Conding with Python

April 9, 2018

0.1 Python primitive data types

Python provides multiple variables types to hold data within. The type of the variable is determined by the assignment operation. The programmer needs not to explicitly declare the variable type. The following is a list of data types.

0.1.1 Boolean Data Type

Variables of this types has only two values, either `True` or `False`. This type of variables is very useful when we need to determine the control flow of the program depending on a truth value of some thing. The name boolean comes after George Boole, a great English mathematician who works on converting logic to mathematical operations. One branches of math is named after him, Boolean Algebra.

0.1.2 Numeric Types

This category holds five data types, `int` for integers, `float` for decimal value numbers, and, interestingly, `complex` for complex numbers.

The following code gives example of variables of each data type:

```
In [13]: int_variable = int(5) # you can write: intVariable = 5
         float_variable = float(-5.5) # you can write: floatVariable = 5.5
         complex_variable = complex(5,4)

         print(int_variable)
         print(float_variable)
         print(complex_variable)
```

```
5
-5.5
(5+4j)
```

You can convert between these data types as well. However, keep in mind that converting from float values to integer values will truncate the decimal part of the number.

```
In [14]: to_float = float(int_variable)
         print(to_float)
```

```
        to_int = int(float_variable)
        print(to_int)
        to_complex = complex(int_variable, float_variable)
        print(to_complex)
```

```
5.0
-5
(5-5.5j)
```

There are plenty of mathematical operations you can do on numbers. The following are just examples. **The table is taken from: https://www.programiz.com/python-programming/modules/math**. *Some functions in the original table are not in this table. Go there for more functions*

List of Functions in Python Math Module

- ceil(x)

Returns the smallest integer greater than or equal to x.

- copysign(x, y)

Returns x with the sign of y

- fabs(x)

Returns the absolute value of x

- factorial(x)

Returns the factorial of x

- floor(x)

Returns the largest integer less than or equal to x

- fmod(x, y)

Returns the remainder when x is divided by y

- isinf(x)

Returns True if x is a positive or negative infinity

- modf(x)

Returns the fractional and integer parts of x

- trunc(x)

Returns the truncated integer value of x

- exp(x)

Returns e**x

- log(x[, base])

Returns the logarithm of x to the base (defaults to e)

- log2(x)

Returns the base-2 logarithm of x

- log10(x)

Returns the base-10 logarithm of x

- pow(x, y)

Returns x raised to the power y

- sqrt(x)

Returns the square root of x

- acos(x)

Returns the arc cosine of x

- asin(x)

Returns the arc sine of x

- atan(x)

Returns the arc tangent of x

- atan2(y, x)

Returns atan(y / x)

- cos(x)

Returns the cosine of x

- sin(x)

Returns the sine of x

- tan(x)

Returns the tangent of x

- degrees(x)

Converts angle x from radians to degrees

- radians(x)

Converts angle x from degrees to radians

- acosh(x)

Returns the inverse hyperbolic cosine of x

- asinh(x)

Returns the inverse hyperbolic sine of x

- atanh(x)

Returns the inverse hyperbolic tangent of x

- cosh(x)

Returns the hyperbolic cosine of x

- sinh(x)

Returns the hyperbolic cosine of x

- tanh(x)

Returns the hyperbolic tangent of x

- erf(x)

Returns the error function at x

- erfc(x)

Returns the complementary error function at x

- gamma(x)

Returns the Gamma function at x

- lgamma(x)

Returns the natural logarithm of the absolute value of the Gamma function at x

- pi

Mathematical constant, the ratio of circumference of a circle to it's diameter (3.14159...)

- e

mathematical constant e (2.71828...)

```
In [125]: # Most of these functions are in the math module.
          # we need to import that moduel in order to use them.
          import math
          x = 2048
          y = 7
          z = 1.25
          w = 1.75

          print(math.ceil(z))
          print(math.floor(z))
          print(math.modf(z))
          print(math.factorial(y))
          print(math.log2(x))
          print(math.exp(math.e)) # e^e
          print(math.pi*math.pow(7,2)) # pi * 7^2
          print(round(z))
          print(round(w))
          print(math.sqrt(w))
```

```
2
1
(0.25, 1.0)
5040
11.0
15.154262241479262
153.93804002589985
1
2
1.3228756555322954
```

0.1.3 Strings

Strings are the data type representing a sequence of characters. Almost every programming language has its way of manipulating strings and python is not an exception.

As we have with numeric data types, strings have their own functions and methods. In this section, we will try to delve a little bit on string functions and methods.

```
In [20]: # Strings can be defined double quoted as follows:
         a = "I am a string"
         # or single quoted as follows:
         b = 'I am another string!'
         print(a)
         print(b)
```

```
I am a string
I am another string!
```

String functions and properties

- **Strings can be compared**

By comparing tow strings, the alphabetically greater string will be returned. Look at the following:

```
In [16]: a = "a"
         b = 'b'
         print (a>b)

         a = "ai"
         b = "ak"
         print(a<b)

False
True
```

- **Length of the string** `len()` function will return the number of characters in a given string. This function is very useful!

```
In [17]: a = "this is string!" # there are 15 characters!! dont forget to count spaces :)
         print(len(a))

15
```

- **String concatenation**

We can put two strings together in one string by just using the '+' operator. as follows:

```
In [22]: a = "This is a string!"
         # do not forget to add space in the last of the first concatenated string
         # or in the first of the second string to so that the resulted string looks nice!
         b = " and this is a second string!"
         # you can have multiple strings in the print statement!!
         print("the resulted string is:", a+b)

the resulted string is: This is a string! and this is a second string!
```

- **Indexing**

You can retrieve any character from a string by indexing it in a square brackets [ ]! **Do not forget that the first character in the string is indexed by 0.** *Remember that* indexing a string with a greater, or lower, number will fire a python error!

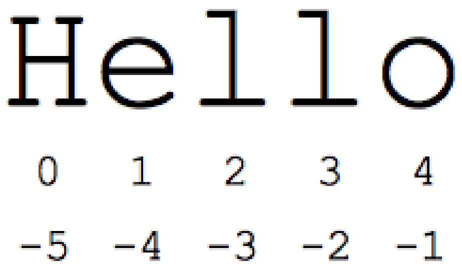

Figure 1

```
In [117]: # the value of variable a is "this is string" from the previous runnable cell.
          print("the sixth character of 'a' string is:", a[5])
          print("the first character of 'a' string is:", a[0])

          # here we will print the last element of the string.
          # by using len() function, it is not necessary to know the length of the string.
          # we need to subtract it by one becuase of the first index which is ZERO.
          print("The last character of 'a' is: "+a[len(a)-1])

          # We may use floor() function with len() to get the middle element of the string as
          # remember that we cannot use float "real" numbers to index string. Integers are onl
          print("the middle character of 'a' is:"+a[math.floor(len(a)/2)])

          # this code will fire an error:
          #print(a[len(a)])
```

```
the sixth character of 'a' string is: i
the first character of 'a' string is: t
The last character of 'a' is: !
the middle character of 'a' is:c
```

Interestingly, Python allows for negative indexing. What does this mean? It means that we can use negative numbers to retrieve characters from strings. Negative indexes will start from the last. Thus index of -1 means the last character of the string and -2 is the second to last and so on.

```
In [28]: a = "this is a string"
         print("the last char is:", a[-1])
         print("the second to last char is:", a[-2])
```

```
the last char is: g
the second to last char is: n
```

Figure 1 is taken from Google developers website and gives a good description about that kind of indexes. I just copy and paste the image.

- **String slicing** String slicing is the python method to retrieve a substring of a given string. As a casual indexing, we can use square brackets to slice a string as follows:

str.[start: stop: step]
start: the position of the first character to retrieve.
stop: the position of the last character to retrieve.
step: how many steps to take.

```
In [40]: print('a is:',a)

         print('starts from the third char to seventh char.')
         print(a[2:6])

         print('starts from the second char to seventh char stepping two chars at a time.')
         print(a[2:6:2])

         #this will return the whole string :)
         print("the whole 'a' string is:", a[0:len(a)])
```

```
a is: this is a string
starts from the third char to seventh char.
is i
starts from the second char to seventh char stepping two chars at a time.
i
the whole 'a' string is: this is a string
```

- **More on slicing**

Many things we can do with strings. - we can leave indexes empty so that python can use the default values. (i.e [: :]). - defalut value for **start** is 0. - default value for **stop** is *len()*. - default value for **step** is 1. - negative indexes are allowed. - [: : -1] will flip the defaults to: [-1: -(len()+1):-1]. Simply, it will reverse the string because it will start from the last element up to the first one. - If negative numbers are used in the **step**, then **start** must be greater than stop and the other way around for positive numbers.

```
In [43]: print("this is 'a' string:", a)

         # using default values will return the whole string:
         print(a[::])

         # reverse the string:
         print(a[::-1])
```

```
this is 'a' string: this is a string
this is a string
gnirts a si siht
```

- **Split**

You can split a given string into a list of strings based on a given delimiter. Suppose that I want to get all the words separated by a space from a string, then this function is useful. Do not worry about the word `list` will talk about it later.

```
In [118]: a = "This is a string!!"
          # if there is nothing passed to the funtion,
          #then the default is the white sapce.
          print(a.split())
```

```
['This', 'is', 'a', 'string!!']
```

- **join**

Join() function is the opposite of the split function. It takes a list of strings and add them up to one string. We can use a given string as a connector. Look at the following code:

```
In [122]: a="this is"
          b="a string"
          # join the two strings with comma in between
          print(", ".join([a,b]))
```

```
this is, a string
```

More python string functions. This section, will talk about general strings functions. The description of each method is given within the code block in the following coding paragraph.

```
In [89]: a = "This is a string"

         # the following functions are self-descriptive functions.
         a = a.upper()
         print("1:",a)

         a = a.lower()
         print("2:",a)


         # the following functions will test the given string according to their namings.

         #This function test whether a string is all alphabets.
         #Spaces and !! will cause the function to return False.
         print("3:",a.isalpha())

         #By the way, You can define a string on the fly and test it against this function.
         #Look at this code:
         print("4:","thisstringisallalphabetics".isalpha())

         #tests for digits:
```

```
print("5:",a.isdigit())
print("6:","123456789".isdigit())

#tests for space:
print("7:",a.isspace())
print("8:","  ".isspace())


# this function will search a string for a given substring.
# it returns the index of teh first occurance.
print("9:",a.find("is"))

# this function will replace a given substring with another substring.

a = a.replace("string","nicer string!!")
print("10:",a)

#
```

```
1: THIS IS A STRING
2: this is a string
3: False
4: True
5: False
6: True
7: False
8: True
9: 2
10: this is a nicer string!!
```

And by this ends the String section.

0.2 Python Operators

Operators are special characters with specific functionalities. There are different types of operands. The following lists will show many operands of different kinds.

I just copied and pasted these lists from this good sources: https://www.geeksforgeeks.org/basic-operators-python/

Operator Description Syntax ________

```
- Arithmetic Operators.
```

- [+] Addition: adds two operands x + y
- [-] Subtraction: subtracts two operands x - y
- [*] Multiplication: multiplies two operands x * y
- [/] Division (float): divides the first operand by the second x / y
-

0.3 [//] Division (floor): divides the first operand by the second x // y

- Relational Operators.

- [>] Greater than: True if left operand is greater than the right x > y
- [<] Less than: True if left operand is less than the right x < y
- [==] Equal to: True if both operands are equal x == y
- [!=] Not equal to - True if operands are not equal x != y
- [>=] Greater than or equal to: True if left operand is greater than or equal to the right x >= y
-

0.4 [<=] Less than or equal to: True if left operand is less than or equal to the right x <= y

- Logical Operators

- and Logical AND: True if both the operands are true x and y
- or Logical OR: True if either of the operands is true x or y
-

0.5 not Logical NOT: True if operand is false not x

- Assignment Operators

- = Assign value of right side of expression to left side operand x = y + z
- += Add AND: Add right side operand with left side operand and then assign to left operand a+=b a=a+b
- -= Subtract AND: Subtract right operand from left operand and then assign to left operand a-=b a=a-b
- = *Multiply AND: Multiply right operand with left operand and then assign to left operand a*=b a=a*b
- /= Divide AND: Divide left operand with right operand and then assign to left operand a/=b a=a/b
- %= Modulus AND: Takes modulus using left and right operands and assign result to left operand a%=b a=a%b
- //= Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand a//=b a=a//b
- **= Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand

```
In [130]: # Examples of Arithmetic Operator
          a = 9
          b = 4

          # Addition of numbers
          add = a + b
          # Subtraction of numbers
          sub = a - b
          # Multiplication of number
          mul = a * b
          # Division(float) of number
          div1 = a / b
          # Division(floor) of number
          div2 = a // b
          # Modulo of both number
          mod = a % b

          # print results
          print(add)
          print(sub)
          print(mul)
          print(div1)
          print(div2)
          print(mod)


          # Examples of Relational Operators
          a = 13
          b = 33

          # a > b is False
          print(a > b)

          # a < b is True
          print(a < b)

          # a == b is False
          print(a == b)

          # a != b is True
          print(a != b)

          # a >= b is False
          print(a >= b)

          # a <= b is True
          print(a <= b)
          # Examples of Logical Operator
          a = True
          b = False

          # Print a and b is False
          print(a and b)

          # Print a or b is True
          print(a or b)

          # Print not a is False
          print(not a)
```

```
13
5
36
2.25
2
1
False
True
False
True
False
True
False
True
False
```

0.6 Python Data Structures

The previous data types are so useful. They are used to hold the data we need to process. However, they are not enough. What about having a dozen of data? defining variables for each is redundant or even ridiculous! That is why python, as many other languages, provides data structures.

0.6.1 Lists

Lists are one of the simplest data types. You can think of it as a sequence of elements. In that context, we can say that strings are special type of lists as they are just a sequence of characters. In fact, both of them "strings and lists" shares many properties and functions. **keep in mind that lists should contain elements of the same type!!**

Playing with lists.

```
In [191]: # to define a new list:
          mylist = []
```

```
# you can also say:
mylist2 = [1,2,3,4,5,6,7,8,9]

# add item to a list:
mylist.append(0)
print("1: mylilst:",mylist)

#add list to list:
mylist.extend(mylist2)
print("2: mylist:", mylist)
# you can also use this shortcut:
#mylist += mylist2

#you can insert items to any index:
mylist.insert(7,7)
print("3: mylist:", mylist)

#Remove the first item from the list whose value is x.
mylist.remove(7)
print("4: mylist",mylist)
#It is an error if there is no such item.
# the following line will produce an error:
#mylist.remove(10)

# or you can delete element at a given index:
del mylist[0]
print("5: mylist:",mylist)

# you can use pop() function to do the same thing.
# However, passing no index to pop() will delete the last element of the array.
mylist.pop()
print("6: mylist:", mylist)
mylist.pop(0)
print("7: mylist:",mylist)


# to clear the list "remove all elements at once":
mylist.clear()
print("8: mylist:",mylist)
# let us return our list back for further manipulations:
mylist = [0,1,2,3,4,5,6,7,8,9]

# as we have done with string, we can index lists as well.
# this will retrive elements in the odd indexes:
sublist = mylist[1::2]
print("9: sublist:",sublist)
#negative indexes applies to lists as well. BUT REVERSED:
sublist = mylist[5:0:-1]
print("10: the first five elements reversed:",sublist)

# you can reverse the list as follows:
mylist.reverse()
print("11: mylist:", mylist)

#you can sort the list as follows:
mylist.sort()
print("12: mylist:", mylist)
# you can also sort with reverse as follows:
#mylist.sort(reverse=True)

# you can get the number an elements is repeated in a list:
print("13: number of 5's:",mylist.count(5))

# len() function can be used to get the number of elements:
print("14: number of elements in mylist:",len(mylist))

# you can retrive the index of any element:
print("15: index of 5:", mylist.index(5))

#--------------------------------------------------------------
# The following methods are only applicable for numeric types:
#--------------------------------------------------------------
# sum a list:
print("16: sum of mylist elements:",sum(mylist))

# max of list:
print("17: max of mylist:", max(mylist))

# min of list:
print("18: min of mylist:", min(mylist))
```

```
1: mylilst: [0]
2: mylist: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
3: mylist: [0, 1, 2, 3, 4, 5, 6, 7, 7, 8, 9]
4: mylist [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
5: mylist: [1, 2, 3, 4, 5, 6, 7, 8, 9]
6: mylist: [1, 2, 3, 4, 5, 6, 7, 8]
7: mylist: [2, 3, 4, 5, 6, 7, 8]
8: mylist: []
9: sublist: [1, 3, 5, 7, 9]
10: the first five elements reversed: [5, 4, 3, 2, 1]
11: mylist: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
12: mylist: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
13: number of 5's: 1
14: number of elements in mylist: 10
```

```
15: index of 5: 5
16: sum of mylist elements: 45
17: max of mylist: 9
18: min of mylist: 0
```

```
In [192]: # range function usage
          # get list from 0,10
          print(range(10))
```

```
range(0, 10)
```

0.6.2 Dictionaries

It is a useful data structure. The different between lists and dictionaries is that lists are indexed by integers while dictionaries are indexed by sort of keys!

Playing with dictionaries

```
In [193]: # initialize a dictionary:
          dict = {'name':'KFUPM', 'age':55}
          print("1: dict:",dict)

          # get only keys:
          print("2: dict keys:",dict.keys())

          # get only values:
          print("3: dict values:", dict.values())

          # update dict:
          dict["name"]= "King Fahd University for Petroleum and Minerals"
          print("4:",dict)

          # add new key-value element:
          dict["short name"]= "KFUPM"
          print("5:",dict)

          # delete a key:
          dict.pop('age',None)
          print("6:",dict)
```

```
1: dict: {'name': 'KFUPM', 'age': 55}
2: dict keys: dict_keys(['name', 'age'])
3: dict values: dict_values(['KFUPM', 55])
4: {'name': 'King Fahd University for Petroleum and Minerals', 'age': 55}
5: {'name': 'King Fahd University for Petroleum and Minerals', 'age': 55, 'short name': 'KFUPM
6: {'name': 'King Fahd University for Petroleum and Minerals', 'short name': 'KFUPM'}
```

0.6.3 accessing list elements

You can access list elements using loops. We will talk about loops later in details. However, we will use it here to access list elements.

```
In [194]: #you can access list elements as follows:

          print("list elements are:")
          for elem in mylist:
              print(elem, end=", ")
          print()
```

```
list elements are:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
```

```
In [197]: # also, you can access list elements by index.
          # we will use range() function to generage iterable integers.
          # its usage is similar to list indexes.
          # range(start, stop, step) with the same default values of list inexing.
          print("elements of range() function:")
          for i in range(10):
              # let us print numbers of range function:
              print(i, end=", ")
          print()
          # now, we can access list elements as follows:
          # multiply each list element by 2:
          print("doubled lists elements:")
          for i in range(len(mylist)):
              mylist[i] *= 2


          print(mylist)
```

```
elements of range() function:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
doubled lists elements:
[0, 8, 16, 24, 32, 40, 48, 56, 64, 72]
```

0.7 Exercises

```
In [201]: # use the following line to ask some input from the user.

          x = input("enter some input: ")
          print("you entered:",x)
```

```
enter some input: Hello world of python! :)
you entered: Hello world of python! :)
```

```
In [ ]: # Write a Python program to remove duplicates from a list.









        # Write a Python program to count the number of characters (character frequency)
        #in a string. and write them to a dictionary
        #Sample String : google.com'
        #Expected Result : {'o': 3, 'g': 2, '.': 1, 'e': 1, 'l': 1, 'm': 1, 'c': 1}









        #Write a Python program to get a single string from two given strings.
        #separated by a space and swap the first two characters of each string!











        # Write a Python program to sum all the items in a list
        # DO NOT USE THE SUM() FUNCTION BUT USE IT TO COMPARE WITH YOUR RESUTS
```

0.8 Resources

1- https://www.pythoncentral.io/pythons-range-function-explained/
2- https://docs.python.org/3/tutorial/datastructures.html
3- https://www.tutorialspoint.com/python/python_loops.htm

4- https://developers.google.com/edu/python/strings