learnpython

September 16, 2015

1 Jupyter Task 3

Sidy Danioko

Sean Paradiso

1.0.1 Python Syntax

- Comments in python are anything preceeded by a '#'

```
In [ ]: # This is a comment
```

- Declaring a variable does not require a type

```
In [7]: a = 5
        myName = "Sean"
```

- The syntax for printing output is print("string1" + "string2")

```
In [8]: a = 5
        print a
        print myName
```

```
5
Sean
```

1.0.2 Python Data Types

- The boolean data type is a binary taking True and/or False

```
In [16]: a = 2
         b = 2

         if False:
             print "They are the same"

         else:
             print "They are different"
```

```
They are different
```

- The integer data type is a primitive data type used for integers

```
In [18]: a = 3
         type(a)
```

```
Out[18]: int
```

- The float data type is a primitive data type used for real numbers

In [21]:
```
a = 6.3
b = 2.6

c = a / b

print(c)

type(c)
```

```
2.42307692308
```

Out[21]: float

- The string data type is a primitive data type used for an array of characters

In [22]:
```
myName = "Sean Paradiso"

print(myName)

type(myName)
```

```
Sean Paradiso
```

Out[22]: str

- The list data type is a primitive data type used for a collection of items of the same or different types in a specific order

In [40]:
```
u = range(0,10)

print(u)

type(u)
print u[0]
print u[1]
u[2]= 100
print u[2]
print u

type(u)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
0
1
100
[0, 1, 100, 3, 4, 5, 6, 7, 8, 9]
```

Out[40]: list

- The tuple data type is an immutable ordered list

In [32]:
```
myTup = (1,2,3,4,5,6,7,8)

print myTup
type(myTup)

myTup(0) = 30 #We cannot change values within the tuple
```

```
  File "<ipython-input-32-6ef0b3d75e39>", line 6
    myTup(0) = 30
SyntaxError: can't assign to function call
```

- The dictionary data type is a key-value system

```
In [39]: FamilyAge = {"Mom": 100, "Dad": 90, "Brother": 50}

         print(FamilyAge)

         for item in FamilyAge:
             print item

         for value in FamilyAge:
             print FamilyAge[value]

         type(FamilyAge)
```

```
{'Dad': 90, 'Brother': 50, 'Mom': 100}
Dad
Brother
Mom
90
50
100
```

```
Out[39]: dict
```

- The set data type is similar to a list with the exception of not allowing duplicates

```
In [46]: mySet = set([1, 2, 3, 3, 3, 4, 5])

         print(mySet)
         type(mySet)

         #Let's add some items to mySet

         mySet.add(8)
         mySet.add("You the man")

         print(mySet)

         type(mySet)
```

```
set([1, 2, 3, 4, 5])
set([1, 2, 3, 4, 5, 8, 'You the man'])
```

```
Out[46]: set
```

1.0.3 Python Control Structures

- The syntax for an if-then-else control structure is used by initiating with the key word "if" followed by a test

```
In [53]: a = input("Please enter an integer: ")
         if(a%2==0):
             print " a is an even number"
         else:
             print " a is an odd number"

         first = raw_input("Please enter your first name :")

         if(len(first) > 4):
             print " Your name is too long"
         else:
             print " Your name is short"
```

```
Please enter an integer: 5
 a is an odd number
Please enter your first name :sdivhjsldivilsdj
 Your name is too long
```

- The syntax for a for loop control structure is used by initiating with the key word "for" which will iterrate through a list. (This means the final value within the given range is the stopping condition and will not be evaluated)

```
In [55]: #This for loop prints the squares of the values within the specified range

         for i in range(1,10):
             print i**2
```

```
1
4
9
16
25
36
49
64
81
```

- The syntax for a for while control structure is used by initiating with the key word "while" which will iterrate through a list until a condition is no longer met. (This means the final value within the given range is the stopping condition and will not be evaluated)

```
In [63]: while True:
             name = raw_input("Please a give me a name or type x to exit: ")
             if name == "x":break
             else:
                 print "Keep going"
```

```
Please a give me a name or type x to exit: 5
Keep going
Please a give me a name or type x to exit: x
```

1.0.4 Python Organizational Structures

- The syntax for defining a function with a name as an organizational structure creates a new function through the use of the key word "def" and returns a value with a return statement

```
In [8]: def sub(x, y):
            print "x is {0} and y is {1}".format(x, y)
            return x - y

        sub(12, 4)
```

```
x is 12 and y is 4
```

```
Out[8]: 8
```

- The syntax for including a docstring is an indented block surrounded by quotes

```
In [9]: def sub(x, y):
            "Return the difference of user define x and y" #<- the preceding is a docstring
            print "x is {0} and y is {1}".format(x, y)
            return x - y

        sub(12, 4)
```

```
x is 12 and y is 4
```

```
Out[9]: 8
```

- The syntax for *args is a list in which superfluous arguments are placed

```
In [26]: def PlayAround(a,b,c):
             return a+b*c

         mytuple=(2,3,5)
         PlayAround(*mytuple)

         def mylist(*food):
             print food

         mylist('apples')
         mylist('apples','chicke','rice')
```

```
('apples',)
('apples', 'chicke', 'rice')
```

- The syntax for kwargs is taking additional arguments and places them in a dictionary with a title of your choosing. This title can be anything as long as it is preceeded by two asterisks (in this example we have kwargs)

```
In [57]: def myfamily(**kwargs):
             for name, age in kwargs.items():
                 print '{0} is {1} years old'.format(name, age)

         myfamily(Dad = 100, Mom = 90)

         def print_table(**kwargs):
             for key, values in kwargs.items():
                 print(key,values)

         print_table(a= 10,b = -3, c = 100)
```

```
Dad is 100 years old
Mom is 90 years old
('a', 10)
('c', 100)
('b', -3)
```