PPiC 3.1

```
>>> myname = "Danning Du"
```

Here the myname declaration initializes a variable and assigns it my entire name "Danning" (First) and "Du" (last). I do not have a middle name. That variable becomes of string type because of it's assigned to a string.

PPiC 3.4 To combine the concatenation operator and slice operator, we first see how can we print me name in the form using each operator.

Using concatenation operators: addition operator (+)

```
>>> lname = "Du"
>>> fname = "Danning"
>>> fullname = lname + ", " + fname
>>> fullname
'Du, Danning'
```

Using slice operator [:]

```
>>> fullname = "Du, Danning"
>>> len(fullname)
11
>>> fullname[0:11]
'Du, Danning'
```

```
>>> # Combine the addition operator and slice operator
>>> lname = "Du"
>>> fname = "Danning"
>>> len(lname)
2
>>> len(fname)
7
>>> fullname = lname[0:2] + ", " + fname[0:7] + "."
>>> print(fullname)
Du, Danning.
```

PPiC 3.6 To write the string mississippi using concatenation and repetition,

```
>>> s='s'
>>> p='p'
>>> 'mi' + s*2 + 'i' + s*2 + 'i' + p*2 + 'i'
'mississippi'
```

s*2 using repetition operator * to repeat s twice, the same for p
Use addition operator (+) to put all strings together.

PPiC 3.8 To find the number of occurrences of the character ‘s’ in the string 'mississippi' using the count method, we use: `astring.count(item)`

```
>>> 'mississippi'.count('s')
4
```

PPiC 3.9 To replace all occurrences of the substring `'iss'` with `'ox'` in `'mississippi'`, we use `astring.replace(old,new)` to replace all occurances of old substring `'iss'` with new substring `'ox'` in `'mississippi'`.

```
>>> 'mississippi'.replace('iss','ox')
'moxoxippi'
```

PPiC 3.11 To make the word `'python'` centered and all capital letters in a string of length 20,
We first use center method to return the string `'python'` surrounded by spaces to make `'python'` 20 characters.
Then, use upper method to return it in all uppercase.

```
>>> 'python'.center(20)
'       python       '
>>> ('python'.center(20)).upper()
'       PYTHON       '
```

PPiC 3.15 To write the indexToLetter function using ord and chr,
We first defines function `indexToLetter` that takes parameter `index`.
As defined by the ASCII, `ord('a')` converts character 'a' to number 97, and `ord('z')` converts character 'z' to number 122. Checks if `index` is in the `range`(97, 123).
`letter` is converted from number to character by using `chr(index)`
If `index` is not in the `range`(97, 123), there is no corresponding character to convert according to the ASCII.
Returns the converted `letter`

```
>>> def indexToLetter(index):
        if index in range(ord('a'), ord('z')+1):
                letter = chr(index)
        else:
                letter = ""
        return letter
```

To test if the function works,

```
>>> indexToLetter(103) # checks if the function indexToLetter(index) works
'g'
>>> indexToLetter(123)
''
>>> indexToLetter(96)
''
```

PPiC 3.18 To write a python function stripSpaces(myString) that takes a string representing a phrase as a parameter and returns the paragraph with the order of the letters intact but the spaces between each word removed,
We first defines function `stripSpaces` that takes parameter `myString`,
Then use the replace method to replace all occurrences of space `" "` with no space `""` in `myString`, and assign it to a variable called `newString`.
Returns `newString` with the order of the letters in `myString` intact but no spaces between each word.

```
>>> def stripSpaces(myString):
        newString = myString.replace(" ","")
        return newString
```

To test if the function works,

```
>>> stripSpaces("be right back")
'berightback'
```

PPiC 3.21 **To write the substitutionDecrypt method,**
We first define function substitutionDecrypt that takes parameters cipherText and key.
Then create an `alphabet` string. There is space at the end of the `alphabet` and the `key` so we can preserve the spaces between the individual words in our secret message.
Converts `cipherText` to all lowercases using lower method
For loops on each character `ch` in `cipherText`, using find method to return the index of the first occurrence of character in string key, assign the index to variable `idx`.
`alphabet[idx]` returns the character at position `idx` and assign it to `plainText` for all the characters in `cipherText`
Return the decrypted cipherText as `plainText`

```
>>> def substitutionDecrypt(cipherText ,key):
        alphabet = "abcdefghijklmnopqrstuvwxyz "
        cipherText = cipherText.lower()
        plainText = ""
        for ch in cipherText:
                idx = key.find(ch)
                plainText = plainText + alphabet[idx]
        return plainText
```

To test if the function works, we write a string for parameter key. We decrypt two cipherText using two different keys.

```
>>> testKey1 = "zyxwvutsrqponmlkjihgfedcba "
>>> testKey2 = "ouwckbjmpzyexavrltsfgdqihn "
>>> substitutionDecrypt("gsv jfrxp yildm ulc",testKey1)
'the quick brown fox'
>>> substitutionDecrypt("fmk lgpwy utvqa bvi",testKey2)
'the quick brown fox'
>>>
```

PPiC 3.23 To write the removeChar function using for loops rather than slice operators,

We first define function `removeChar` that takes parameter `string` and `idx`.
Then create a variable `newStr`.
For loops on `i` from 0 to the length of the string, checks if `i` is not equal to the parameter `idx`.
For every `i` not equal to `idx`, `newStr` added up the character at position `i` in string
Returns the `newStr` after remove `string[idx]` from string

```
>>> def removeChar(string,idx):
        newStr = ""
        for i in range(0,len(string)):
                if i != idx:
                        newStr = newStr + string[i]
        return newStr
```

To test if the function works,

```
>>> removeChar("Hello world",5)
'Helloworld'
>>> removeChar("Pineapple",7)
'Pineappe'
```