Intro to Programming

4-09-19

Sign In!

https://tinyurl.com/CADIntroPySp19-7

https://repl.it/languages/python3

Last Week's Review

What will the following code output?

```
def greet_user(first, last, middle=""):
    """ Greet a user """
    if middle:
        return ("Hello, " + first.title() + " " + middle.title()
+ " " + last.title())
    else:
        return ("Hello, " + first.title() + " "  + last.title())

print(greet_user(last="Fenves", middle="L.", first="Gregory")
```

Last Week's Review

What will the following code output?

```
def greet_user(first, last, middle=""):
    """ Greet a user """
    if middle:
        return ("Hello, " + first.title() + " " + middle.title()
+ " " + last.title())
    else:
        return ("Hello, " + first.title() + " "  + last.title())

print(greet_user(last="Fenves", middle="L.", first="Gregory")
```

OUTPUT: Hello, Gregory L. Fenves

Tuples

- Similar to lists, but it can't be modified after creating it
  - Elements can't be removed, appended, replaced
  - Instead of having `[]`, tuples are defined between `()`
- Access is the same: `tup[1]` is the second item in the tuple
- Can redefine the whole tuple

```
my_tuple = ("x", "y", "z")
```

`my_tuple:`

Index	0	1	2
Element	"x"	"y"	"z"

Tuples

Good Syntax:

```
my_tuple = (3, 4, 2)
print(my_tuple[1])

for value in my_tuple:
    print(value)
my_tuple = ("haha")
print(my_tuple[0])
```

OUTPUT:

```
4
3
4
2
haha
```

Tuples

Good Syntax:

```
my_tuple = (3, 4, 2)
print(my_tuple[1])

for value in my_tuple:
    print(value)
my_tuple = ("haha")
print(my_tuple[0])
```

Will cause errors:

```
my_tuple = (3, 4, 2)
my_tuple[1] = 1
```

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

Tuples

Good Syntax:

```
my_tuple = (3, 4, 2)
print(my_tuple[1])

for value in my_tuple:
    print(value)
my_tuple = ("haha")
print(my_tuple[0])
```

Will cause errors:

```
for i in range(len(my_tuple):
    my_tuple[i] = my_tuple[i] + 1
```

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

User Input

- You can take user input with the `input()` function
  - `input()` pauses your program and waits for the user enter some text
  - It will then return a string
  - You can pass it a string as an argument and it will print it out to the console

```
def parrot():
    """ Repeats whatever you say """
    message = input("*SQUAWK* Tell me something!")
    print("*SQUAWK* " + message + " *SQUAWK*")
```

Input, continued

- Note that `input()` will return **`strings.`**
- The code to the right will result in a **`TypeError`**
  - This is because you can't compare a string to an int
- To avoid this, use `int()` on the input function

```
def can_vote():
    age = input("How old are you? ")
    if age >= 18:
        return True
```

`TypeError` because age is a 'str' right now

Input, continued

- Note that `input()` will return **`strings.`**
- The code to the right will result in a **`TypeError`**
  - This is because you can't compare a string to an int
- To avoid this, use `int()` on the input function

```
def can_vote():
    age = int(input("How old are you? "))
    if age >= 18:
        return True
```

Using while loops

- We can use while loops to keep programs running as long as the user wants
  - There are multiple ways to do this
- The simplest is checking the user's input directly
- As soon as the user types in 'quit', the while loop will terminate and the program will end

```
def parrot_while_var():
    """ Repeats whatever you say """
    message = ""
    while message != "quit":
        message = input("*SQUAWK* Tell me something! ")
        print("*SQUAWK* " + message + " *SQUAWK*")
    print("Goodbye!")
```

Using while loops, cont'd

- Another way is through the use of flags
  - This is helpful in more complicated programs where multiple things could cause the program to end, or the input is of varying data types
- The flag variable acts as a signal to the program to quit - by changing it to **`False`** we can terminate the while loop

```
def parrot_while_var():
    """ Repeats whatever you say """
    message = ""
    while message != "quit":
        message = input("*SQUAWK* Tell me something! ")
        print("*SQUAWK* " + message + " *SQUAWK*")
    print("Goodbye!")
```

Using while loops, cont'd

- Yet another way is through the use of the **`break`** statement
- As we learned before, we can use break to exit a while loop.

```
def parrot_while_break():
    """ Repeats whatever you say """
    while True:
        message = input("*SQUAWK* Tell me something! ")
        if message == 'quit':
            break
        else:
            print("*SQUAWK* " + message + " *SQUAWK*")
    print("Goodbye!")
```

Example program

You can use **`input()`** and **`while`** loops together to make programs that wait for user input.

```
1.  def even_odd():
2.      active = True
3.      user_in = ""
4.      while active:
5.          user_in = input("Enter a number and I will tell you if it is even or odd: ")
6.          if user_in != "quit":
7.              user_in = int(user_in)
8.              if (user_in % 2 == 0):
9.                  print("Your number is even")
10.             else:
11.                 print("Your number is odd")
12.         else:
13.             active = False
```

Exercises

- Write a program that polls users about their dream vacation - similar to "If you could visit one place in the world, where would you go?" Continue to take input until the user enters "quit". Include a block of code that prints the results of the poll .

- Check our Week 7 Github and download `p1.py` for another exercise: (https://github.com/cadtexas/sp19-intro-to-python/tree/master/4-09-19_Week_7)

Thanks for coming!

- Next week - perhaps a project!
- Please fill out our feedback form, especially if you'd like to specify which topics we cover next week!
  - https://forms.gle/EDHGxj9o93uTxtdz9