School Nova

Computer Science

Nested dictionaries, zip()
User-defined functions: first look

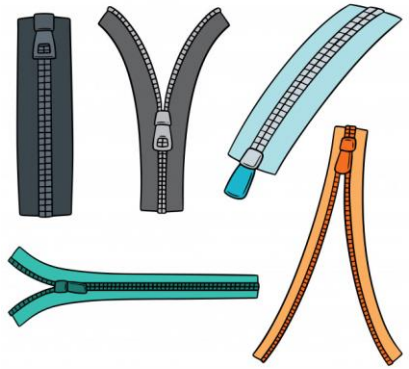
1/12/2020

By Oleg Smirnov

Nested dictionary

```
spring_months = {1:"March", 2:"April", 3:"May"}
summer_months = {1:"June", 2:"July", 3:"August"}
year = {"spring":spring_months, "summer":summer_months}

print(year["spring"][1])
>>> March

fall_months = {1:"September", 2:"October", 3:"November"}
year["fall"] = fall_months

print(year["fall"][3])
>>> November
```

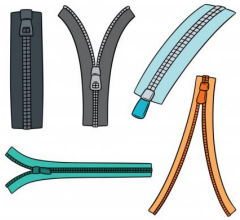
Dictionary from two lists



```
listA = ["France", "Italy", "Spain"]
listB = ["Paris", "Rome", "Madrid"]

R = dict()

for i in range(len(listA)):
    R[listA[i]] = listB[i]

# alternative approach
R = dict(zip(listA, listB))
```

You cannot create a dictionary if zipping more than two lists!
(We need two because we need a “key” and a “value”)

zip()



listA, listB = ["France", "Italy", "Spain"], ["Paris", "Rome", "Madrid"]

zip(listA, listB) creates a zip object, consisting of tuples:

```
print(zip(listA, listB))
>>> <zip object at 0x000001FD9FB14888>

print(list(zip(listA, listB)))
>>> [('France', 'Paris'), ('Italy', 'Rome'), ('Spain', 'Madrid')]

print(dict(zip(listA, listB)))
>>> {'France': 'Paris', 'Italy': 'Rome', 'Spain': 'Madrid'}
```

If the length of input lists is not equal, zip() discards the "extra" elements from the longer list.

User-defined functions INTRO

You can create your own functions in Python!

A function is a task (which is likely used more than once).
Python does not have functions for all possible tasks in the world.
BUT we can create as many user-defined functions as we want.

Why?

1) Avoid repeating the same lines of code again and again (and again). A function may consistent of multiple lines of code, which we do not want to repeat. At the same time, calling a function is usually a single line of code.
2) Functions allow us to examine parts of our code in isolation (and easily find those bugs!)
3) Functions (that are proven to work well) allow us to ignore on what's "inside" and focus on the more important and immediate problems

User-defined functions, part 1

```
# this function does something simple
def myfun():
    print("Hello! I hope you are having a nice day!")
for i in range(100): myfun() # and this is how we use this function

# this function returns an object
import datetime
def today():
    return(datetime.date.today())
print(f"Today is {today()}") # and this is how we use this function

# this functions accepts an argument and returns an object (value)
def sum_powers(x):
    result = x + x**2 + x**3
    return(result)
print(sum_powers(3)) # and this is how we use this function
```

User-defined functions, part 2

```
# this functions accepts two arguments and returns an object (value)
def sum_powers(x, y):
    result = 0
    for i in range(1, y + 1):
        result = result + x**i
    return(result)
print(sum_powers(2, 2)) # and this is how we use this function

# this function accepts any number of arguments and returns an object (value)
def product(*x):
    prod = 1
    for i in x:
        prod = prod * i
    return(prod)
print(product(5, 10, 20)) # and this is how we use this function
```