1.3]
[1.3]
[1.3]
[1.6]
[1.9]
[1.4]
[1.6]
[1.4]
[1.5]
[1.4]
[4.7]
[4.5]
[4.9]
[4.]
[4.6]
[4.5]
[4.7]
[3.3]
[4.6]
[3.9]
[3.5]
[4.2]
[4.]
[4.7]
[3.6]
[4.4]
[4.5]
[4.1]
[4.5]
[3.9]
[4.8]

[4.]
[4.9]
[4.7]
[4.3]
[4.4]
[4.8]
[5.]
[4.5]
[3.5]
[3.8]
[3.7]
[3.9]
[5.1]
[4.5]
[4.5]
[4.7]
[4.4]
[4.1]
[4.]
[4.4]
[4.6]
[4.]
[3.3]
[4.2]
[4.2]
[4.2]
[4.3]
[3.]
[4.1]
[6.]
[5.1]
[5.9]
[5.6]
[5.8]
[6.6]
[4.5]
[6.3]
[5.8]
[6.1]
[5.1]
[5.3]
[5.5]
[5.]
[5.1]
[5.3]
[5.5]
[6.7]
[6.9]

```
 [5. ]
 [5.7]
 [4.9]
 [6.7]
 [4.9]
 [5.7]
 [6. ]
 [4.8]
 [4.9]
 [5.6]
 [5.8]
 [6.1]
 [6.4]
 [5.6]
 [5.1]
 [5.6]
 [6.1]
 [5.6]
 [5.5]
 [4.8]
 [5.4]
 [5.6]
 [5.1]
 [5.1]
 [5.9]
 [5.7]
 [5.2]
 [5. ]
 [5.2]
 [5.4]
 [5.1]]
```

Out[170]: 150

There are no missing values because there are 150 values in the array, which is the same as the number of values in the entire dataset.

In [175]:
```
iris["numeric species classification"] = petal_length
```

In [189]:
```
classification = []
for i in range(0, len(petal_length)):
    if petal_length[i] < 3:
        classification.append("Short")
    else:
        classification.append("Long")
iris["petal length classification"] = classification
print(len(iris[iris['petal length classification'] == 'Short']), "values are assigned
```

```
50 values are assigned 'Short'.
```