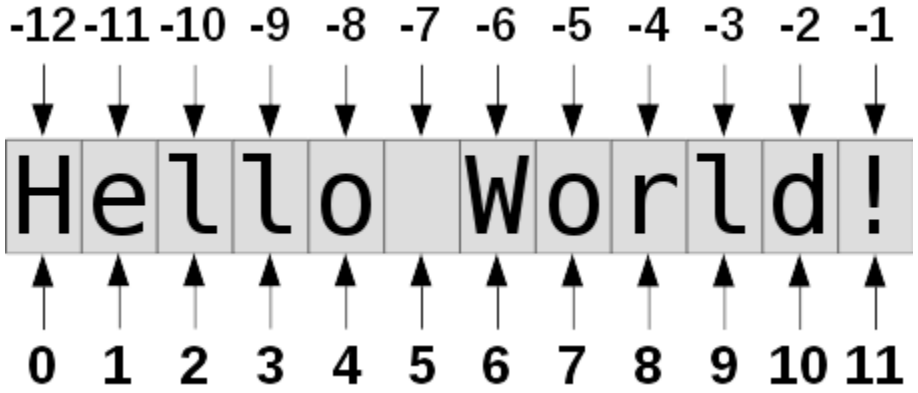
# Strings

# رشته

- رشته یک نوع داده ای در زبان برنامه نویسی پایتون می باشد که از دنباله ای از کارکترها تشکیل شده است که هر کارکتر در رشته در یک موقعیت مشخص قرار می گیرد.
- رشته ها در زبان پایتون درون " " یا ' ' و یا """ """ قرار می گیرند.
- برای دسترسی به کارکترهای یک رشته از indexing استفاده می کنیم. برای ایجاد index یک عدد صحیح را که نشان دهنده موقعیت کارکتر می باشد درون یک جفت براکت قرار می دهیم.



# رشته

- برای دسترسی به کارکترهای یک رشته از loop هم می توان استفاده کرد.

```
word = “HELLO”
for c in word:
    print(c)
```

Output:

```
H
E
L
L
O
```

# عملیات روی رشته ها

- برای محاسبه طول یک رشته از تابع len() استفاده می شود.

```
>>> my_string = 'Hello, World!'
>>> len(my_string)
Output:  13
```

- برای پیوند دادن چند رشته از عملگر + استفاده می شود.

```
>>> str1 = 'Hello‘
>>> str2 = ‘World!'
>>> print (str1 + str2)
Output: 'HelloWorld!‘
```

- برای تکرار یک رشته از عملگر * استفاده می شود.

```
>>> print (str1 * 3)
Output:  HelloHelloHello
```

# Slicing

- بعضی اوقات لازم است تا قسمتی از یک رشته را استخراج کنیم. به این عمل **Slicing** گفته می شود.

```
substring = bigstring[start:end:step]
```

- substring در عبارت بالا شامل همه کارکترهای bigstring از موقعیت start تا موقعیت end (بجز کارکتر موجود در موقعیت end) می باشد.
- برای انجام عمل slicing حداقل می بایست یک نقطه شروع و یا یک نقطه پایان مورد استفاده قرار گیرد.

## Slicing - ادامه

- مثال

word = “Superman sings in the shower.”

print(word[0:8]) Output: Superman

print(word[9:14]) Output: sings

print(word[:5]) Output: Super

print(word[9:]) Output: sings in the shower.

print(word[-7:]) Output: shower.

print(word[0:len(word):3]) Output: Seasgit or

# Immutable

- رشته ها تغییرناپذیر هستند. یعنی کارکترهای یک رشته را نمی توان تغییر داد. ولی مقدار کل یک رشته را می توان تغییر داد.

```
>>> my_string = 'iLovePizza'
>>> my_string[5] = 'a'
```

TypeError: 'str' object does not support item assignment

- راه حل استفاده از یک رشته جدید

```
>>> greeting = 'Hello, world!‘
>>> new_greeting = 'J‘+ greeting[1:]
>>> print(new_greeting)
Output: Jello, world!
```

## تابع enumerate

- تابع enumerate یک تابع از پیش‌ساخته شده و Built-in در زبان پایتون می باشد. این تابع برای یک لیست، عناصر و اندیس های آن را با هم در نظر می گیرد.

```
>>> str1 = ‘cold‘
>>> list_enumerate = list(enumerate(str1))
>>> print(list_enumerate)
```

Output: [(0,’c’), (1,’o’), (2,’l’), (3,’d’)]

- در تابع enumerate شروع اندیس گذاری را می توان تغییر داد.

```
>>> list_enumerate = list(enumerate(str1, 1))
>>> print(list_enumerate)
```

Output: [(1,’c’), (2,’o’), (3,’l’), (4,’d’)]

# تابع enumerate - مثال

- برنامه ای بنویسید که هر یک از کارکترهای رشته a ۳ بار تکرار کرده و هر یک را در یک خط مجزا با شماره ایندکس نمایش دهد.

```
>>> a = ‘Test‘
>>> for i,v in enumerate(a):
        print(“{}) {}”.format(i,v*3))
```

Output:

```
0) TTT
1) eee
2) sss
3) ttt
```

# Escape Character

| Code | Result |
|---|---|
| \' | Single Quote |
| \\ | Backslash |
| \n | New Line |
| \r | Carriage Return |
| \t | Tab |
| \b | Backspace |
| \f | Form Feed |
| \ooo | Octal value |
| \xhh | Hex value |

- برای اضافه کردن کارکترهایی که برای پایتون مفهوم خاصی دارند به یک رشته، نمی توانیم به سادگی آنها را در متن قرار دهیم. (مثل رشته I'm a student) برای این کار از escape character استفاده می کنیم.

```
>>> Print("Hello\nWorld!")
```

Output:

```
Hello
World!
```

- اگر در ابتدای رشته کارکتر r را قرار دهیم از escape character چشم پوشی می کند.

# String Format

- با استفاده از تابع format() می توانیم آرگومان هایی را دریافت کرده و آنها را درون یک رشته جایی که با {} مشخص شده، قرار دهیم.

```
>>> age = 20
>>> str1 = “My name is Jack, I am {}”
>>> print(str1.format(age))
```

Output: `My name is Jack, I am 20`

- تابع format می تواند تعداد نامحدودی آرگومان از ورودی دریافت کند.

```
>>> num1 = 5
>>> str1 = “{} ** 2 =”
>>> print(str1.format(num1), num1*num1)
```

Output: `5 ** 2 = 25`

## String Format - ادامه

- در تابع format از شماره ایندکس می توان برای تضمین اینکه آرگومان در جای مناسب قرار گیرد استفاده کنیم.

```
>>> text = “{1}, {0} and {2}”.format(‘John’,’Bill’,’Sean’)
>>> print(text)
```

Output: `Bill, John and Sean`

# String Methods

- شی رشته نیز مثل سایر اشیا در زبان های برنامه نویسی دارای تعداد زیادی method می باشد.
- همه این متدها مقداری را بر می گردانند و مقدار رشته اصلی را تغییر نمی دهند.
- متد **upper()** تمام کارکترهای موجود در متن را به حروف بزرگ تبدیل می کند.

```
S = "Converts Lowercase Letters In String To Uppercase"
print(S.upper())

# output: CONVERTS LOWERCASE LETTERS IN STRING TO UPPERCASE
```

- متد **lower()** تمام کارکترها را به حروف کوچک تبدیل می کند.

```
S = "Converts UPPERCASE Letters In String To Lowercase"
print(S.lower())

# output: converts uppercase letters in string to lowercase
```

# String Methods

- متد **count()** تعداد کارکتر مشخصی را که در یک رشته تکرار شده است، می شمارد. آرگومان ورودی این متد زیر رشته ای بوده که می خواهیم دفعات تکرار آن را پیدا کنیم.

```
S = "returns the number of occurrences of substring"
substring = "n"
print("the number of 'n' in this string is: " , S.count(substring))

# output: the number of 'n' in this string is:  4
```

- متد **find()** مکان یک زیر رشته را برای ما پیدا می کند. در صورت عدم وجود مقدار 1- را باز می گرداند. برای بررسی وجود یک زیر رشته از اپراتور in هم استفاده می شود.

```
S = "this method return index if found and -1 otherwise"
print(S.find("index"))
print(S.find("with"))
# 19
# -1
```

# String Methods

- متد **index()** موقعیت اولین رخداد یک مقدار مشخص در رشته را بر می گرداند.
- ساختار کلی متد بصورت زیر می باشد:

*string.index(value, start, end)*

- مثال

```
S = “Hello, welcome to my world.“
x = S.index(“e",5,10))

# output: 8
```

# String Methods

- متد **split(separator, maxsplit)** کلمات یک رشته را به یک لیست تبدیل می کند.

```
S = "zero \nto \nhero"
print(S.split())
print(S.split(" ",1))

"""
output:
    ['zero', 'to', 'hero']
    ['zero', '\nto \nhero']
"""
```