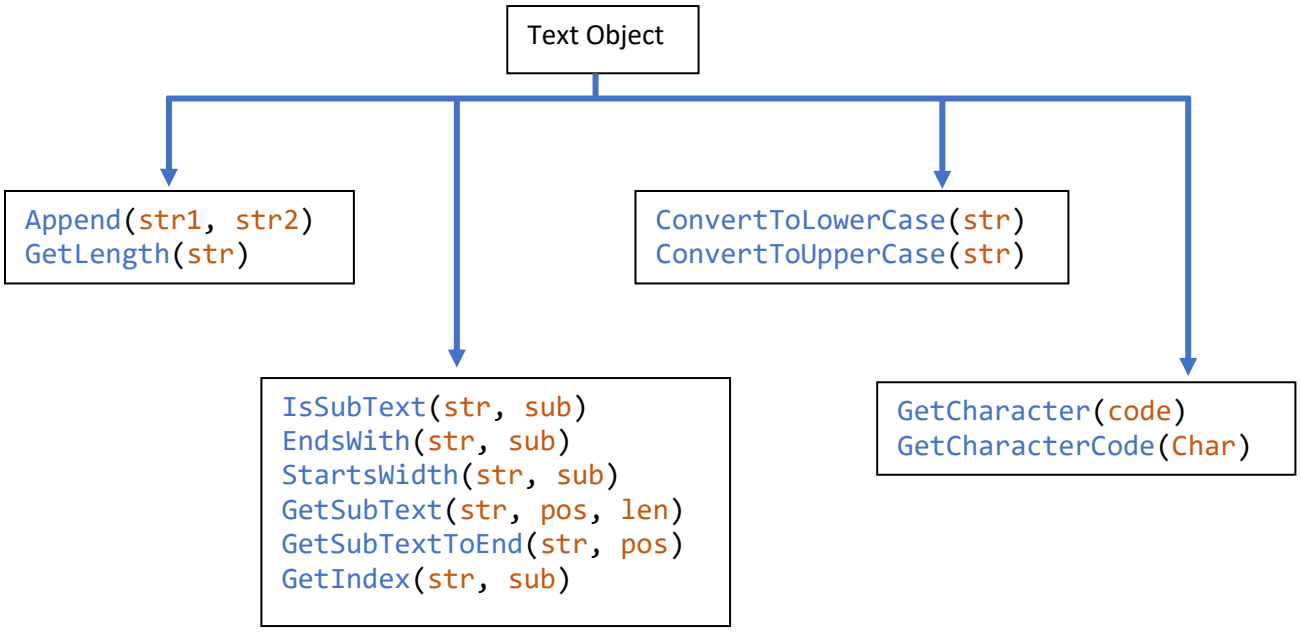
Het Tekst Object

Je hebt al eerder met *strings* (tekenreeksen) gewerkt. Om samen te vatten: een string is een reeks tekens die tussen dubbele aanhalingstekens staat, zoals "stringY sgrinGy striNg stRing". Deze tekens kunnen letters (zowel hoofdletters als kleine letters), cijfers (0 tot 9) en andere symbolen op een toetsenbord bevatten (zoals +, -, &, @ enzovoort). Je kunt strings in je programma's gebruiken om namen, adressen, telefoonnummers, boektitels, namen van Star Trek-afleveringen en meer op te slaan. Het **Text**-object in Small Basic bevat veel handige methoden om met tekenreeksen te werken. De onderstaande afbeelding toont de volledige lijst van de methoden van het tekstobject. De methoden zijn onderverdeeld in vier groepen die we in de volgende paragrafen zullen bespreken.



Strings Samenvoegen en hun Lengte bepalen

Het combineren van strings en het vinden van hun lengte is een veelvoorkomende taak bij het programmeren. In het Engels worden hiervoor vaak de woorden *concatenate* of *append* gebruikt.

De methode `Append()` kan twee tekenreeksen samenvoegen, zoals weergegeven in het volgende codevoorbeeld:

```
1 str = Text.Append("He-", "Man")
2 TextWindow.WriteLine(str)          'Toont: He-Man
```

Eerder heb je geleerd hoe je strings samenvoegt met het +-teken. Maar de methode `Append()` is handig als u tekst hebt die door het +-teken als getallen wordt behandeld, zoals in het volgende voorbeeld:

```
1 res = Text.Append("1", "5")
2 TextWindow.WriteLine(res)          'Uitvoer: 15 (1 gevolgd door 5)
3 TextWindow.WriteLine("1" + "5")    'Uitvoer: 6
```

De eerste instructie voegt de twee strings ("1" en "5") toe en wijst het resultaat toe aan de variabele `res` (afkorting van resultaat). De uitvoer van de tweede instructie laat zien dat de tekenreeks "5" is toegevoegd aan de tekenreeks "1", wat resulteert in een nieuwe tekenreeks "15". De derde verklaring laat zien dat u deze aaneenschakeling niet kunt doen met het + -teken. De operator + interpreteert zijn twee operatanden als getallen (1 en 5) en telt deze getallen bij elkaar op, daarom geeft de derde instructie 6 weer.
Het gebruik van `Append()` is de enige manier om getallen samen te voegen, *concatenate*, in Small Basic.

De lengte van een string bepalen

Het aantal tekens in een string bepaalt de lengte. Om de lengte van een tekenreeks te vinden, kunt u de methode `GetLength()` gebruiken, zoals in het volgende voorbeeld:

```
1 res = Text.GetLength("")                  'res = 0 (lege string)
2 TextWindow.WriteLine(res)
3 res = Text.GetLength("Colegio Arubano")   'res = 15
4 TextWindow.WriteLine(res)
5 res = Text.GetLength("2023")              'res = 4
6 TextWindow.WriteLine(res)
7 res = Text.GetLength(-107.5)              'res = 6
8 TextWindow.WriteLine(res)
```

`GetLength()` behandelt zijn argumenten als een tekenreeks en retourneert het aantal tekens in die tekenreeks. Regel 1 laat zien dat een lege string nul lengte heeft. Regel 3 laat zien dat de lengte van de string "`Colegio Arubano`" 15 is, omdat deze string 15 karakters bevat (spaties zijn ook karakters). Regel 5 roept `GetLength()` aan met het getal `2023` als argument. `GetLength()` behandelt dit getal als een tekenreeks ("`2023`") en retourneert `4` als de lengte van deze tekenreeks. Een soortgelijk proces vindt plaats in regel 7 voor het getal `-107,5`, waarbij `GetLength()` `6` retourneert (vier cijfers, het minteken en de komma).

Strings uit elkaar halen: substrings

Net zoals je strings kunt samenvoegen om langere te maken, kun je strings ook scheiden in kleinere strings, die *substrings* worden genoemd. Een substring is slechts een deel van een grotere string. Het tekstobject heeft zes methoden waarmee u met substrings kunt werken.

1 De methode IsSubText()

Je kunt `IsSubText()` gebruiken om erachter te komen of de ene tekenreeks deel uitmaakt van een andere. Deze methode heeft twee argumenten: de string die je wilt doorzoeken en de substring die je wilt zoeken. Het retourneert "`True`" of "`False`", afhankelijk van of de substing in de bronreeks staat. Hier zijn enkele voorbeelden:

```
1 myString = "The quick brown fox"
2 res = Text.IsSubText(myString, "brown")    'res = "True"
3 TextWindow.WriteLine("1 is " + res)
4 res = Text.IsSubText(myString, "BROWN")    'res = "False"
5 TextWindow.WriteLine("2 is " + res)
6 res = Text.IsSubText(myString, "DOG")      'res = "False"
7 TextWindow.WriteLine("3 is " + res)
```

Zoals deze voorbeelden laten zien, is `IsSubText()` hoofdlettergevoelig bij het zoeken naar subtekenreeksen. Dit is de reden waarom zoeken naar "BROWN" in regel 4 "`False`" oplevert.

2 De EndsWith()-methode

Gebruik `EndsWith()` om erachter te komen of een string eindigt met een bepaalde substring. Hier zijn enkele voorbeelden:

```
1 myString = "The quick brown fox"
2 res = Text.EndsWith(myString, "fox")    'res = "True"
3 TextWindow.WriteLine("1 is " + res)
4 res = Text.EndsWith(myString, "x")      'res = "True"
5 TextWindow.WriteLine("2 is " + res)
6 res = Text.EndsWith(myString, "FOX")    'res = "False"
7 TextWindow.WriteLine("3 is " + res)
8 res = Text.EndsWith(myString, "dog")    'res = "False"
9 TextWindow.WriteLine("4 is " + res)
```

Nogmaals, het geval van de tekenreeks is van belang: de zoekopdracht naar "`FOX`" in regel 6 retourneert "`False`".

3 De StartsWith()-methode

Gebruik `StartsWith()` om uit te zoeken of een string begint met een bepaalde substring. Hier zijn enkele voorbeelden:

```
1 myString = "The quick brown fox"
2 res = Text.StartsWith(myString, "The")    'res = "True"
3 TextWindow.WriteLine("1 is " + res)
4 res = Text.StartsWith(myString, "T")      'res = "True"
5 TextWindow.WriteLine("2 is " + res)
6 res = Text.StartsWith(myString, "the")    'res = "False
7 TextWindow.WriteLine("3 is " + res)
```

Op dezelfde manier levert de zoekopdracht naar "`the`" in regel 6 "`False`" op.

4 De GetSubText()-methode

Om tekst van elke positie in een tekenreeks te extraheren, kunt u `GetSubText()` gebruiken. Voor deze methode zijn drie argumenten nodig: de bron string om je substring vandaan te halen, de startpositie als de substring en de lengte van de gewenste substring. Bekijk de onderstaande afbeelding om te begrijpen hoe deze methode werkt.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
T	h	e		q	u	i	c	k		b	r	o	w	n		f	o	x

Het eerste teken heeft positie 1, het tweede teken heeft positie 2, enzovoort. Beschouw nu de volgende voorbeelden:

```
1 myString = "The quick brown fox"
2 res = Text.GetSubText(myString, 1, 3)    'res = "The"
3 TextWindow.WriteLine("1 is " + res)
4 res = Text.GetSubText(myString, 0, 3)    'res = ""
5 TextWindow.WriteLine("2 is " + res)
6 res = Text.GetSubText(myString, 17, 3)    'res = "fox"
7 TextWindow.WriteLine("3 is " + res)
8 res = Text.GetSubText(myString, 17, 4)    'res = "fox"
9 TextWindow.WriteLine("4 is " + res)
```

Regel 2 krijgt een substring van lengte 3 vanaf positie 1, die de string "`The`" retourneert. Regel 4 krijgt geen substring die begint op positie 0, omdat de eerste geldige positie 1 is. In plaats daarvan wordt een lege string geretourneerd. Regel 6 krijgt de drieletterige substring die begint op positie

17, die "fox" retourneert. Regel 8 vraagt om een substring met lengte 4 vanaf positie 17. Omdat die substring verder reikt dan het einde van de string, wordt de lengte ingekort en retourneert de methode "fox", waarvan de lengte 3 is.

U kunt GetSubText() in een For-lus gebruiken om toegang te krijgen tot de afzonderlijke tekens van een tekenreeks. De volgende code schrijft bijvoorbeeld elk teken van strIn op een nieuwe regel. Voer de code in en voer deze uit om er zeker van te zijn dat je begrijpt hoe het werkt:

```
strIn = "Informatica is het vak van de toekomst."
For N = 1 To Text.GetLength(strIn)          'Voor elke karakter
  ch = Text.GetSubText(strIn, N, 1)         'Krijgt het karakter op positie N
  TextWindow.WriteLine(ch)                  'Geeft het weer op een nieuwe regel
EndFor
```

Het onderstaande voorbeeld toont de tekst teken voor teken achter elkaar na een kwart seconde. Simuleert oude computers.

```
strIn = "Informatica is het vak van de toekomst."
For N = 1 To Text.GetLength(strIn)          'Voor elke karakter
  ch = Text.GetSubText(strIn, N, 1)         'Krijgt het karakter op positie N
  TextWindow.Write(ch)                      'Geeft het weer op een nieuwe regel
  Program.Delay(250)                        'Pauzeerd uitvoer voor 1/4 sec (1000 milli secnden)
EndFor
```

5 De GetSubTextToEnd()-methode

De methode GetSubTextToEnd() is vergelijkbaar met de GetSubText(), behalve dat het een substring retourneert van de ene positie helemaal naar het einde van de string. Er zijn twee argumenten voor nodig: de bron string waar je de substrings vandaan wilt halen en de startpositie van de substring. Hier zijn enkele voorbeelden:

```
1 myString = "The quick brown fox"
res = Text.GetSubTextToEnd(myString, 13)    'res = "own fox'
TextWindow.WriteLine("1 is " + res)
res = Text.GetSubTextToEnd(myString, 19)    'res = "x"
TextWindow.WriteLine("2 is " + res)
res = Text.GetSubTextToEnd(myString, 20)    'res = ""
TextWindow.WriteLine("3 is " + res)
```

Regel 2 krijgt de substring die begint op positie 17, die "own fox" retourneert. Regel 4 vraagt om de substring vanaf positie 20. Omdat de bron string slechts 19 tekens bevat, retourneert deze methode een lege tekenreeks.

6 De GetIndexOf()-methode

Je geeft de methode `GetIndexOf()` de substring door waarnaar je wilt zoeken, en het retourneert de indexpositie van die substring in de brontekst. Hier zijn enkele voorbeelden:

```
1 myString = "The quick brown fox"
res = Text.GetIndexOf(myString, "The")      'res = 1
TextWindow.WriteLine("1 is " + res)
res = Text.GetIndexOf(myString, "quick")    'res = 5
TextWindow.WriteLine("2 is " + res)
res = Text.GetIndexOf(myString, "QUICK")    'res = 0
TextWindow.WriteLine("3 is " + res)
res = Text.GetIndexOf(myString, "o")        'res = 13
TextWindow.WriteLine("4 is " + res)
res = Text.GetIndexOf(myString, "dog")      'res = 0
TextWindow.WriteLine("1 is " + res)
```

Opdracht 1:

Schrijf een programma dat de karakters van een invoer string in omgekeerde volgorde weergeeft.

Hint: start een lus die telt vanaf de lengte van de tekenreeks tot 1, en gebruik `GetSubText()` om elk teken te extraheren.

Letters of Kleine letters

Soms wilt u strings in hoofdletters of kleine letters weergeven. De methoden `ConvertToLowerCase()` en `ConvertToUpperCase()` kunnen dat voor u doen. Voer de onderstaande code in in Small Basic en voer het programma uit.

```
'Converteer Hoofd-, kleine letters
var1 = "Ewok"
lwrCase = Text.ConvertToLowerCase(var1)    'lwrCase = "ewok"
TextWindow.WriteLine(lwrCase)              'Toont: ewok
TextWindow.WriteLine(var1)                 'Toont: Ewok
uprCase = Text.ConvertToUpperCase(var1)    'lwrCase = "ewok"
TextWindow.WriteLine(uprCase)              'Toont: EWOK
TextWindow.WriteLine(var1)                 'Toont: Ewok
```

Hoofdletter ongevoelige vergelijking:

```
'StringVergelijking
While("True")
  TextWindow.Write("Wie is je favorite Shrek figuur? ")
  name = Text.ConvertToUpperCase(TextWindow.Read())
  If (name = "DONKEY") Then
    TextWindow.WriteLine("Je hebt 200 ogre punten gewonnen!")
  Else
    TextWindow.WriteLine("Je hebt 100 ogre punten gewonnen!")
  EndIf
EndWhile
```

Unicode karakters:

Alle computergegevens worden opgeslagen als binaire reeksen van nullen en enen. De letter A is bijvoorbeeld 0100001. De afbeelding tussen een teken en een binaire weergave wordt codering genoemd. Unicode is een universeel-coderingsschema waarmee u meer dan een miljoen tekens uit vele talen kunt coderen. Elk teken krijgt een uniek nummer toegewezen. Voor het teken A is het codepunt bijvoorbeeld 65.

De methode `GetCharacterCode()` retourneert het codepunt van een teken. Maar de methode `GetCharacter()` doet het tegenovergestelde.

Tik het onderstaande Small Basisprogramma in voer het.

```
'CharCode
str = "ABab12"
For N = 1 To Text.GetLength(str)
  ch = Text.GetSubText(str, N, 1)          'Pakt het Nde karakter
  code = Text.GetCharacterCode(ch)         'Pakt de code
  TextWindow.WriteLine(ch + ": "+ code)    'Toont ch en de code
EndFor
```

Een aanhalingsteken weergeven

Stel dat u de tekenreeks "Eureka" wilt weergeven met dubbele aanhalingstekens in de uitvoer. Als u `TextWindow.WriteLine("Eureka")` schrijft, geeft Small Basic Eureka weer zonder de aanhalingstekens, omdat de aanhalingstekens het begin en einde van een tekenreeks aangeven. Maar Small Basic retourneert een ***syntax error*** als u

TextWindow.WriteLine(""Eureka"") schrijft. Door het codepunt van het aanhalingsteken te gebruiken, kunt u de aanhalingstekens aan de tekenreeks toevoegen, zoals weergegeven in het volgende codefragment:

```
QUO = Text.GetCharacter(34)                'Krijgt het dubbele aanhalingsteken
TextWindow.WriteLine(QUO + "Eureka"+ QUO)  'uitvoer is: "Eureka"
```

Een string met meerdere regels maken

Je kunt een string met meerdere regels maken door het ***line feed*** (codepunt 10) in een string in te sluiten. Voer het volgende codefragment in als voorbeeld:

```
LF = Text.GetCharacter(10)
TextWindow.WriteLine("Regel 1" + LF + "Regel 2")
```

Wanneer je deze code uitvoert, worden de twee strings, "Regel 1" en "Regel 2", weergegeven op twee regels. Het resultaat is identiek aan wat je krijgt als je de volgende twee instructies gebruikt.

```
TextWindow.WriteLine("Regel 1")
TextWindow.WriteLine("Regel 2")
```

Opdracht 2: Hoe werkt het?

Het volgende programma geeft de letters van het alfabet weer. Leg uit hoe het programma werkt.

```
For code = 65 To 90
  ch = Text.GetCharacter(code)
  TextWindow.WriteLine(ch)
EndFor
```

Opdracht 3: Klinkers tellen

Schrijf een programma dat het aantal klinker telt in een door een persoon ingevoerde zin. De uitvoer ziet er als volgt uit:

```
Voer een zin in: Small Basic is leuk
De zin bevat [6] klinkers.
```

```
Voer een zin in: Informatica is het vak van de toekomst
De zin bevat [13] klinkers.
```

Opdracht 4: Palindroom nummer checker

Een ***palindroom*** is een getal, woord of zin die voor- en achteruit hetzelfde leest. 12344321 en 1122332211 zijn bijvoorbeeld palindromen. Ook het woord raar is een palindroom.

Schrijf een programma dat controleert of een geheel getal (*integer*) dat wordt ingevoerd door de gebruiker een palindroom is.

Als de gebruiker niks invoert maar op de <ENTER>-toets drukt of het getal nul invoert dan moet het programma stoppen met vragen voor nummers.

```
Voer een getal in (ENTER=Stop): 1234321
1234321 is palindroom.

Voer een getal in (ENTER=Stop): 12345678
12345678 is geen palindroom.
```