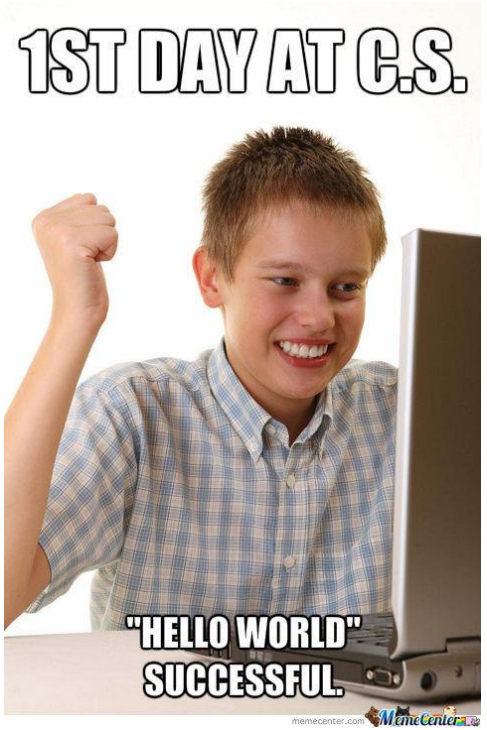
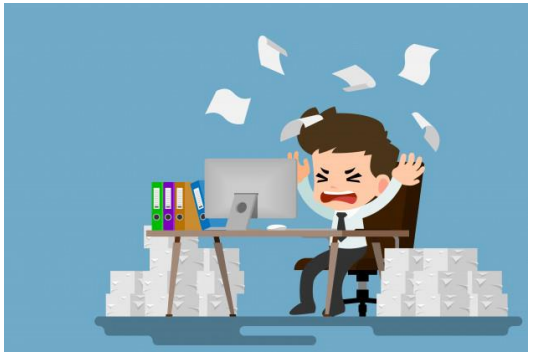
School Nova Computer Science



9/28/2019

By Oleg Smirnov

Work organization



- Organize your folders in a way that will make it easy for you to find your code. For example, create a Folder “CS101” (or “CS201”). Inside, create folders “Classwork” and “Homework”. You may also want to create additional folders organized by date.
- Always save your Python script using a file name that accurately describes the code. For example, Smirnov_homework2_version1.py.
- Do *NOT* accept the default filenames such as temp.py, untitled0.py, and so on.
- Save your Python script folder in the corresponding folder (for example, “Homework”), where you can easily find it later. You may need it later, or you may need to upload your file.

Coding, first tips, p1

- We *generally* follow the mainstream conventions as outlined in Python Enhancement Proposal (PEP): https://www.python.org/dev/peps/pep-0008/
- Add comments to your code:

```
# This is a comment. Python will ignore anything after the #
symbol. (Unless # is placed inside a string!)
```

- Comments can be very useful if you want to save information for other people, other programmers or your teacher, or even yourself at a later date.
- Maintain consistent style throughout your code. For example, use either ‘single-quoted strings’ or “double-quoted strings” throughout your code. Do not mix. I recommend using double-quoted strings.
- Limit lines to 79 characters. We will explore ways to wrap long lines.

Coding, first tips, p2

- For example, add spaces around math operations:

  write “a + b” instead of “a+b”

- Add a space after a comma, colon, semicolon (but note before):

  write “a, b = 1, 2” instead of “a,b=1,2”

- Avoid unnecessary spaces (or extra spaces at the end of a line):

  write “month = 9” instead of “month = 9 ”

- Name your variables, functions, classes, and so on in a way that makes intuitive sense. For example,

  write: name = “Oleg” instead of n = “Oleg”

- Use spaces instead of tabs, when writing code.

To Be Continued

Homework advice

- If a question requires a specific answer, use print(), for example:

  print(type(“my_string”))

- print() may also be used for longer answers or to print your clarification if something does not work, for example:

  print(“multiplying a string by a float gives me an error”)

- If you think your code needs clarification, you can add a comment. Write in proper English and complete sentences.
- Ideally, I should be able to execute your script without any errors. If you cannot fix a problem, you need to add a comment to your code with an explanation.
- If something does not work, do not give up. Do your best and submit whatever things your tried (even if they do not seem to work).

Objects and variables, p1

- In Python, when you create a variable, you actually create an **object** and a r**eference** to it: for example, *a* = 5, where 5 is an object and *a* is a reference to it (you can think of a reference as a name).
- If you create another variable *b* = 5, it will be pointing to the same object!
- In Python you don't need to declare what kind of an object it is. Python assigns a type to the object automatically based on the content. (Yes, it is that smart)
- In many cases, you can change the type of a variable if you want, in which case you will create a new object!
- If you change the value of a variable, you create a new object!

Objects and variables, p2

- id() shows the unique integer identification of any object: id(a).
- Try different examples to see that variable names are like stickers or labels.
- Are there objects without labels? Not for long. For example,

  a = 5 (creates an object with value 5 and reference *a*)

  a = 6 (creates a new object with value 6, the object above with value 5 becomes "garbage" since there is no reference to it)
- Python removes garbage to free memory ("garbage collection")
- *Note on variable names*: can only contain numbers, letters, and underscore ("_"); cannot start with a number; case sensitive.

Python Data Types, p1

name = “John Smith”	String
year = 2019	Integer
Pi = 3.14159	Float
Pi = “3.14159”	String
result = True result = False	Boolean
result = “True”	String

Python Data Types, p2

str()	Convert to string Almost anything can be converted Cannot convert variables which are not defined: str(a), a not defined Cannot convert math expressions which yield an error: str(10/0)
Int()	Convert to integer Can convert strings without decimal part: int('5') Cannot convert strings with decimal part: int('5.5') is an error Cannot convert strings that do not look like an integer: int('ok") error Can convert a Boolean and float (rounding down): int(True), int(5.5)
float()	Convert to a floating point number (number with a decimal part) Can convert a string that looks like a number: float('5.5') Can convert a Boolean and integer: float(True), float(5)
bool()	Converts to a Boolean, which is either True or False Most variables are seen as True, with a few exceptions such as bool(0)
type()	The result is the type of what is inside parenthesis (Returns the type of the argument)

Python Data Types, p3

String + String String * Integer String * Bool	String
Integer + Integer Integer * Integer	Integer
Integer / Integer (even if no remainder) Integer + Float Integer * Float	Float
String + Integer String + Float String + Bool String * Float String / Integer	Errors
Bool + Integer Bool + Float	Integer Float