Bloomberg STARTUP

Variables

A **variable** is a name given to a piece of data that can be used to refer to it later.

```python
string_variable = "Hello World!"
int_variable = 23
float_variable = 71.5
```

Functions

A **function** is a reusable, named piece of code that can take inputs, called parameters.

```python
def print_greeting(name):
    greeting = "Hello, " + name
    print(greeting)
    return greeting
```

- **def** tells the program you are defining a function
- You can pass values into the function, called **parameters**
- The **:** tells the program you are starting the code block for the function
- The function code must all be **indented** the same amount
- The function can **return** a value

Conditionals

if *boolean expression*:
    statements
elif *boolean expression*:
    statements
else:
    statements

```python
x = int(input("Enter a number: "))
if x > 0:
    print(x, "is positive")
elif x < 0:
    print(x, "is negative")
else:
    print(x, "is 0")
```

- **if** tells the program you are starting a conditional statement
- A condition evaluates to either **True** or **False**
- The **:** tells the program you are starting the statements block
- The statement code must all be **indented** the same amount
- **elif** tells the program you want to evaluate the 2nd condition if the 1st condition evaluates to **False**
- **else** tells the program you want to execute the statements if both the 1st and the 2nd conditions evaluate to **False**

While Loops

while *boolean expression*:
    statements

```python
number = 1
while number <= 10:
    print(number)
    number = number + 1
```

- **while** indicates that the code block will repeat while the condition is **True**
- A condition evaluates to either **True** or **False**
- The **:** tells the program you are starting the statements block
- The statement code must all be **indented** the same amount
- **Incrementing** the number is very important to end the loop
- Beware of **infinite** loops

Boolean Logic

a	b	a **and** b	a **or** b
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

a	**not** a
True	False
False	True

Bloomberg

STARTUP

Lists

A **list** is a collection of elements which are organized in order from first to last.

```
animals = [ "lion", "bear", "cat" ]
numbers = [ 4, 23, 16, 38 ]
```

- A list starts with a **[**
- Each Item in the list is called an **element**, separated by commas
- The position of elements in the list is called the **index**
- **]** tells the program you are at the end of a list

List Functions

Access elements in a list:
list[index]

Add an element to the end of a list:
list.append(x)

Remove an element from a list:
list.remove(x)

Organize a list:
list.sort()

Functions that operate on lists:
len(list)
max(list)
min(list)
sum(list)

For loops

For loops allow us to repeat a block of code a certain number of times.
for *element* **in** *list/range of numbers/string***:**
do something for each element

```
for number in range(1, 11):
    print(number)

animals = [ "lion", "bear", "cat" ]

for animal in animals:
    print(animal)
```

- **for** indicates that the code block will repeat in a loop
- The **variable** can be used to access each element
- The **:** tells the program you are starting the statements block
- The statement code must all be **indented** the same amount

Dictionaries

Dictionaries store data in the form of key-value pairs. A **key** is the name of the data. A **value** is the piece of data you want to associate to the key name. In order to look up the **value**, you need to use the **key**.

```
tasty_snacks = {
    "oreos" : 2.75,
    "doritos" : 1.25,
    "donuts" : 0.80
}
```

- A dictionary starts with a **{**
- Elements are comma separated **key-value** pairs
- Key-Value pairs are written as **key : value**
- Keys must be **unique**
- **}** tells the program you are at the end of a dictionary

Edit data in a dictionary:

```
tasty_snacks["oreos"] = 2.50
```

Add data to a dictionary:

```
tasty_snacks["pringles"] = 4.50
```

Loop through a dictionary:
for *key* **in** *dictionary***:**
do something for each key-value pair

```
for snack in tasty_snacks:
    print(snack, "cost",
          tasty_snacks[snack])
```

Test if a key is in a dictionary:
if *key* **in** *dictionary***:**
statements

```
if "oreos" in tasty_snacks:
    print("I love oreos!")
```