Programming in Python

Sequences

Strings are Sequences

- Each item in a sequence has a position number

index	0	1	2	3	4
element	t	u	l	i	p

```
word = "tulip"
print(word)
```

```
tulip
```

- First index is 0
- Last index is length - 1

```
size = len(word)
print(size)
```

```
5
```

Access an Element

- Use an index to access a single element
- Pseudocode

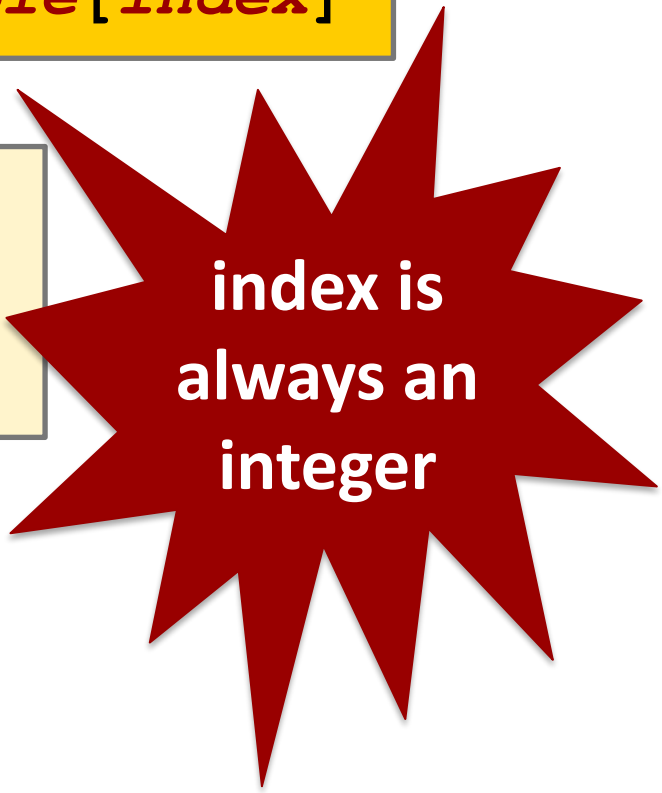
```
element = variable[index]
```

- Example code

```
word = "tulip"
letter = word[3]
print(letter)
```

- Output

```
i
```



What will happen?

```
animal = "snow leopard"
letter = animal[3]
print("letter at index 3 is", letter)
letter = animal[12]
print("letter at index 12 is", letter)
```

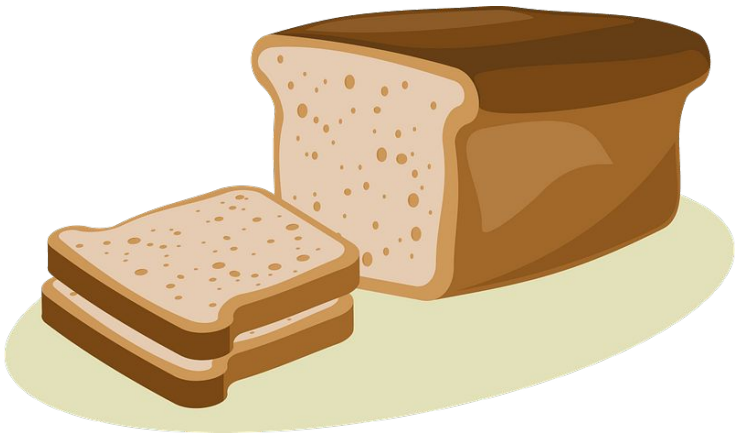
```
letter at index 3 is w
Traceback (most recent call last):
  File "main.py", line 10, in <module>
    letter = animal[12]
IndexError: string index out of range
```

Slicing

- Use slicing to get multiple elements substring
- Pseudocode

```
variable[index1:index2]
```

- **`index1`** is the start position
- **`index2`** is the ending position
  - › Up to but not including



Slicing Example

```
message = "Write code"
slice = message[2:4]
print(slice)
message = "Fish out of water"
slice = message[1:3]
print(slice)
message = "Programming is fun"
slice = message[15:len(message)]
print(slice)
```

```
it
is
fun
```

Slicing

- Pseudocode

```
variable[index1:index2]
```

- **`index1`** is optional
  - › default is **0**
- **`index2`** is optional
  - › default is **`len(variable)`**

Slicing Example

```
msg = "Have a great day!"
slice = msg[:4]
print(slice)
msg = "Programming is fun"
slice = msg[15:]
print(slice)
```

```
Have
fun
```

Programming in Python

String Methods

String Lower

- Use str.lower() to get a lowercase version

```
user = input("Are you 18 or older (y/n)? ")
if user.lower() == "y":
   print("You can vote!")
else:
   print("You are not old enough to vote.")
```

```
Are you 18 or older (y/n)? Y
You can vote!
```

```
Are you 18 or older (y/n)? q
You are not old enough to vote.
```

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 lowercase
***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

Return Strings

- Assign the new string to the original variable

```
user = input("Are you 18 or older (y/n)? ")
user = user.lower()
if user == "y":
   print("You can vote!")
else:
   print("You are not old enough to vote.")
```

Convert Strings

- What happens?

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

```
Enter your age: twenty
Traceback (most recent call last):
  File "main.py", line 1, in <module>
    age = int(input("Enter your age: "))
ValueError: invalid literal for int() with base 10:
'twenty'
```

Check for Integer

- Update code

```
age_str = input("Enter your age: ")
while age_str.isdigit() == False:
  age_str = input("Enter your age: ")
age = int(age_str)
print("You are", age)
```

```
Enter your age: twenty
Enter your age: 2 0
Enter your age:  20
Enter your age: 20
You are 20
```

Check Strings

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

Method	Description
***string*`.isalnum()`	Returns **`True`** if **`string`** contains only letters and numbers Returns **`False`** otherwise
***string*`.isalpha()`	Returns **`True`** if **`string`** contains only letters Returns **`False`** otherwise
***string*`.isdigit()`	Returns **`True`** if **`string`** contains only digits (integers) Returns **`False`** otherwise
***string*`.isspace()`	Returns **`True`** if **`string`** contains only whitespace Returns **`False`** otherwise

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 Returns **False** otherwise
string**.startswith(value)**	Returns **True** if **string** starts with the specified value 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 Returns -1 if the value is not found
***string*`.index(value)`	Returns the index of the first occurrence of value Raises an exception if the value is not found

Example

```
food = "qabili palau"
print(food)
index = food.find(" ")
print(index)
new_food = food[index+1:]
print(new_food)
```

```
qabili palau
6
palau
```

0	1	2	3	4	5	6	7	8	9	10	11
q	a	b	i	l	i		p	a	l	a	u

Programming in Python

Lists

Loops

- Remember loops

```
total = 0

for num in range(30):
    score = int(input("Enter score: "))
    total += score

average = total / 30
print("Average = " + average)
```

```
Enter score: 97
Enter score: 82
# 27 more times
Enter score: 91
Average = 90
```

- What if we needed to keep all of the scores?
- Do we have to create 30 variables?

List

- New type of variable
- A collection of things
- Can contain different variable types
  - ints, floats, strings, ...
  - even other lists
- Mutable
- Sequence

Lists

- Pseudocode

```
variable = [item1, item2, ...]
```

- Code

```
# list to hold numbers
scores = [97, 82, 88, 93, 95, 91]

# list to hold strings
colors = ["red", "orange", "yellow", "green"]

# empty list
names = []
```

Empty List

- You can create an empty list one of two ways:

```
variable = []
```

```
variable = list()
```

```
# create an empty list
scores = []
print(scores)

# create an empty list
names = list()
print(names)
```

```
[]
[]
```

Use list()

- You can use the list() function to convert a string into a list

```
variable = list(string)
```

```
word = "python"
print(word)
letterList = list(word)
print(letterList)
```

```
python
['p', 'y', 't', 'h', 'o', 'n']
```

Sequences

- Lists are sequences
  - › Have order

```
scores = [97, 82, 88, 93, 95, 91]
```

index	0	1	2	3	4	5
element	97	82	88	93	95	91

```
colors = ["red", "orange", "yellow", "green"]
```

index	0	1	2	3
element	"red"	"orange"	"yellow"	"green"

Concatenate

- Concatenate lists together

```
animals = ["emu", "pig"]
print(animals)
pets = ["dog", "cat", "boa"]
print(pets)
zoo = pets + animals
print(zoo)
```

```
['emu', 'pig']
['dog', 'cat', 'boa']
['dog', 'cat', 'boa', 'emu', 'pig']
```

Access an Element

- Use an index to access an element

```
colors = ["red", "orange", "yellow", "green"]
colors += ["blue", "purple"]
print(colors)
fav_color = colors[4]
print(fav_color)
```

```
['red', 'orange', 'yellow', 'green', 'blue', 'purple']
blue
```

- What type is **fav_color**?

string

Slice

- Slice a list to get multiple elements

```
colors = ["red", "orange", "yellow", "green"]
colors += ["blue", "purple"]
print(colors)
cool_colors = colors[3:]
print(cool_colors)
```

```
['red', 'orange', 'yellow', 'green', 'blue', 'purple']
['green', 'blue', 'purple']
```

- What type is **cool_colors**?

`list`

Number of Elements

- Use the **`len()`** function

```
colors = ["red", "orange", "yellow", "green"]
colors += ["blue", "purple"]
print(colors)
num = len(colors)
print("There are " + str(num) + " colors.")
```

```
['red', 'orange', 'yellow', 'green', 'blue', 'purple']
There are 6 colors.
```

Programming in Python

List Methods

Append

- Add an element to the end of the list

```
colors = ["red", "orange", "yellow", "green"]
print(colors)
colors.append("blue")
colors.append("purple")
print(colors)
```

```
['red', 'orange', 'yellow', 'green']
['red', 'orange', 'yellow', 'green', 'blue', 'purple']
```

- More efficient than using + (concatenate)
- Only can add one element at a time

Remove

- Remove an element

```
colors = ["red", "orange", "yellow", "green"]
print(colors)
colors.remove("yellow")
print(colors)
```

```
['red', 'orange', 'yellow', 'green']
['red', 'orange', 'green']
```

- Make sure the element is in the list

```
if "blue" in colors:
    colors.remove("blue")
```

Index

- Get the index of an element

```
colors = ["red", "orange", "yellow", "green"]
print(colors)
num = colors.index("yellow")
print(num)
```

```
['red', 'orange', 'yellow', 'green']
2
```

Random Element

- Choose a random element using the **random.choice()** function from the random module

```
import random

colors = ["red", "orange", "yellow", "green"]
print(colors)
col = random.choice(colors)
print(col)
```

```
['red', 'orange', 'yellow', 'green']
orange
```

- What type of variable is **col** ?

string

Lists are Sequences

- A **`for`** loop repeats based on a sequence

```
for newVariable in sequence:
```

```
animals = ["fox", "panda", "zebra", "koala", "bear"]
for item in animals:
    print(item)
```

```
fox
panda
zebra
koala
bear
```

- What type is **`item`**?

`str`

Lists are Sequences

- A **`for`** loop repeats based on a sequence

```
for newVariable in sequence:
```

```
scores = [97, 82, 88, 93, 95, 91]
for item in scores:
    print(item)
```

```
97
82
88
93
95
91
```

- What type is `item`?

```
int
```