Day18_String_methods

9. setdefault(key, default=None)

Insert key with a value of default if key is not in the dictionary.
Return the value for key if key is in the dictionary, else default.

```
print(dct)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik'}
dct.setdefault(5,"Covishield")
print(dct)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik',
  5: 'Covishield'}

print(dct.setdefault(2,"Covaxin"))
# moderna
print(dct)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik',
# 5: 'Covishield'}

print(dct.setdefault(3))
#johnson and johnson

print(dct.setdefault(6))
#None
print(dct)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik',
# 5: 'Covishield', 6: None}
```

10 update() :updates the dictionary with the given dictionary object

```
print(dct)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik', 5: 'Covishield', 6: None}

dct.update({7:'covaxin'})
print(dct)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik',
# 5: 'Covishield', 6: None, 7: 'covaxin'}

dct1 = {8:"Eight",9:"Nine"}
dct.update(dct1)
print(dct)
```

```
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik',
# 5: 'Covishield', 6: None, 7: 'covaxin', 8: 'Eight', 9: 'Nine'}
```

11. fromkeys(iterable, value=None): Create a new dictionary with keys from iterable and values set to value.

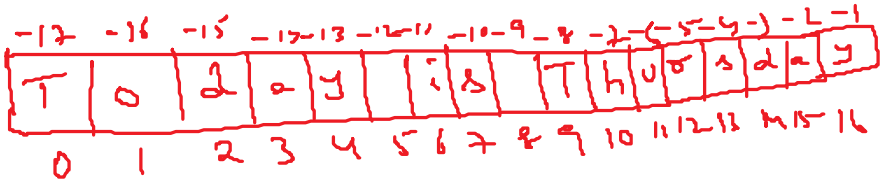
```
class Emp:
    def set(self):
        pass
    def get():
        pass
e1 = Emp()
e1.set()
Emp.get()
```

```
dct2 = dict.fromkeys([10,20,30,40])
print(dct2)
#{10: None, 20: None, 30: None, 40: None}

dct2 = dict.fromkeys([10,20,30,40],'Numbers')
print(dct2)
#{10: 'Numbers', 20: 'Numbers', 30: 'Numbers', 40: 'Numbers'}
```

String methods

```
s1 = "Today is Thursday"
```




```
s1[1:8]
#'oday is'

s1[3]
#'a'

#Repeating the strings
a = 3
print(a*3)
s1 = "Python"
print(s1*3)
#PythonPythonPython
```

```
#concatenation
print(s1+s1+s1)
#PythonPythonPython

#Checking membership
s1 = "Today is Thursday"
s2 = "Thursday"
s3 = "wednesday"

if s2 in s1:
   print(s2," is a part of ",s1)
else:
   print(s2," is not a part of ",s1)
#Thursday  is a part of  Today is Thursday

if s3 in s1:
   print(s3," is a part of ",s1)
else:
   print(s3," is not a part of ",s1)
#wednesday  is not a part of  Today is Thursday

#compare the string

s2 = "Thursday"
s3 = "wednesday"

if s2==s3:
   print("s1,s2 are same")
else:
   print("s1,s2 are not same")

#s1,s2 are not same


s2 = "wednesday "
s3 = "wednesday"
if s2==s3:
   print("s1,s2 are same")
else:
   print("s1,s2 are not same")

#s1,s2 are not same
```

#strip() functions : These are used for stripping the unneccessary
#spaces from the string

strip(): takes off the spaces from both sides
lstrip(): takes off the space from left
rstrip(): takes off the space from right

```
s4 = "  Weekend  "
print(s4)
print(s4.strip())
print(s4)
```

```
  Weekend
Weekend
  Weekend
```

```
s4 = "  Weekend  "
print(s4)
print(s4.lstrip())
print(s4)
```

```
  Weekend
Weekend  #right space still exists
  Weekend
```

```
 s4 = "  Weekend  "
print(s4)
print(s4.rstrip())
print(s4)
```

```
  Weekend
  Weekend#left still exist
  Weekend
```

```
s2 = "wednesday "
s3 = " wednesday"
if s2.strip()==s3.strip():
   print("s1,s2 are same")
else:
```

```
   print("s1,s2 are not same")
#s1,s2 are same


#Finding the substrings using find and index

find(substring,beginindex,endindex)

s1 = "Weekend is a long way away, always weekend is good"
help(s1.find)

find(...) method of builtins.str instance
   S.find(sub[, start[, end]]) -> int

   Return the lowest index in S where substring sub is found,
   such that sub is contained within S[start:end].  Optional
   arguments start and end are interpreted as in slice notation.

   Return -1 on failure.

s1.find("way")
#18

s1.find("way",19)
#23


s1.find("way",24,len(s1))
#30

s1.find("friday")
#-1

#Retrieve all the index locations of 'way' in s1

#index()
help(s1.index)

index(...) method of builtins.str instance
   S.index(sub[, start[, end]]) -> int

   Return the lowest index in S where substring sub is found,
   such that sub is contained within S[start:end].  Optional
   arguments start and end are interpreted as in slice notation.
```

Raises ValueError when the substring is not found.

```
s1 = "Weekend is a long way away, always weekend is good"
s1.index("way")
#18

s1.index("way",19)
#23


s1.index("way",24,len(s1))
#30

s1.index("friday")
#ValueError: substring not found


#replacing the string

replace(old,new)

s1 = "Weekend is a long way away, always weekend is good"
print(s1)
print(s1.replace("is a","is not a"))
print(s1)

Weekend is a long way away, always weekend is good
Weekend is not a long way away, always weekend is good
Weekend is a long way away, always weekend is good
```