Python fundamentals day 15

Today's Agenda

- Strings
- Handling a string
- Representation of string

Strings

We have brief knowledge about strings from datatypes section. Let us begin with knowing what are the different ways of creating strings in python

```
s1= ''  # an empty string
print(s1)
s2= 'P'  # single character string
print(s2)
s3= 'Python' # combination of characters
print(s3)
s4= '''C
Java
Python'''  # multiline string
print(s4)
s5= str(99.9) # a real number converted to string
print(s5)
```

For creating string both ' ' and " " can be used. But for creating a multiline string its mandatory to use ''' ''' (triple quotes).

Output:

```
In [2]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')

P
Python
C
Java
Python
99.9
```

Handling a string

We know that a string can be created using ' ' or " ". But what if we want to print a statement like Practice makes "Perfect" on the output screen. Let's see what are the different ways of handling such strings in python

```
s='Practice makes perfect'
print(s)
```

Output:

```
In [3]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Practice makes perfect
```

Great!! But what if we want perfect within quotes?? Let's see further

```
s='Practice makes 'perfect''
print(s)
```

This will certainly give an error. We can predict by noticing the colour change in the string

Considered as a string

'Practice makes 'perfect' ' //perfect is unknown entity for compiler

Considered as a string

Output:

```
  File "C:\Users\rooman\Anaconda3\lib\site-
packages\spyder_kernels\customize
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'),
namespace)

  File "C:/Users/rooman/OneDrive/Desktop/
python/test.py", line 1
    s='Practice makes 'perfect''
                       ^
SyntaxError: invalid syntax
```

Let us see what are the different ways to debug this

- **Using escape character (\)**

```
s='Practice makes \'perfect\'' # \ is the escape character
print(s)
```

Output:

```
In [5]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Practice makes 'perfect'
```

Anything that is after the \ is considered to be a part of string.

- **Using quotes**

```
s='Practice makes "perfect"'
print(s)
```

Output:

```
In [6]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Practice makes "perfect"
```

If the string is enclosed within ' ' then we can use " " inside the string and vice versa.

```
s="Practice makes 'perfect'"
print(s)
```

Output:

```
In [7]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Practice makes 'perfect'
```

Now that we know how to print a sentence with characters enclosed in ' ' and " ". Let us see what if we want to print "Practice" makes 'Perfect'. Notice that here we are using both ' ' and " " in one sentence itself

By now we can guess that we certainly cannot enclose the above sentence in ' ' or " ", because that might give an error. So let's try with ''' ''' and see what happens

```
s=''''"Practice" makes 'perfect''''
print(s)
```

Output:

```
  File "C:\Users\rooman\Anaconda3\lib\site-
packages\spyder_kernels\customize
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'),
namespace)

  File "C:/Users/rooman/OneDrive/Desktop/
python/test.py", line 1
    s=''''"Practice" makes 'perfect''''
                                      ^
SyntaxError: EOL while scanning string literal
```

The error basically means that compiler doesn't know if the string is ending with pair of double quotes ("") or triple quote and a single quote (''' ') or single quote and a triple quote (' ''') . This can be resolved by giving proper space to separate one from each other

```
s=''''"Practice" makes 'perfect' '''
print(s)
```

Output:

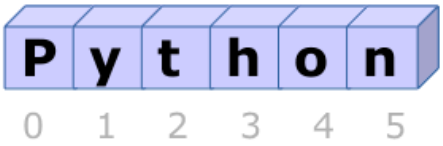
```
In [9]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
"Practice" makes 'perfect'
```

In general outer quotes and inner quotes must not be same.

Representation of string

Let us now see how strings are represented in python

```
s="Python"
print(s)
```



Output:

```
In [10]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Python
```

When a string is created, a string type object is created on memory. As string is a sequence type object, which means it is ordered and has positional values or index values. When we tell print(s) entire string gets printed. But as string is collection of characters we can also access one character at a time. Let us see how

```
s="Python"
print(s)
print(s[4]) #accessing single character
```

Output:

```
In [11]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Python
o
```

A string is also an iterable in python. Recollect the for loop session, where anything on which a for loop can be applied on is called as an iterable.

```
s="Python"
print(s)
print(s[4])

for i in s:  #passing string s as iterable
    print(i)
```

Output:

```
In [12]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Python
o
P
y
t
h
o
n
```

If you recollect datatypes session, you know that in python strings are immutable. Let us verify this

```
s="Python"
print(s[4])
s[4]="i"
print(s[4])
```

Here we are trying to change the 4th position of string Python which is o to i

Output:

```
  File "C:\Users\rooman\Anaconda3\lib\site-
packages\spyder_kernels\customize
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'),
namespace)

  File "C:/Users/rooman/OneDrive/Desktop/
python/test.py", line 3, in <module>
    s[4]="i"

TypeError: 'str' object does not support item
assignment
```

The above error basically tells us that, in python string values cannot be changed or strings are immutable.

Now let us see some interesting things that strings exhibit

```
s1="hello"
s2="world"
print(s1)
print(s2)
print(id(s1)) #checking the address of s1 object
print(id(s2)) #checking the address of s2 object
print(s1[4])
print(s2[1])
```

Output:

```
In [14]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
hello
world
2429892154032
2429825298736
o
o
```

We see that both the objects s1 and s2 are stored in different addresses. Now, both the strings have l and o in common. Let us see at what address o is stored in both the strings

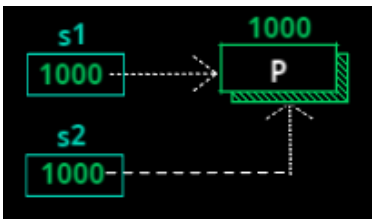
```
s1="hello"
s2="world"
print(s1)
print(s2)
print(id(s1)) #checking the address of s1 object
print(id(s2)) #checking the address of s2 object
print(s1[4])
print(s2[1])
print(id(s1[4])) # address of o in hello
print(id(s2[1])) #address of o in world
```

Output:

```
In [15]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
hello
world
2429892154032
2429825298736
o
o
2429794338736
2429794338736
```

Interestingly we see that both the o's are stored in same address. Before knowing how let us take a simple example and understand

```
s1="P"
s2="P"
print(s1)
print(s2)
print(id(s1))
print(id(s2))
```



Output:

```
In [17]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test1.py', wdir='C:/Users/
rooman/OneDrive/Desktop/python')
P
P
2429796067568
2429796067568
```

As strings are immutable, if the values are same then the address will also be same. Keeping this in back of your mind let us go back to the previous example and know more

Python internally maintains a dictionary, the name of this dictionary is interned dictionary. All the single characters are stored as keys and address of those characters are stored as values.

Let us see how exactly that happens

```
s1="hello"
s2="world"
print(s1,s2)
print(id(s1),id(s2)) #address of strings
print(s1[2],s1[3],s2[3]) #all the l's
print(id(s1[2]),id(s1[3]),id(s2[3])) # address of all l's
print(s1[4],s2[1]) #all the o's
print(id(s1[4]),id(s2[1])) #address of all o's
```

Output:

```
In [20]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
hello world
2429891670576 2429825298736
l l l
2429795759920 2429795759920 2429795759920
o o
2429794338736 2429794338736
```

Pictorial representation of string in dictionary format: