Python Language

Enes Kemal Ergin

Questions of the Week

- What are variables and how we use them?
- How we use spacing in Python?
- Commenting on what?
- Can we use basic math operations?
- What are the numeric types in Python?
- What is so essential about strings?
- What is Boolean type?

Little bit about the modes…

- Python has **interactive environment**, which means you can type and see the result at that moment. You don't have to compile all the file and then wait for the result…
- I will demonstrate examples at beginning on both command prompt and canopy python shell. They both pretty same btw.
- But later on we can use **scripting mode** which requires to put all you got in file save it as .py file than run it on Python.

Variables/Identifiers

- We might use bunch of different data types but for storing them we all need variables.
- **variable_example = “My name”**
- A variable stores a piece of data and gives a specific name…
- We call that piece of data value, in our case is “My name” string type.
- From now on unless we rewrite, or close the shell we can use variable_example variable with “My name” value in it.
- Python has some limitations to naming variable.
  - You cannot allow punctuation characters such as: $,@,%
  - Python is case sensitive so Name and name is different variables.
  - You cannot use reserved words. >>>(next page)

List of Reserved Words

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Crucial spacing

- Python depends on proper indentation. Good news, you don't have to put brackets to point out that you're going one level deeper.
- Spacing in Python is used to structure your code.
- Don't try to understand the concept for now:
  - **def square(x):**
    - **return (x ** 2)**
- Other wise it will throw an indentation error. If you get something like:
  - **IndentationError: expected an indented block**
- You should check your code for spaces…
- But a great thing about Python it won't crash after the bad line. Lines of code before error will be resulted without error.

Commenting Around

- '#' sign is using as a line comments
- ''' triple single quotation marks are opens a multi line comments after you close with ''' all content between will be comment
- Comments are insignificant and essential at the same time.
- Comments are very reader friendly.
- **name = "E.Kemal" # This is line comment**
- **''' This is multi line comment**

  Means that you can write every line

  In between triple quotes

  Then Python ignores them

  '''

Python knows Math

- Math is fun in Python.
- You can use Python as a basic calculator.
- **1 + 1**
- **7 - 9**
- **2 * 5**
- **5 ** 2**
- This is what you can do with Python in real time.
- **21 / 3**
- Some thing is not quite right here!

Division Puzzle

- When you divide 21 / 3 it gives you result 7.0 with fraction. Why?
- Python version 3.X is gives a float number when you divide 2 integers to avoid some mistakes which occurred in Python version 2.X
- In old version (still popular) when you divide 21/3 it gives you 7. It caused serious issues in interpreter like:
- 1 / 2 it resulted as 0, which is unacceptable for people in scientific areas.
- Now in Python version 3 solved it and added "//" double division option for people wanted to use division with out fraction
- For more information : https://www.python.org/dev/peps/pep-0238

Modulus operator

- “%” it’s quite famous to use modulus operator
- **9 % 3**
- **0**
- **9 % 2**
- **1**
- It will give the remainder.
- It is very handy when you developing algorithms like great common divisor algorithm : GCD_algorithm

Numeric Types

- Integer
- Floating-point
- Complex
- Booleans
- Built-in modules for numeric types

- Tip: type() function

Getting into Numeric

- 1234, -24, 999999999999 are integer type
- 1.23, 14e-10, .5 are float type
- 3+4j, 3.0+4.0j, 3j are complex type
- Boolean :
  - True, 1
  - False, 0
- There are some built-in functions to use for numeric types:
  - pow <- takes the power
  - abs <- gives the absolute of the number
  - round <- rounds the number
  - int <- makes the integer
- For further mathematical functions there is math library that we can use. (import math)

Strings

- String is everything between ‘ ’ or “ ”
- “ ” has no different from ‘ ’
- “”” will give us a change to assign
  Paragraphs in to the variable.”””
- We can concatenate our strings s1 + s2
- Repeat strings s1 * 3
- We can slice our strings in various ways
- We can get the length of our string
- Replacement with strings
- Removing whitespaces
- Split on delimiter
- Lower case conversion, and more…

String Basics

- ‘blueberries’
- “blueberries”
- Will give the same result.
- We can use this rule in :

  “Classes’ homework”
- Normally if you use same quotation mark it will cause a misconception but using different kinds works just fine.
- We can concatenate two strings or two variable contains strings.
- len() function gives the length of a string for instance or variable contains string:
  - len(“This is string!”) # important note: it counts the spaces as well

More Basics

- How to use multiple block string:

```
paragraph  = """ triple quotes are using as multi line block strings
        it is very useful when you have a lot to say
you can design your own paragraphs. Python will understand the spaces…
"""
```

- If we put our variable into the interpreter it will result as

```
' triple quotes are using as multi line block strings\n\tit is very useful when you have a lot to say \nyou can design your own paragraphs. Python will understand the spaces…'
```

- To avoid this we can print(paragraph)

I Will Slice Your String!

- Slicing is very handy when you are manipulation texts/strings.
- **s1 = “Watermelon”**
- Let’s slice this “Watermelon”

Indexing

s1[0] will result as ‘W’

s1[-2] will result as ‘o’

So our indexation starts from 0 but backwards it starts from -1

I want the water part of the “Watermelon”, what should I do?

Slicing

s1[:5] # : is slicing operator and it takes the last parameter minus one’s index

s2 = s1[:5] # “Water”

s3 = s1[5:] # “melon”

Going Deep on Strings

- There are a lot of methods you can manipulate strings with:
  - **We can change a string with s = s + “spam”**
  - **s = s.replace(‘x’, ‘y’)**
  - **s.capitalize()**
  - **s.lower()**
  - **s.join([parameters])**

- Let’s write some of them into Python interpreter.

To Do List

- Complete exercise #1
- **Find a project** idea about your passion.