INFO5002: Intro to Python for Info Sys

Week 4

Slides created by: Zachary Doucet

LVX
VERITAS
VIRTVS

Northeastern
University

Week 4

I. Loops

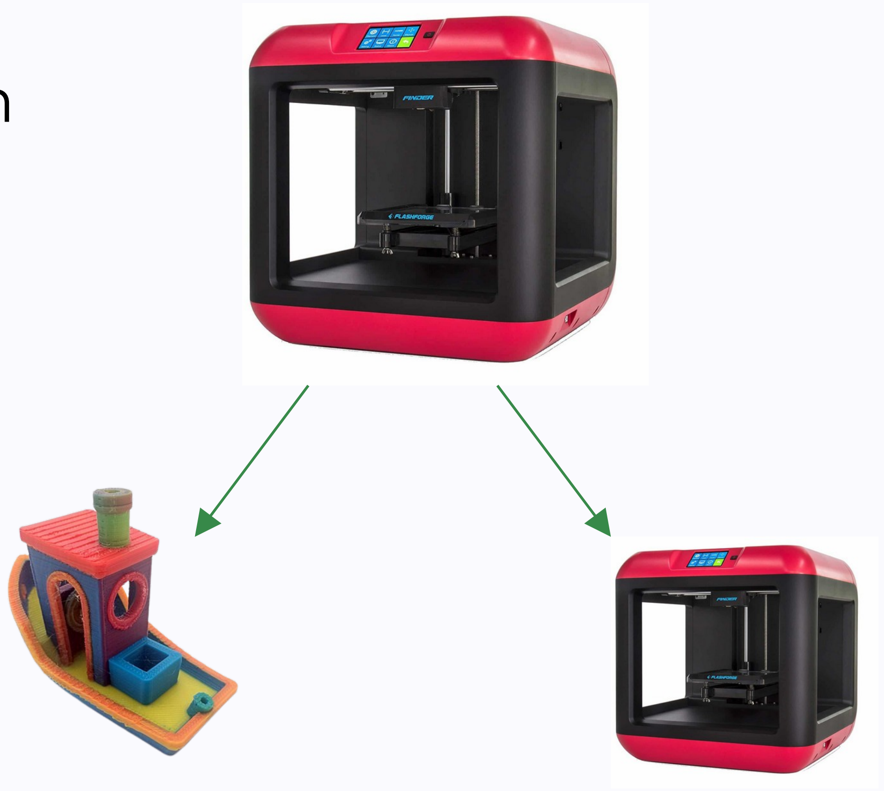
II. Advanced Data Types

Recap

Higher Order Functions

- Functions usually return data.

- What if we return a function?

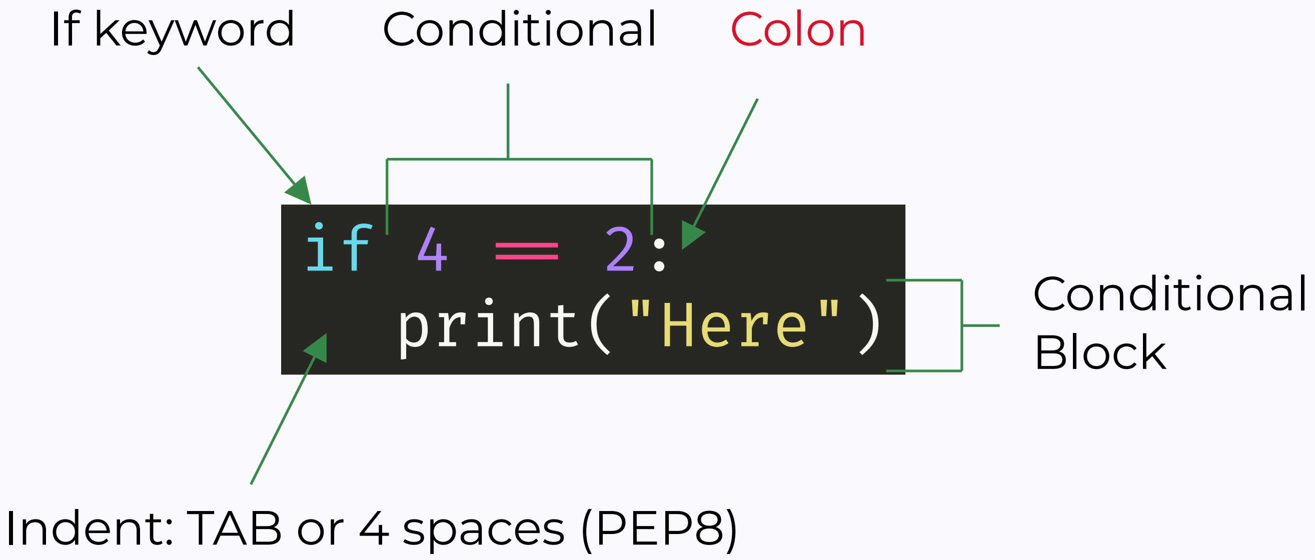


Functions calling themselves

```python
def countdown(t):
  print(t)
  countdown(t-1)
```

Source: Sergey Pykhonin

Conditionals allow for branching



```python
if 4 == 2:
    print("Here")
```

Loops

PCC 113-127

Loops reduce repetition

- We repeat a group of operations together under a loop to reduce re-writing.
- Let's say we want to print "hello" five times without using the repetitions operator.

```
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
```

```
do block five times:
    print("hello")
```

But how?

The while loop

- If we want to repeat a block of code while a condition is True then use while.

```python
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
```

$\longrightarrow$

```python
x = 0
while x < 5:
    print("hello")
    x += 1
```

Change the path

- You can change the executing path with break and continue.

How many times does hello print?

5 and 3

```python
x = 0
while True:
    print("hello")
    x += 1
    if x >= 5:
        break
```

```python
x = 0
while x < 5:
    x += 1
    if x % 2 == 0:
        continue
    print("hello")
```

Let’s practice

- Create the following function:

    I. print_hello_x which takes in an integer and prints “hello” integer number of times. Don’t use the repetition operator.

    II. riemann_sum which takes in an integer and returns the sum of all numbers from 0 to the argument (including). Do not use the closed form.

    III. riemann_sum_lower which takes in two integers and returns the sum of all the numbers between the first (including) to the second (including). Do not use closed form.

And some more

- Create the following function:

  I. sum_even which given an input sums together all the numbers from 0 to the input (inclusive) that are even.

  II. get_age which keeps asking for user input until the age is a valid human age (assume humans don't live beyond 150y) and returns that age.

  You can get user input with input. `x = input("This is the prompt")`

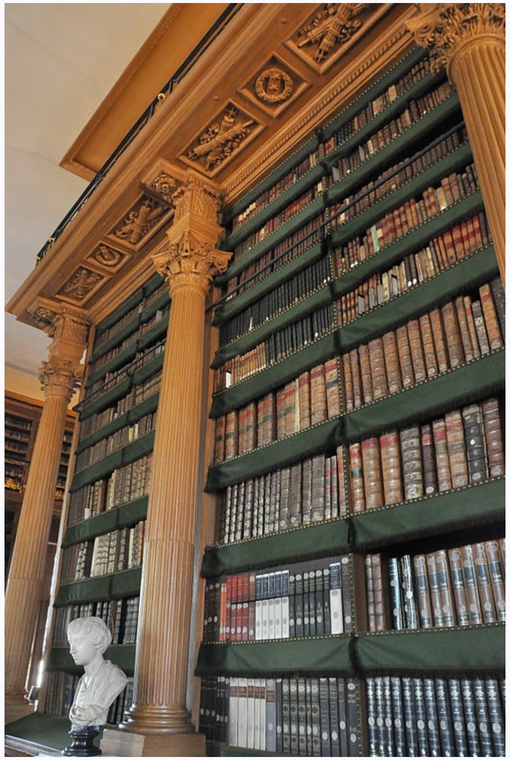
  You can turn a string into an int with int. `x = int(x)`

Advanced Data Types

PCC 33-70 and 91-112

Lists

- To hold a collection of data you can use lists.
  - Todo list
  - Bookshelf
  - Roster


Source: Wikimedia

Working with lists

- We can create a list with brackets.

```
x = []
y = [1, 2, 3]
```

0 1 2 index

- Lists hold items in a specific order.

- Use brackets to access items on a list.

```
z = [2, 3, 4]
z[1]
```

Watch out: IndexError

- Can also use brackets to assign at a location.

```
q = ["a", "b", "c"]
q[2] = "f"
```

```
x = [1, 8, 4, 2]
_len = len(x)
```

- We can get the number of elements in a list with len.

Modifying lists

- We can add to the end of a list with append.

```
x = [1, 2]
x.append(3)
```

- We can add to a specific index with insert.

```
y = ["a", "c"]
y.insert(1, "b")
```

- We can delete at a specific index with del or pop.

```
x = [1, 2, 3]
del x[0]
```

or

```
x = [1, 2, 3]
x.pop(0)
```

- We can remove at the end with pop.

```
x = [1, 2, 3]
x.pop()
```

Modifying lists (continued)

- We can remove a specific value *once* with remove.

```
x = [1, 8, 4, 2]
x.remove(8)
```

- We can get elements between two indices with **slice**.

```
x = [1, 2, 3, 4, 5]
y = x[1:2]
```

inclusive   exclusive

Tuples

- Immutable ordered collection to store multiple data together.
- Create using parentheses.

```
salad = ("spinach", "tomato", "vinegar")
```

- Get an element with index operator.

```
first_ingredient = salad[0]
```

- Get number of elements with len function.

Dictionaries

- Collection of ordered mutable key-value pairs.
- You cannot have duplicate keys.
- Define with braces and colons.

```
x = {"model": "Kia Rio", "year": 2003, "mpg": 25.32}
```

- Get element with index operator. `x["model"]`
- Modify value of specific key with index op. `x["year"] = 2012`

Dictionaries (continued)

- Get number of elements with len function.

Sets

- Unordered unindexed collection of unique values.
- Define with braces. `fruits = {"mango", "apple", "pear"}`
- Add with add. `fruits.add("banana")`
- You **cannot** access a specific element (unindexed) therefore must use a loop!
- Remove element with remove. `fruits.remove("mango")`

Sequence iteration with for

- We can iterate over a sequence using the for loop.

```
x = [1, 2, 3, 4, 5]
sum = 0
for i in x:
   sum += i
```

```
y = (1, 10, 15)
product = 1
for val in y:
   product *= y
```

```
z = "turnip”
reverse = ""
for c in z:
   reverse = c + reverse
```

```
d = {"x": 1, "y": 2}
resultant = 1
for k in d:
   resultant /= d[k]
```

```python
x = {"apricot", "apple", "banana"}
num_a = 0
for fruit in x:
   for character in fruit:
      if character == "a":
         num_a += 1
```

Remember trying to print 5 times?

- We initially did so with a while loop but we can do so also with a for loop.

```python
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
```

```python
x = 0
while x < 5:
    print("hello")
    x += 1
```

special library function

```python
for i in range(0,5):
    print("hello")
```

The range function

- We can create a sequence of numbers starting from *a* to excluding *b* with a step of *c* with range.

```
range(start, stop, step)
```

What do I see when I execute?

```
for i in range(2, 10, 2):
    print(i)
```

```
2
4
6
8
```

The range function shorthands

`range(one_argument)` == `range(0, one_argument, 1)`

`range(arg1, arg2)` == `range(arg1, arg2, 1)`

Popular Functions

Python docs

Embed variables into strings

- We used the concatenation operator to build strings.
- We can make it more readable with **string interpolation** (know in python as f-strings).

```
first_name = "grace"
last_name = "hopper"
full_name = f"{first_name} {last_name}"
```

String special characters

Characters	**Effect**
\n	New line
\t	Add a tab
\r	Carriage return
\f	Form feed
\b	Backspace
\\	Backslash
\’	Single quote
\”	Double quote

String functions

- .title(): capitalises every word in a string.

```
x = "hello there"
x = x.title()
```

- .upper(): capitalises every character in a string.
- .lower(): minimises every character in a string.
- .split(separator?, maxsplit?): splits string along separator, default “ ”, maxsplit number of times, default infinity.

```
x = "today is Friday".split()
# ["today", "is", "Friday"]
```

```
x = "tomorrow".split("o")
# ["t", "m", "rr", "w"]
```

Data type casting

You can convert data types to:

- Int with int

```
x = int(2.8)
x = int("3")
```

- Float with float

```
x = float(6)
x = float("8.2")
```

- String with str

```
x = str(1)
x = str("12.9")
```

- Tuple with tuple

```
x = tuple([1, 2, 3])
x = tuple("apple")
```

- Set with set

```
x = set([1, 2, 3])
x = set("apple")
```

Interfacing

- print: copy input string to user's terminal.

```
print("Hello there")
```

- input: ask user for input with optionally a prompt of input argument.

```
x = input()
y = input("Age: ")
```

Documentation to find more!

- Python is a rich language with many built in features.
- To find all the build in functions you can look in the documentation: https://docs.python.org/3/.