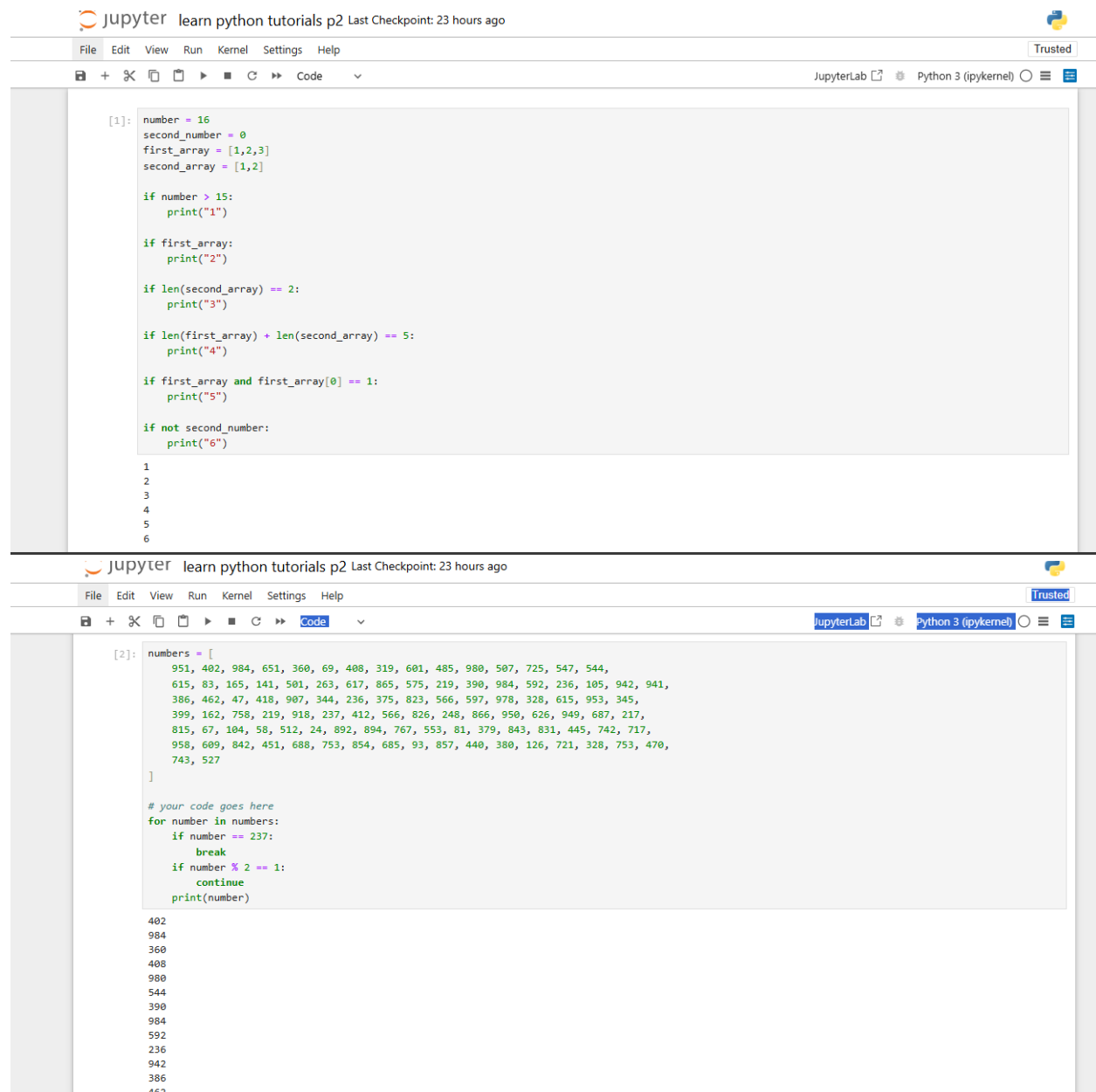


Learn Python Tutorials Part Two Python Code Jupyter Notebook

Lojain Idris



The image displays two screenshots of a Jupyter Notebook interface. The top screenshot shows a code cell with a series of conditional statements and variable assignments. The bottom screenshot shows a code cell with a list of numbers and a loop that prints specific numbers from the list.

Top Screenshot:

```
[1]: number = 16
second_number = 0
first_array = [1,2,3]
second_array = [1,2]

if number > 15:
    print("1")

if first_array:
    print("2")

if len(second_array) == 2:
    print("3")

if len(first_array) + len(second_array) == 5:
    print("4")

if first_array and first_array[0] == 1:
    print("5")

if not second_number:
    print("6")
```

Output: 1, 2, 3, 4, 5, 6

Bottom Screenshot:

```
[2]: numbers = [
    951, 402, 984, 651, 360, 69, 408, 319, 601, 485, 980, 507, 725, 547, 544,
    615, 83, 165, 141, 501, 263, 617, 865, 575, 219, 390, 984, 592, 236, 105, 942, 941,
    386, 462, 47, 418, 907, 344, 236, 375, 823, 566, 597, 978, 328, 615, 953, 345,
    399, 162, 758, 219, 918, 237, 412, 566, 826, 248, 866, 950, 626, 949, 687, 217,
    815, 67, 104, 58, 512, 24, 892, 894, 767, 553, 81, 379, 843, 831, 445, 742, 717,
    958, 609, 842, 451, 688, 753, 854, 685, 93, 857, 440, 380, 126, 721, 328, 753, 470,
    743, 527
]

# your code goes here
for number in numbers:
    if number == 237:
        break
    if number % 2 == 1:
        continue
    print(number)
```

Output: 402, 984, 360, 408, 980, 544, 390, 984, 592, 236, 942, 386, 462, 941, 345, 217, 717, 470, 527



```

592
236
942
386
462
418
344
236
566
978
328
162
758
918

[2]: def list_benefits():
      return "More organized code", "More readable code", "Easier code reuse", "Allowing programmers to share and connect code together"

      # Modify this function to concatenate to each benefit - " is a benefit of functions!"
      def build_sentence(benefit):
          return "%s is a benefit of functions!" % benefit

      def name_the_benefits_of_functions():
          list_of_benefits = list_benefits()
          for benefit in list_of_benefits:
              print(build_sentence(benefit))

      name_the_benefits_of_functions()

      More organized code is a benefit of functions!
      More readable code is a benefit of functions!

```

```

Easier code reuse is a benefit of functions!
Allowing programmers to share and connect code together is a benefit of functions!

[3]: class Vehicle:
      name = ""
      kind = "car"
      color = ""
      value = 100.00
      def description(self):
          desc_str = "%s is a %s %s worth $%.2f." % (self.name, self.color, self.kind, self.value)
          return desc_str

      # your code goes here
      car1 = Vehicle()
      car1.name = "Fer"
      car1.color = "red"
      car1.kind = "convertible"
      car1.value = 60000.00

      car2 = Vehicle()
      car2.name = "Jump"
      car2.color = "blue"
      car2.kind = "van"
      car2.value = 10000.00

      # test code
      print(car1.description())
      print(car2.description())

      Fer is a red convertible worth $60000.00.
      Jump is a blue van worth $10000.00.

```

```
[5]: import re

# Your code goes here
find_members = []
for member in dir(re):
    if "find" in member:
        find_members.append(member)

print(sorted(find_members))

['findall', 'finditer']
```