# ITP 115

## String Processing

## Input

- The **input** function in Python *always* returns a string even when we want the user to enter a number.
- We use the **int** function to convert the string to an integer.

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))
```

# Bad Input

- What if the user doesn't enter a number?

```
age = int(input("Enter your age: "))
```

```
Enter your age: twenty
Traceback (most recent call last):
  File "../Errors.py", line 6, in <module>
    age = int(input("Enter your age: "))
ValueError: invalid literal for int() with base 10: 'twenty'
```

# String Error Checking Methods

- `string` is a variable holding a string

| Method | Description |
|---|---|
| `string.isalnum()` | Returns `True` if `string` contains only letters and numbers<br>Returns `False` otherwise |
| `string.isalpha()` | Returns `True` if `string` contains only letters<br>Returns `False` otherwise |
| `string.isdigit()` | Returns `True` if `string` contains only digits<br>Returns `False` otherwise |
| `string.isspace()` | Returns `True` if `string` contains only whitespace<br>Returns `False` otherwise |

## Example – isdigit

- Use the **isdigit** method to make sure the user enters a number.

```
ageStr = input("Enter your age: ")
while ageStr.isdigit() == False:
   ageStr = input("Enter a number for your age: ")

age = int(ageStr)
```

## Check Strings

- `string` is a variable holding a string
- Returns a Boolean

| **Method** | **Description** |
|---|---|
| `string.endswith(value)` | Returns `True` if `string` ends with the specified value<br>Returns `False` otherwise |
| `string.startswith(value)` | Returns `True` if `string` starts with the specified value<br>Returns `False` otherwise |

## Search Strings

- `string` is a variable holding a string
- Returns an integer

| Method | Description |
|---|---|
| `string.count(value)` | Returns the number of times value appears in the string |
| `string.find(value)` | Returns the index of the first occurrence of value<br>Returns -1 if the value is not found |
| `string.index(value)` | Returns the index of the first occurrence of value<br>Raises an exception if the value is not found |

## Return Strings

- **`string`** is a variable holding a string

| Method | Description |
|---|---|
| `string.upper()` | Returns the uppercase version of the string |
| `string.lower()` | Returns the lowercase version of the string |
| `string.capitalize()` | Returns a new string where the first letter is capitalized and the rest are lowercases |
| `string.title()` | Returns a new string where the first letter of each word is capitalized and all others are lowercase |
| `string.strip()` | Returns a new string where all the white space (tabs, spaces, and newlines) at the beginning and end is removed |
| `string.replace(old, new)` | Returns a new string where occurrences of the string old are replaced with the string new |

## Sequences Have Indices!

- Each individual item in a sequence is automatically given a position number
- This number is called an **index** and tells what position the item is in
- The **first index** is **zero (`0`)**
- The **last index** is the **number of items – 1**

## Example: Strings and Indices

```
word = "spamalot"
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| s | p | a | m | a | l | o | t |

- First index is **zero**
- Last index is the **length – 1**
  *(8 letters, but last index is 7)*

## Sequences and Random Access

- Using indices, we can directly access single items from a sequences

- To read a single item from a sequence, we use the [ ] **operator**

- Syntax

```
sequenceVariable[index]
```

## Strings – Random Access

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| s | p | a | m | a | l | o | t |

```
msg = "spamalot"
print(msg[2])
```

```
a
```

```
print(msg[6])
```

```
o
```

# Strings – Random Access

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| s | p | a | m | a | l | o | t |

```
msg = "spamalot"
print(msg[13])
```

```
Error
```

# Index Out of Range

- Only valid indices of a sequence are
  `0` to `length-1` *
- Error if you read index beyond `length-1`
  - Also called "Out of bounds"
- **Common mistake**
  - If a sequence has 5 items, what is the index of the last item?

** Python supports negative indices, which go from -1 to -(length). This is not common in programming languages and we won't use it*

# Slicing

- We can use `[index]` to get a <u>single item</u> from a sequence
- We can use **slicing** to get <u>multiple items</u> from a sequence
- Slicing works with any sequence (e.g. string, list, etc.)

# Slicing

- Syntax

```
sequenceVariable[startPosition:endPosition]
```

*Access from start position*

*Go **UP TO BUT NOT INCLUDING** end position*

# Slicing Strings

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| s | p | a | m | a | l | o | t |

**Examples**

```
print(msg[2:6])
```

```
amal
```

```
print(msg[3:4])
```

```
m
```

```
print(msg[0:7])
```

```
spamalo
```

# Slicing Strings

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| s | p | a | m | a | l | o | t |

- What if we want the whole string?

```
print(msg[0:8])
```

```
spamalot
```

## Slicing Strings

- What if we want the whole string BUT we don't know how long the string is?

```
msg = input("Enter a word: ")
print(msg[0:len(msg)])
```

*This works because we go from* ***0*** *up to but not including* ***length***

# Useful Slicing Tricks

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| s | p | a | m | a | l | o | t |

- Start at beginning

```
print(msg[:3])
```

```
spa
```

- Go to end

```
print(msg[4:])
```

```
alot
```

- Entire word

```
print(msg[:])
```

```
spamalot
```

# `find()`

- Searches a string for first match of a substring
- Returns a index the first match
  - `Or -1 if not found`
- Syntax

```
index = string.find(subString)
```

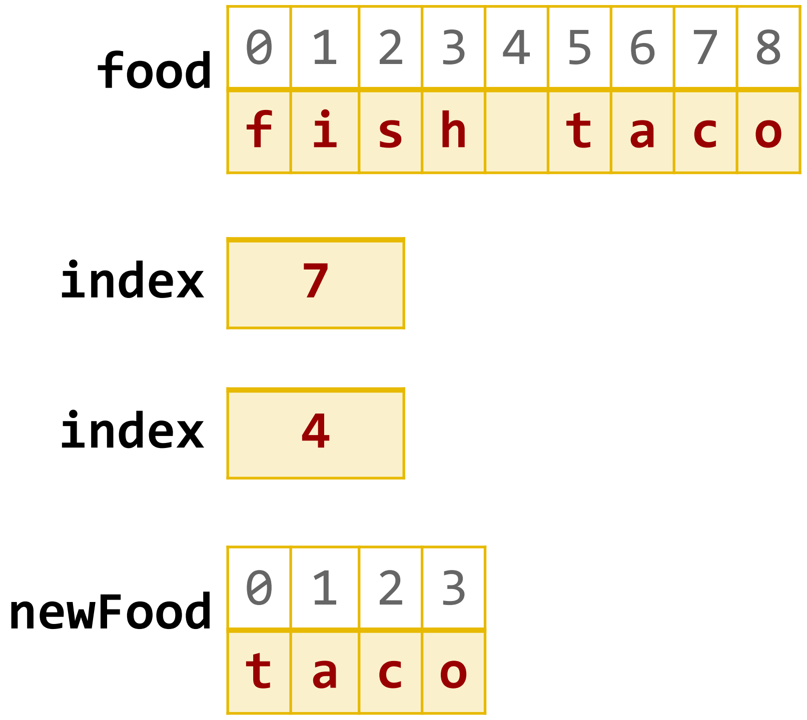
# Example find()

```
food    = "fish taco"

index   = food.find("c")

index   = food.find(" ")

newFood = food[index+1:]
```



## Two Categories of Sequences

- Mutable – changeable
  - Can modify A SINGLE item in the sequence

- Immutable – unchangeable
  - Can **NOT** modify A SINGLE item in the sequence

## Strings are Immutable

```
word = "game"
print (word)
word[0] = "l"
```

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