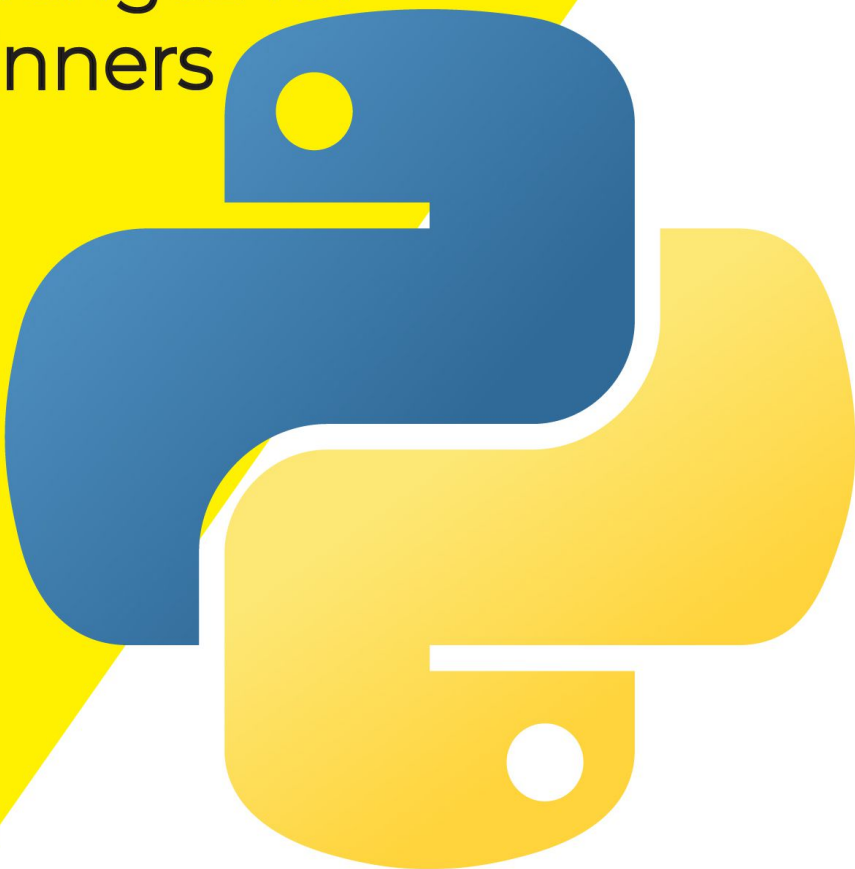
Practice PYTHON

Questions and Challenges for Beginners



Benjamin Bennett Alexander

Practice Python

Questions and Challenges For Beginners.

Benjamin Bennett Alexander

Feedback

I welcome and appreciate your feedback and reviews. Please leave reviews on the platform you purchased this book from. Please send your feedback and queries to: benjaminbennettalexander@gmail.com

Table of Contents

About this Book

This book has over 300 questions and challenges that will test your Python language knowledge as a beginner. It has both multiple-choice and open-ended questions. Learning to program is not just about reading books, but it's about practicing and writing code. **Problem-solving is one of the most effective ways to learn a programming language**. Whether you are new to Python or programming in general, or you just want to brush up on your Python skills, this book has enough questions and challenges to help you kickstart your Python career. Practice Python is about asking you questions and challenging you to find the answers. **Answers, explanations, and example codes are provided for all the questions and challenges**. You are also at liberty to go online and dig deeper into the various topics that we cover in this book.

In this book, we cover everything that you need to know as a Python beginner. Once you master these Python basics, you will have the skills and knowledge to go on to learn and build great things with Python. So, what are some of the things that will cover in this book?

- **Variables and Syntax**. Understanding the proper way to create variables and the set of rules for writing proper Python syntax is important for every beginner.
- We will cover Python data structures – **lists, tuples, sets, and dictionaries.**
- We will explore strings and various ways we can manipulate them using string methods.
- We will cover **Python operators** and **Python math.**
- Learning to organize code into **functions** is one of the most important skills that help programmers write clean code. Every Python beginner will become comfortable with **functions** after tackling questions and challenges in this book.
- We will explore conditional statements – **If, elif, and else statements.**

- Programming is about handling errors. We will explore how **errors** are handled in Python.
- We shall tackle **modules and libraries** in Python
- Python is an object-oriented programing language, so every beginner should know about **classes** and **inheritance.**
- We will also cover **dates, lambda functions** and so much more.

We will cover a lot of stuff that every Python beginner should know. If you can answer and solve the challenges in this book, then you have mastered the basics of Python and you are on your way to writing more complex Python code.

Downloading Python and IDE

You will need to have Python 3 installed on your machine to run the codes in this book successfully. If you do not have Python installed on your machine, you can download it here:

https://www.python.org/downloads/

Make sure that you download the latest version of python.

To write and execute code you will need a code editor or an Integrated Development Environment (IDE) to be installed on your machine. There are many code editors and IDEs out there that you can use. I recommend any of the following:

1. **Pycharm**
   Download the community version from the link below:
   https://www.jetbrains.com/pycharm/download/#section=windows

2. **Jupyter notebook**
   Use the link below to find out how you can install the Jupyter notebook on your machine.
   https://jupyter.org/install.html

3. **IDLE Shell**
   IDLE comes pre-installed with python for windows and mac, so you will not need to download it separately. When you click to open python, you will open the IDLE shell to write and execute code. Pretty easy to use for beginners.

4. **Sublime**
   You can download sublime text from the link below:
   https://www.sublimetext.com/download

5. **Atom**
   https://atom.en.uptodown.com/windows

You can explore any of these and settle for the one that you are comfortable with.

Syntax and Variables

1. Which of the following correctly defines a syntax in Python?
    a. It's a set of rules on how a Python code will be written and interpreted
    b. It's a set of commands found in Python
    c. It's a set of in-built functions and classes in Python programming
    d. All the reserved words found in Python

2. Which one of the following correctly describes a variable?
    a. A reserved memory location for an object
    b. A number in Python
    c. A string and number in Python
    d. A string in Python

3. You cannot start a Python variable with a capital letter. True or False
    a. True
    b. False

4. Mary writes the following code:

```
myname = "Mary"
```

From the above code, which one is a variable?
    a. Mary
    b. myname and "Mary
    c. myname
    d. None of the above

5.John is trying to create a variable name for a value called orange. Which one of the following codes will return an error?

a. fruit = "Orange"
b. Fruit = "Orange"
c. _fruit = "orange"
d. -fruit- = "Orange"

6.In Python, indentation is merely for readability only. It has no impact on how the code is interpreted. True or False?

a. True
b. False

7.Look at the code below:

```
x = 2
    print(x)
```

When this code is run it will generate an error. True or False?

a. True
b. False

8.Which one of the following is an improper name for a variable in Python?

a. seven
b. numberSeven
c. Number-seven
d. number_Seven

9.Mike writes down the following code:

```
9Myname = "Mike"
```

Mike runs the code and it generates a syntax error. What is wrong with the code?

a. The name Mike should not be in quotation marks
b. The name Mike should not start with a capital letter
c. The variable name should not start with a number
d. Myname should not be written as one word

10.Peter wants to create a string, so he writes the following code:

```
myname = Peter
print(myname)
```

Peter tries to run the code but it generates an error. What is wrong with the code?

a. The variable name should start with a small letter
b. The name Peter should start with a small letter
c. The name Peter should have been put in quotation marks
d. All the above

11.Which of the following variable names are not allowed in Python? Select all that apply.

a. My_name
b. my_name
c. My-name
d. _myname
e. My name

12. Which one of the following cannot be used as a variable name?
    a. False
    b. Name2
    c. Cars
    d. Car

13. John writes the following code:

```
list = 7
print(list)
```

Rose tells John that he cannot use ‘list’ as a variable name. John argues that he does not see the reason why he should not use it when the code runs just fine. Explain to John why Rose is right to insist that the variable name be changed.

14. Which one of the following is a reserved keyword in Python and cannot be used as a variable name?
    a. del
    b. jail
    c. may
    d. lay

15. Jane wants to use the word ‘print’ as a variable name. Can she use this name in Python?
    a. Yes, she can use it as a variable name as it is not a reserved word in Python
    b. print is a built-in function in Python so it should not be used
    c. She can use it because some Python reserved words can be used as variable names in some instances
    d. Yes, using the word will not generate an error

16. Kelly writes the following code:

```
fruit = "Mango"
print(Fruit)
```

The code generates an error ‘Fruit is not defined’. What is wrong with Kelly’s code?

    a. Variable names are case-sensitive so fruit is not equal to Fruit
    b. Mango should not be put in quotation marks
    c. There is no variable called fruit
    d. All the above.

17. Beyonce writes the following code:

```
Fruit = "Mango"
```

Which of the following is not equal to the code above? Select all that apply.

    a. Fruit =str('Mango')
    b. Fruit = "Mango"
    c. fruit = Mango
    d. Fruit =int(Mango)

18. John writes the following code:

```
a, b, c = "alphabet", "banana", "car"
```

Which one of the following codes will print “car”?

a. print(a)
b. print(b)
c. print(c)
d. print(:2)

19. Which one of the following will be interpreted as a comment in Python?

a. //This is a comment//
b. #This is a comment
c. \\This is a comment\\
d. **This is a comment**

20. John wants to write a code with a variable name car and its value should be Toyota. Which of the following will not generate an error? Select all that apply.

a. car = "Toyota"
b. car = 'Toyota'
c. car = Toyota
d. Car = Toyota

21. What is the difference between a statement and an expression in Python?
    a. A statement is a line of code that does something while an expression is a line of code that evaluates into something
    b. An expression is a line of code that does something while a statement is a line of code that evaluates into something
    c. A statement is a line of code that combines something while an expression is a line of code that divides something
    d. A statement is a line of code that runs all the time while an expression is a line of code that runs when called

22. Which one of the following is an expression?
    a. while loop
    b. print(name)
    c. ‘firstname’ + ‘lastname’
    d. if statement

Data Types and Typecasting

23. Which of the following is a data type in Python? Select all that apply.
    a. variable
    b. float
    c. integer
    d. complex

24. Which one of the following data types is an integer?
    a. 20
    b. 21.2
    c. 21.22
    d. 0.1

25. Mary writes the following code:

```
y = 1.5
print(type(y))
```

What data type do you get when you run this code?
    a. float
    b. int
    c. tuple
    d. list

26. John writes the following code:

```
name = "John"
print(type(name))
```

What data type do you get when you run this code?
    a. str
    b. float
    c. set
    d. tuple

27.

```
fruits = ["apples", "mangoes", "Oranges"]
print(type(fruits))
```

What data type do you get when you run this code?

a. dict
b. tuple
c. list
d. bool

28.

```
apples = {"apples", "mangoes", "Oranges"}
print(type(apples))
```

What data type do you get when you run this code?

a. list
b. dict
c. set
d. tuple

29.What data type do you get when you run the code below?

```
fruits = ("mango", "orange", "lemon")
print(type(fruits))
```

a. dict
b. mango
c. tuple
d. list

30.

```
cars = ["BMW", "Mazda", "Honda", False, 1,2,9,10]
print(type(cars))
```

What data type do you get when you run this code?

a. tuple
b. list
c. float
d. dict

31.Sam writes the following code:

```
y = False
print(type(y))
```

What data type do you get when you run this code?

a. Set
b. bool
c. dict
d. float

32.Michael writes the following code:

```
y = {"name": "Peter", "school": "yale"}
print(type(y))
```

What data type do you get when you run this code?

a. dict
b. tuple
c. list
d. let

33. Which one of the following best explains typecasting?
    a. Converting one data type into another using a Python function
    b. Typecasting is printing out Python data
    c. Converting numerical type data into a string
    d. Writing clean code in Python

34. Mary wants to apply typecasting to convert variable x, so she writes the following code:

```
x = 5
x = float(5)
print(x)
```

What is the output of the code?
    a. 5.0
    b. 5
    c. 5.1
    d. 4

35. Using typecasting, write a code that converts the following code into an integer.

```
x = "10"
```

Print out the data type of your new variable.

36. Using typecasting convert the following code into a string.

```
x = 10
```

Print out the data type of your new variable.

37. Convert the following code into a float using typecasting. Print out the data type of your new variable.

```
x = 23
```

38.John wants to convert a variable y which is an integer into a string. How does he convert an int variable into a string using typecasting?

a. y = string(y)
b. y = str(y)
c. y =float(y)
d. y = int(y)

39.In Python, casting is the process of converting a variable into another type. What is the output of the following code?

```
y = 4.67
y = int(4.67)
print(y)
```

a. 5
b. 4
c. 4.7
d. 5.7

Python Tuples

40. Alison wants to create a tuple with one element. Is it possible to create a tuple with one element in Python? If it is possible, create one.

41. Create a tuple with three items – Jane, Mary, Alice. Use the variable names for this tuple. Use the type() function to check the data type of your variable.

42. Write a code to print the name 'Alice' from your tuple (question 41).

43. Write a code to check the length of the tuple (question 41). Print out the length.

44. Your boss Jerry has asked you to add another item to your tuple (question 41). Write a code to add "Maria" to the tuple.

45. Your boss Jerry has come back and now wants the name Maria (question 41) removed from the tuple. Write a code to remove the name Maria from the tuple.

46. Your boss suspects that the name Jane appears more than once in the tuple of names. Use the tuple method count() to check how many times the name Jane appears in your tuple (question 41).

47. Which one of the following is a tuple method?
    a. cut()
    b. append()
    c. index()
    d. sort()

48. Which one of the following is true about tuples?
    a. No duplicates are allowed in a tuple
    b. You cannot create a tuple of Boolean data types
    c. You cannot have a tuple with different data types
    d. There is no limit to the number of items you can add to a tuple

49. There are two ways to create a tuple, using round brackets () and using a tuple constructor. Create a tuple of colors – Red, Blue, Green, using a tuple constructor. Print the tuple of colors.

50. What is the difference between a tuple and a set? Select all that apply.
    a. Tuples have ordered values while sets don't
    b. Sets have ordered values while tuples don't
    c. Sets are mutable while tuples are immutable
    d. Tuples are immutable while sets are mutable

51. Mary has written down the following codes:

```
x = (1, 2, 4, 6, 7, "John")

y = {2, 1, 4, 6, 6, 7}
```

Which one is a set and which one is a tuple?
    a. x is a tuple, is a set
    b. y is a tuple, x is a set
    c. x is a tuple, y is a tuple
    d. x is a set, y is a set

52. You can use the sort() method to sort a tuple in ascending order only. True or False?
    a. True
    b. False

53.Can you have duplicate items in a tuple?

a. No, you cannot have duplicates in a tuple
b. Yes, you can have duplicates in a tuple

54.John wants to know the length of the tuple below:

```
animals = ("dog", "cat", "duck", "chicken")
```

Which one of the following codes will give the length of the tuple?

a. print(len(animals))
b. print(len.animals)
c. print(len(animals)
d. print(animals.len)

55.John wants to change the last item in the tuple (question 54).
Which one of the following is the right code to update the tuple?

a. animals[3] = "cow"
b. animals[2] = "cow"
c. animals[:3] = "cow"
d. Tuples are immutable

56.Which one of the following codes will delete ‘dog’ from a tuple (question 54)?

a. animals. delete["dog"]
b. animals. delete("dog")
c. Impossible to delete an item in a tuple
d. animals.delete[dog]

57.Jane has written down the following codes:

```
animals = ("dog", "cat", "duck", "chicken")

fruits = ("oranges", "apples", "mangoes", "bananas")
```

Write a code to combine the two tuples.

58.Jane wants to double the code elements in the following tuple:

```
cars = ("BMW", "Isuzu", "Nissan")
```

Write a code to double each element in the tuple.

59.Which one of the following is a tuple method?

a. add()
b. append()
c. count()
d. delete()

Lists in Python

60. This data structure is mutable, its values can be stored between square brackets, and it can store different types of data. Which data structure is this?
    a. tuple
    b. dict
    c. list
    d. Boolean

61. Create a list of numbers 0 – 10. Give your list variable name - numbers. Print out the class/data type of your list.

62. Ashanti writes down the following code:

```
numbers = [10, 20, 40, 70, 60, 90, 100, 55, 78, 89]
```

    Write a code to print out numbers 70-60 from your list. Use list slicing.

63. Write another code to print out numbers 70 – 100 from the list (question 62).

64. Your boss has asked you to sort your list (question 62) in ascending order. Using the sort() method, write a code to sort your list. Print out the sorted list.

65. A sort() method sorts a list only in ascending order. True or False?
    a. True
    b. False

66.Which one of the following is not a list method?

a. sort()
b. copy()
c. return()
d. pop()

67.Jimmy writes the following code:

```
cars = ["Volvo", "Honda", "BMW",
        "Toyota", "Toyota"]
print(cars.count("Toyota"))
```

What is the outcome of running this code?

a. 1
b. 2
c. 3
d. 4

68.Mark has written down the following code:

```
cars = ["BMW", "Honda", "Nissan", "Toyota"]
```

Mark wants to print “Toyota” in the list. What code should Mark write?

a. print(cars[3:])
b. print(cars[4:])
c. print (cars[3:1])
d. print(cars[3])

69.Which one of the following is not a property of a list in Python?

a. It can hold duplicate items
b. It can be changed
c. It is immutable
d. It is ordered

70.Janet writes down the following code:

```
a = [1,2,4,4,5,6,7,8,9]
```

Janet wants to remove 4,4,5 from the list. Help Janet with a code that will remove these numbers. Print out the updated list.

71.Kate writes the following code:

```
a = [1,2,2,3,3,4,5,6,9,8]
```

Which of the codes below will return the length of the variable a?

a. print(len(a))
b. print(len(a)
c. print(a.len)
d. print(len.a))

72.Which one of the following is not a list method?

a. pop()
b. drop()
c. remove()
d. sort()

73.Pat writes the following code:

```
girls_names = list(("Mary", "Rose", "Kate", "Amanda"))
```

Pat wants to remove the last name from the list. Which one of the following codes will remove the name?

a. girls_names.remove()
b. girls_names.pop()
c. girls_names.sort()
d. girls_names.drop()

74. Which one of the following methods will add an item to the end of the list?

a. add()
b. append()
c. join()
d. insert()

75. What is the difference between append and extend methods?

a. extend adds a single item to the list, while append increases the list by the number of iterable items
b. append adds a single item to the list, while extend increase the list by the number of iterable items
c. There is no difference between the two
d. append is a string method while extend is a list method

76. Simon writes the following code:

```
cars = ["Volvo", "Honda", "BMW", "Toyota"]
```

Simon writes another code to replace some items in the list below.

```
cars[2:4] = ["Audi","Ford"]
```

Which cars in the original list will be replaced?

a. Volvo and Honda
b. BMW and Toyota
c. Honda and Toyota
d. Volvo and Toyota

77.

```
cars = ["Volvo", "Honda", "BMW", "Toyota"]
```

Which one of the following codes will replace Volvo with Benz?

a. cars[-1]= "Benz"
b. cars[1]= "Benz"
c. cars[2]= "Benz"
d. cars[-4]= "Benz"

78.Mary writes the following code:

```
cars = ["Volvo", "Honda", "BMW", "Toyota"]
print(cars.index("Volvo"))
```

What is the outcome of running this code?

a. 0
b. 1
c. 2
d. 3

79.Which one of the following is not a list method?

a. sort()
b. sorted()

80.Steve writes the following code:

```
x = [1, 3, 5, 7, 8, 9, 2, 4, 5, 7, 10]
```

Using the sorted() function, write a code to sort the list in ascending order. Create a new variable for the sorted list.

Python Sets

81. Which of the following is true about sets? Select all that apply.
    a. Sets are surrounded by square brackets
    b. Sets are ordered
    c. You can change sets after they are created
    d. Sets are surrounded by curly brackets

82. Create a set of three items called a car. The set should have 3 items Benz, Ford, and Toyota. Print out the car class type.

83. Mary is asking that you change your set (question 82). She wants you to replace Toyota with Honda. Write a code to replace the cars. Print out the code.

84. Peter writes the following code:

```
set1 = {10, 67, 90, 91, 77, 73}
set2 = {2, 1, 4, 6, 8, 7}
```

Peter wants to merge set2 elements to set1 to create one set. Write a code to merge the sets. Print out the set. (Hint: there is a set method you can use for this).

85. Write a code to remove the number 77 from the merged set you created (question 84). Use the set method. Print out the updated set.

86. Which one of the following is not a set method?
    a. remove()
    b. method()
    c. pop()
    d. union()

87. Write a code to clear all the elements of the following set. Print out the results.

```
v = {12, 90, 89, 23, 56}
```

88. Create a set of numbers 1-10 using the set() constructor. The variable name of the set should be num. Print out the num class type.

89. Write a code using len() to determine the length of your set (question 88).

90. You can add new items to a set. True or False?
    a. True
    b. False

91. What is the difference between the update() method and the union() method?

92. Which of the following is true?
    a. Sets can have duplicates
    b. Sets are ordered
    c. Sets have a similar bracket type to dictionaries
    d. You cannot put a set inside a set.

93. Write a code to add items from cars2 to cars below. Print out the combined set.

```
cars = {"Volvo", "Honda", "BMW", "Toyota"}

cars2 = {"Audi","Ford"}
```

Operators in Python

94. Which one of the following operators is used to add values together?
    a. *
    b. +
    c. –
    d. %

95.

```
x = 2**3

print(x)
```

What answer do we get when we run the code?
    a. 6
    b. 8
    c. 12
    d. 16

96.

```
y = 6 * 3

print(y*2)
```

What is the result of this expression?
    a. 11
    b. 12
    c. 36
    d. 40

97. Which one of the following is a division operator in Python?

a. //
b. /
c. %
d. \\

98.

```
x = 6//4

print(x*2)
```

What is the result of this code?

a. 6
b. 3
c. 4
d. 2

99.

```
x = 6/4

print(x*2)
```

What do you get when you run this code?

a. 1.5
b. 2.0
c. 3.0
d. 4.0

100. What is the difference between // operator and the / operator?

a. // operator return the result with decimal places, while / operator does not return decimal places
b. //operator returns the result without decimal places, while / operator returns the results with decimal places
c. // is a division operator while / is a modulus operator
d. There is no difference between the two operators

101. Which one of the following is the same as c = c + 5?

a. c =+ 5
b. c += 5
c. c - = 5
d. c = - 5

102. John has written down the following code:

```
b = 12%3

print(b)
```

What is the result of running this code?

a. 4
b. 0
c. 3
d. 12

103. Which one of the following is not a logical operator in Python?

a. and
b. or
c. not
d. equal

104. Tina writes and runs the following code:

```
a = 10 != 5

print(a)
```

What is the result of this code?

a. True
b. False
c. 5
d. It generates an error

105. What is the shorthand for y = y // 5?

a. y = y/5
b. y /=5
c. y //=5
d. y =//

Strings and Boolean Values

106. What is a string in Python?
    a. A sequence of words not surrounded by quotation marks
    b. A sequence of characters surrounded by quotation marks
    c. A sequence of numbers
    d. A sequence of numbers surrounded by square brackets

107. The strip() method and split() method do the same thing to a string. True or False?
    a. True
    b. False

108. Which one of the following variables will not return a class type of string?
    a. y = str(5)
    b. y = '5'
    c. y = "5"
    d. y = 5

109. Mary writes the following code:

```
y = "I am learning python"
```

Which one of the following codes will slice the word 'learning' from variable y?
    a. print(y[:5])
    b. print(y[6: 13])
    c. print(y[5:13])
    d. print(y[5:]

110.

```
y = "I am learning Python"
```

Which one of the following codes will slice the word ‘Python’ from the variable y?

a. print(y[-6:])
b. print(y[-5:])
c. print(y[-5:9])
d. print(y[-6:-9])

111.

```
y = "I am learning Python"
```

Write a code to replace ‘learning’ with ‘studying’ in the variable y.

112.Mark writes the following code:

```
fruits = "#mangoes, oranges#"
```

Write a code that removes the # characters from the code.

113.Dwayne writes the following code:

```
y = "I don’t like "lazy" people"
print(y)
```

Dwayne runs the code but it returns an error. What is wrong with the code?

a. He cannot use opening and closing quotes
b. He cannot use double quotes inside a string surrounded by double quotes
c. He cannot use capital letters inside the string
d. All the above.

114. Which of the following string methods will capitalize the first character in a string?
    a. capitalize()
    b. upper()
    c. casefold()
    d. capital()

115. Which one of the following string methods converts a string into a lower case?
    a. lower()
    b. small()
    c. casefold()
    d. all the above

116. Which one of the following string methods converts the whole string into an upper case?
    a. upper()
    b. capitalize()
    c. casefold()
    d. capital()

117. Derick is trying to figure out the length of a string. Which one of the following methods will give him what he is looking for?
    a. values()
    b. len()
    c. size()
    d. number()

118. Which one of the following escape characters tells Python to start a new line?
    a. \\
    b. \b
    c. \n
    d. n\

119.Karen writes the following code:

```
a = bool(a)
print(a)
```

What value will this code return?

a. Zero
b. True
c. False
d. Empty

120.What will the following code return when you run it?

```
a = 7 > 7-1
print(a)
```

a. True
b. False
c. Error
d. Nothing

121.Janet writes the following code:

```
home = "The place where I relax"
print(isinstance(home, str))
```

What will this code return?

a. False
b. True
c. Str
d. Int

Python Dictionary

122.Which one of the following is a dictionary?

a. student = {"name": "John", "age": 23}
a. student = ("name": "John", "age": 23)
b. student = ["name": "John", "age": 23
b. student =dict[name: John, age: 23]

123.Which one of the following is not a property of a dictionary?

a. Immutable
b. Ordered
c. No duplicate keys
d. None of the above

124.John wants to access the value of the following dictionary:

```
cars = {"name": "Ford", "year": 1992}
```

Which one of the following codes will print out the values Ford and 1992?

a. print(cars.values())
b. print(cars.keys())
c. print[(cars.values())]
d. print (Ford.1992)

125.Write a code to access the keys and values of the car's dictionary (question 124). Use a for loop.

126.John is trying to change the name of the car name (question 124) to BMW. Write a code that will change the value of the dictionary.

a. cars["name"]= "BMW"
b. cars["values"] = "BMW"
c. cars("name") = "BMW"
d. cars[values] = BMW

127. Help John with a code that will change the key from "name" to "brand" (question 124).
    a. Cars["values"] = "brand"
    b. cars ["keys"]= "brand"
    c. cars ["name"] = brand
    d. Keys cannot be changed

128. Write a code to add a key named “color” and its value of “white” to the dictionary (question 124).

129. Write a code to change the value 1992 to 2004 in the dictionary (question 124).

130. Which one of the following is a dictionary method?
    a. add()
    b. update()
    c. count()
    d. append()

131. Alison writes the following code:

```
cars = {"names": "Nissan", "year": 2007}
```

Alison wants to remove the key “year” from the dictionary. Write a code to remove the key from the dictionary.

132. Alison changed his mind and decides to delete the dictionary. Write a code that will delete the dictionary.

133. Mary writes 3 dictionaries:

```
student1 = {"name": "John", "grade": "F"}
student2 = {"name": "Alison", "grade": "B"}
student3 = {"name": "Ben", "grade": "C"}
```

Write a code that will combine these dictionaries into one nested dictionary. Call the dictionary – students.

134. Mary decides to remove student3 from the nested dictionary. Help her with a code that will remove the student3 from the nested dictionary (question 133).

135. Which one of the following methods removes all the elements from a dict?

a. remove()
b. pop()
c. popitem()
d. clear()

136. Here is a dictionary:

```
mydict = {"name": "Mary", "occupation": "Singer"}
```

Write a code to create a copy of this dict.

a. newdict = mydict.copy
b. newdict = mydict.copy()
c. newdict = mydict.copy[]
d. newdict = copy(mydict)

137. Using the get() method write a code to access the occupation value("singer") (question 136).

138.Which one of the following is not a dict method?

a. popitem()
b. delete()
c. pop()
d. setdefault()

139. Which one of the following is true about dictionaries?

a. Dictionaries allow duplicate keys
b. Dictionaries allow duplicate values
c. Dictionaries do not allow duplicate values and keys
d. Dictionaries allow duplicate keys and values

140.What is the main difference between a dict and a list?

a. Items in a list are accessed by keys, while items in a dict are accessed by values.
b. Items in a dict are accessed by keys while items in a list are accessed by referencing their position
c. Items in a list are not ordered while items in a dict are ordered
d. None of the above

141.Jane writes the following code:

```
x = {"name": "carol", "age": 23, "school": "Yale"}
```

Write a code to access the school Yale from the dictionary.

142.Which of the following is a true similarity between a dictionary and a list?

a. They are both immutable
b. They are both mutable
c. They are both sequential
d. They are both surrounded by square brackets

143. What is the difference between a dictionary and a tuple?
    a. Dictionary is mutable while a tuple is immutable
    b. Dictionary is immutable while a tuple is mutable
    c. Dictionary is ordered, while a tuple is not ordered
    d. There is no difference between a dictionary and a tuple

144. Write a for loop to print all the values in the following dictionary.

```
x = {"name": "carol", "age": 23, "school": "Yale"}
```

145. Write a code to loop through all the keys in the dictionary (question 144).

146. Write a code to check if the key “name” is in the dictionary x. If the key is in the dictionary, let the code print ‘name is one of the keys ’. If it is not in the dictionary, the code should say ‘name is not one of the keys’ (question 144).

147. John writes the following code:

```
a = {'name': 'John', 'age': 25}
b = {'school': 'Yale', 'location': 'USA'}
```

Write a code to combine these two dictionaries using a bitwise operator.

Functions in Python

148.Which one of the following correctly defines a function?

a. A block of code that performs a specific task and runs all the time
b. A block of code that performs a specific task and runs when called
c. A block of code that loops a function
d. A block of code that runs non-stop and only stop when a break is called

149.Which one of the following is a keyword for creating a function?

a. function
b. def
c. fun
d. dif

150.Which one of the following is a valid way of creating a function?

a. def my_car:
b. def my_car():
c. def my car:
d. def_my_car():

151.What are the arguments in a function?

a. Values that are passed into a function when it is called
b. Information that is left out of a function
c. The name of the function
d. None of the above

152. John wants to create a function named car with the parameter called carname. Which one of the following is a valid function?
    a. def_car (carname):
    b. def car (carname):
    c. def car ("carname"):
    d. Def car("carname"):

153. Create a function called add that takes two parameters number1, number1. Return the sum and print out the answer.

154. John wants to create a function called my_function that will have one parameter, car, and a default argument, BMW. Which one of the following correctly creates the function?
    a. def my_function (car_BMW):
    b. def my_function ("car_BMW"):
    c. def my_function (car = "BMW"):
    d. def_myfunction (car =BMW):

155. What is the purpose of the return statement in a function?
    a. To tell the function to return the results of the function
    b. To replace print
    c. To print the function results
    d. To delete the function results

156. Which of the following is an advantage of a function? Select all that apply.
    a. With functions, you don't have to repeat code
    b. It makes the code clean, making it more readable
    c. It keeps the code hidden until we run the particular function
    d. All the above

157. Write a function called MultiplyAndDivide with three parameters a, b and c. The function multiplies the two parameters a and b and divides it by c. Create a local variable for the function (variable name = results). Pass 12, 24, 8 as arguments of the function. Your function should return 36.0. Print the results.

158. Which one of the following is true?
    a. A function cannot accept zero parameters
    b. A function name can also include reserved Python keywords
    c. A function can accept zero parameters
    d. A function does not accept a local variable

159. Mary told John to create one function as it is possible to use different arguments with a function every time it is called. Is Mary right that we can use different arguments with a function every time it is called?
    a. She's not right, we cannot use different arguments for one function
    b. She's right, we can use different arguments for one function
    c. We can use different variables but not different inputs
    d. A function does accept different types of inputs.

160. Debra wants to write a function, but she is not sure how many arguments the function should have. How can she properly write this function?
    a. Pass ** as the argument of the function
    b. Pass *args as the argument of the function
    c. Write a function without arguments
    d. She must know the actual number of arguments before writing the function

161. Is there a limit to the number of parameters that a function can take?
    a. A function cannot take over 10 parameters
    b. A function cannot take over 5 parameters
    c. A function cannot take over 2 parameters
    d. There is no limit to the number of parameters a function can take

162. Debra creates the following function:

```
def subtract(num1,num2):
sub1 = num1 - num2
return sub1
print(subtract(24,26))
```

The function generates an error. What is wrong with this function?

    a. The sub1 variable should have started with a capital letter
    b. The sub1 and return statement should have been indented
    c. The print statement should have been outside the function
    d. The function is missing the return call

163. How do you use a variable that is inside a function outside a function?
    a. You cannot use a variable outside a function in a function
    b. Declare the variable, a global variable
    c. Change the name of the variable outside the function to global
    d. Use a keyword in the name of the function

164.Jane writes the following function:

```
def my_fun()
        sum1 = 11 + 11
        return sum1

print(my_fun())
```

This function generates an error. What is wrong with this function?

a. There is nothing wrong with the function
b. : is missing at the end of the function name
c. The function should not be indented
d. sum is a reserved word in a function

165.What is a local variable when it comes to functions?

a. A variable outside a function
b. A variable declared inside a function
c. A variable imported into the function
d. None of the above

166.Karen writes the following code:

```
def subtract(num1, num2):
        sub1 = num1 - num2
        return sub1
```

Karen wants to call and print this function. The value of num1 is 10, and num2 is 5. Which of the following will call and print the function?

a. print(10,5)
b. print(10 - 5)
c. print(subtract(10,5))
d. subtract(10,5))

167. Ken has created the following function:

```
def my_function():
```

Ken does not know yet what to add to the function. He wants to keep the function empty. Now when he runs the function, he gets an error. What should he add to the function to avoid the error?

a. Add the keyword pass to the function
b. Add the keyword return to the function
c. Add the keyword global to the function
d. Add the word the keyword del to the function

168. Write a function called convert that takes an argument km and converts it to miles. Convert 109kms to miles and print out the answer. Round the answer to zero decimal points.

169. Write a function called rectangle_area, which takes two parameters a and b, calculates the area of a rectangle, and multiplies it by 2. Return the answer and print it out (a = 15, b = 67).

170. Write a function that calculates the area of a triangle. The function takes two parameters, base, and height. Name the function area_of_triangle. Pass **24(base)** and **67(height)** as function arguments.

Conditional Statements

171. Create a function called max_num with one parameter, a. If a is less than 10, it should print 'The number is less than 10'. Else it should print 'The number is not less than 10'. Pass the number 12 as an argument for the function.

172. Mika writes the following code:

```
a = 11
b = 10

if a != b:
print("it's not equal")
```

The code generates an error. What is wrong with the code?

a. The if statement should be in a function
b. The print statement is not indented
c. "it's not equal" should not be in inverted commas
d. The print statement should not be in round brackets

173. Which one of the following comes after the if statement in a code with more than one conditional statement?

a. else statement
b. elif statement
c. or statement
d. and statement

174. Write a code (conditional statement) that compares two variables a and b. Variable a = 10, and variable b = 20. If a is greater than b, it should return "a is greater than b". If b is greater than a, return "b is greater than a". if the two numbers are equal, return "The two numbers are equal". (a = 10, b = 20).

175. Mikel writes the following code:

a = 22

b =34

Write an if statement that prints out “B is superior to A” if b is greater than a. Otherwise print out “A is superior to B” if a is greater than b.

176. Which one of the following reserved keywords is used in conditional statements?

a. and
b. more
c. greater
d. than

177. Write the following code:

If a is greater than b and c is greater than b print out “Life is not fair”, otherwise print ‘Life is beautiful. (a =12, b= 15, c = 18).

178. Write the following into code:

If a is greater than b or b is greater than c prints out “Life is good”. Otherwise, print ‘Life is great ’(a = 22, b = 34, c = 56).

179. Write the following into code. Put everything in one line (shorthand syntax). If a is greater than b or a is greater c, print out “Life is good”.

180. Which one of the following is a reserved keyword used in conditional statements?

a. less than
b. or
c. more
d. less

For Loop

181. Which of the following statement best explains a for loop?
    a. A loop that iterates over a function
    b. A loop that iterates over an if statement
    c. A loop that iterates over a given sequence of items
    d. A loop that iterates over a string

182. William writes the following code:

```
a = "string"
```

Write a for loop that iterates through the variable a and prints out all the letters.

183. Peter writes the following code:

```
animals = ("dog", "cat", "duck", "chicken")
```

Write a for loop that iterates through the tuple and prints out all the items.

184. Which one of the following is used to stop the iteration of a for loop?
    a. halt
    b. return
    c. break
    d. cancel

185. Ken writes the following code:

```
cars = ["Volvo", "Honda", "BMW", "Toyota","Toyota"]
```

Using a for loop and if statement, write a code that iterates through the list. The code should break after it prints “Honda”.

186. Which one of the following is a reserved keyword used in a for loop to stop and continue the iteration?

a. break
b. continue
c. proceed
d. next

187. Which of the following best explains the continue statement in a for loop?

a. To cancel the iteration
b. To stop the current iteration and begin a new one
c. It has no impact on the iteration
d. It breaks the iteration

188. Mary writes the following code:

```
cars = ["Volvo", "Honda", "BMW",
        "Toyota", "Toyota"]
```

Mary wants to iterate through the code, but she wants to skip "BMW". Using a for loop and if statement, write a code that will print everything in the list except BMW.

189. What is the result of running the following code?

```
for x in range (10):
    if x == 5:
        break
    print(x)
```

a. 1,2,3,4
b. 0,1,2,3,4,5
c. 0,1,2,3,4,5,6,7,8,9
d. 0,1,2,3,4

190.What is the result of running the following code?

```
for x in range(10):
        if x == 5:
            continue
        print(x)
```

a. 1,2,3,4,5,6,7,8,9,10
b. 0,1,2,3,4,5,6,7,8,9,10
c. 0,1,2,3,4,6,7,8,9,10
d. 0,1,2,3,4,6,7,8,9

191.Ezil writes the following code for his project.

```
students ={"Student1": "Mary", "student2":
          "John","student3":"Peter"}
for x in students.values():
    print(x)
```

What is the result of the following code?

a. Mary, John, Peter
b. Student1, student2
c. John, Peter
d. Student1, student3, student3

192.John writes the following code:

```
x = "hello"
for letter in x:
    if letter== "e":
        continue
    print(letter)
```

What is the result of running the code?

a. hello
b. hllo
c. ello
d. llo

193.What is the result of running the following code?

```
colors = ["Red", "Green", "Blue"]
    for x in colors:
      if x == "Blue":
        break
      print(x)
```

a. Green
b. Green, blue
c. Red, Green
d. Red, Green, Blue

194.Which of the following is the result of running this code?

```
colors = ["red", "green", "blue"]
    for x in colors:
        print(x)
        if x == "green":
          break
```

a. red, blue
b. red, green
c. green
d. red, green, blue

195.Which of the following is the result of running the code below?

```
colors = ["red", "green", "blue"]
    for x in colors:
        print(x)
        if x == "green":
          continue
```

a. red, green, blue
b. green, blue
c. red, green
d. red, blue, green

196.Mary writes the following code:

```
for x in range (2, 16, 8):
      print(x)
```

What is the result of running the code?

a. 2, 3,4,5,6,7,8,9,10,11,12,13,14,15
b. 2,4,6,8
c. 2,3,4,5,6,7
d. 2,10

197.What is the result of running this code?

```
for x in range(2, 16, 4):
   print(x)
```

a. 2,6,10,14
b. 2,8,10,14
c. 4,8,10,14
d. 2,16,4

198.What is a nested for loop?

a. A big loop
b. A loop within another loop
c. A small loop
d. None of the above

199.John writes the following codes:

```
a = [ "yellow", "green", "red"]

b = [1,2,3,4]
```

Write a nested for loop for variables a and b. Print out the colors and numbers.

200.Which one of the following is false?

a. You cannot use a for loop in a function
b. You can use a for loop inside and outside a function
c. A for loop can be used to iterate through a set
d. All the above.

While Loops

201. What is the difference between a while loop and a for loop?
    a. A for loop iterates through a given number of items or range of objects, while a while loop iterates through items or objects as long as the condition is true
    b. A while loop iterates through a given number of items or range of objects, while a for loop iterates through items or objects as long as the condition is true
    c. A for loop can be stopped while a while loop will iterate forever
    d. There is no difference between the two

202. Lex writes the following code:

```
a = 1

while a < 10:
        print("This number is less than 10")
```

Maria warns Lex that if he runs this code, the code will run forever. Is maria right?
    a. No
    b. Yes

203. Write a while loop that prints out "This number is less than 10" five (5) times while the variable a is less than 10. (a = 1). Remember to increment the variable a by 2 for every iteration.

204. How can you stop the while loop? Select all that apply
    a. The while loop will stop when a condition is false
    b. You can stop the while loop by using the break statement
    c. You cannot stop a while loop
    d. A while loop will stop when the computer runs out of memory

205.Ben writes the following loop:

```
a = -1
while a < 5:
  a += 1
  if a == 3:
    continue
  print(a)
```

Which of the following is the output of this code?

a. 1,2,3,4,5,6
b. 0,1,2,3,4,5
c. 0,1,2,4,5,
d. 0,1,2,3,4,5,6,7

206.What is the outcome of the following code?

```
a = -1
while a < 5:
  a += 1
  if a == 3:
    break
  print(a)
```

a. -1, 2,3
b. 0,1
c. -1, 2,3
d. 0,1,2

207.Write a while loop that prints out “This number is less than 5”, while variable a is less than 5. If a is no longer less than 5, it should print “End of the loop”. Remember to increment a. (a =1).

208. John wants to use a while loop to iterate through the dictionary. Peter said he can't use a while loop to iterate through a dictionary? Is Peter, right?
    a. John is right, you can use a while loop to iterate through the dictionary
    b. Peter is right, you cannot use a while loop to iterate through the dictionary

209. Can you use a for loop inside of a while loop?

210. Which one of the following is true?
    a. If statement can only be used with while loops and not for loops
    b. If statement can be used with while loops and for loops
    c. If statements can only be used with for loops and not with while loops
    d. All the above is not true

Python Math

211.Janet writes the following codes:

a = 23

b = 45

Write a function called add that adds these two variables and returns the sum.

212.Write a function called multiply that multiplies the variables (question 210) and return the answer.

213.Kevin writes the following codes:

a = 45

b = 66

c = 12

Write a function that multiplies a by b and divides that with c. Print out the answer. Call the function multiply_and_divide.

214.Jane writes the following code:

```
x = (10,20,21,23,24,56,78,89)
```

Using min, write a code that finds the min of x. Create a variable for the min of x. Print out the answer.

215.Using a for loop and if statement, write a code that finds the max of x (question 213). Print out the answer.

216.Which one of the following is not a math method?

a. math.floor()
b. math.fabs()
c. math.divide()
d. math.comb()

217. Write a function that finds the square root of the variable below. Print out the answer.
x = 64 (Use a math method.Import math)

218. Jane wants to know whether a given variable is a number. Which of the following should she use?

a. math.isinf()
b. math.isnan()
c. math.isqrt()
d. math.gcd()

219. John wants to write a code that will return an absolute positive number of the following code:

```
a = -5.91
```

Write a code that will return the absolute value of a variable using a math method.

220. What is the result of the following code?

```
z = math.floor(23.3)
print(z)
```

a. 22
b. 23
c. 21
d. 20

221. What is the result of the following code?

```
a = math.ceil(23.4)
print(a)
```

a. 24
b. 23
c. 22
d. 21

222.What is the result of the following code?

```
p = max (12,13,16)
s = 23
print(p*s)
```

a. 368
b. 276
c. 299
d. 943

223.John writes the following code:

```
c = 16 % 8
print(c)
```

a. 2
b. 0
c. 8
d. 4

224.Mary writes the following code:

```
a = 2 * 2* 2 * 2* 2
```

Jane suggests that for a cleaner code Mary should use exponentiation. Rewrite the code using exponentiation.

225.Write a code that finds a remainder when you divide a by b. Print out the results.

a = 89
b = 9

226.Mary writes the following code:

```
a = 5 * 5 * 5 * 5
```

Mark suggests that Mary use the math method, pow() instead, for a cleaner code. Use pow() to rewrite this code? Print the answer.

227. What is the result of the following code?

```
a = math.trunc(18.9)
print(a)
```

a. 12
b. 13
c. 18
d. 19

228.What is the result of the following code?

```
x = pow(5, 3, 8)
print(x)
```

a. 240
b. 9.6
c. 3.75
d. 5

229.Which of the following is the outcome of the following code?

```
x = math.pi
y = 2*x
print(y)
```

a. 6.283
b. 3.141
c. 4.141
d. 2.223

230.Jane writes the following code:

```
x = min(12, 23,45,78, 90)
y = max(12,90,89,23,56)
v = (x,y)
z = math.prod(v)
print(z)
```

a. 455
b. 45
c. 121
d. 1080

User Input

231. Which one of the following methods is used to get input from the user?
    a. request()
    b. input()
    c. enter()
    d. include()

232. Write a code that asks a user to enter their name. The variable of the code should be my_name.

233. Write a function called adds that adds two variables a and b. The code should request the user to input a and b. Remember to convert the user input into integer using typecasting. Print out the answer. The user inputs you should enter once prompted by the code should be, a = 45, b = 30. Your code should print out a final answer of 75.

234. Write a function called multiply that multiplies variable a by b and divides it with c. The function should request the users for input a, b and c. (a = 45, b = 60, c =15).

235. Write a function that asks people for their age. Name the function your_age. Print out their age as "Your age is + their age + years old".

Python Variable Scope

236.What is a global variable?

a. A variable inside a function that can only be accessed in a function
b. A variable that can be accessed within the local and global scope
c. A variable that can be changed
d. A variable that is created within a function

237.Janet writes the following code:

```
def myfunction():
    c = 200
    return c
myfunction()
```

Janet wants to make the variable c into a global variable so it can be accessed outside the function. What change should she make to the function?

a. Create the variable outside the function
b. Use the global keyword within the function to declare c a global variable
c. Use the global keyword outside the function to declare c a global variable
d. Delete the variable c and put it outside the function.

238. What is a local variable?

a. A variable that is declared inside a function and can only be used in that function
b. A variable that is declared outside a function and can only be used outside a function
c. A neutral variable
d. A variable that cannot be modified

239. Any variable declared outside a function is a global variable. True or False?
    a. True
    b. False

240. A keyword “local” must be used with a variable in a function to make it local. True or False?
    a. True
    b. False

241. Any variable declared inside a function is automatically a global variable. True or false?
    a. True
    b. False

Error Handling

242. Which one of the following reserved keywords is used in error handling?
    a. del
    b. try
    c. False
    d. True

243. All errors will crash the program if not handled. True or False?
    a. True
    b. False

244. Which one of the following correctly defines the purpose of the keyword except?
    a. It deletes errors in the try block code
    b. It handles the errors raised in the try block
    c. It crashes the code when an error is discovered
    d. It discovers errors in the code

245. Which one of the following keywords is used to raise exceptions?
    a. raise
    b. finally
    c. error
    d. try

246. Which one of the following breaks down the finally keyword?
    a. A code that runs only when there is no exception in the code
    b. A code that runs after a try block whether the exception is handled or not
    c. A code that deletes errors that are handled in the except block of code.
    d. A code that deletes the error raised in the try block of code

247. Jane writes the following code:

```
print(x)
```

The code generates an error and crashes the program because x is not defined. Using the keywords try and except, write a code that will handle the NameError generated by this code. The code should print "x is not defined" when it encounters a NameError.

248. Mary writes the following code:

```
a = int(input("Enter a number"))
b = int(input ("Enter another number"))
print(a/b)
```

Mary is concerned that the program will crash if someone tries to divide a number by zero (0). Using try and except, ensure that the code can handle the ZeroDivisionError. When a ZeroDivisionError error occurs, it should print out "You cannot divide a number by 0. Try again".

249. Add a finally block to your code (question 248). The finally block should print "End of code".

250. Modify your code (question 249), so that it also can handle 'ValueError'. If someone enters something that is not a number, print 'You have not entered a valid number. Try again '.

251. For a cleaner code, your boss has asked you to put your code (question 250) in a function. Create a function called divide for your code. Use a while loop inside the function to ensure that the function runs until it returns the results of the expression (the division of a/b).

252. Wendy wants to write a code to get a user's age. She is worried that people may enter invalid numbers like 0 or numbers above 150 for age. Help her with a function called your_age that will ask people to enter their age. The function should be able to handle invalid inputs. Use the variable name – age, for inputs from users. If a person enters a number less than 1, the code should print 'You cannot enter a number that is less than 1. Please try again '. If someone enters a number that is above 150, the code should say 'You cannot enter a number above 150. Please try again '. If someone enters a valid age, it should print "Your age is: + age + years old". Hint: Use a loop and conditional statement to handle the error.

Pip and Modules

253.What is a module in Python?

a. A function and statement in Python
b. A Python file that contains Python code
c. A Python library
d. Any Python code is a module

254.Which of the following is the correct way to name a module?

a. add .py at the beginning of the name
a. add .py at the end of the name
b. add _py at the end of the name
c. add py_ at the end of the name

255.Mary has created a function file named add.py. She wants to use this function in a new file. Write a code to import this module into her new file.

256. What is the difference between a module and a library?

a. A module is a collection of files with different functionalities, while a library is a single file with single functionality
b. There is no difference between a module and a library
c. A library is a collection of files with different functionalities, while a module is a file with single functionality and statement
d. A module is a function while a library is a statement

257.What is pip in Python?

a. A module in Python
b. A library in Python
c. A package manager in Python
d. A function in Python

258. Steward has found a library called pandas online and he wants to use it in his data science project. How does Steward install this library on his machine using pip? Do a little research for Steward.

259. Sean writes the following code:

```
def add(x,y):
    sum = x + y
    return sum
```

Run this code and save it as add.py. Once it's saved, open another python file and import the module you have saved. Use the module to find the sum of:
x = 23
y = 12
Print out the answer.

260. Jane wants to import only a certain function from a module called add.py. She wants to import the sum function from the module. Which is the correct syntax for importing this sum function?

a. import sum
b. from add import sum
c. import sum from add
d. import add import sum

261. Dwayne wants to rename a module he wants to use in his file. He wants to shorten the name. The name of the module is myfunction.py. Dwayne wants to shorten the name to mf. Which of the following is the correct way Dwayne can import and shorten the name of the module?

a. import mf
b. from myfunction import mf
c. import myfunction as mf
d. import mf as myfunction

262. Which one of the following functions is used to check the attributes and methods in a module?

a. dir()
b. dy()
c. bin()
d. sin()

Dates in Python

263. Write a code to import time, date, and datetime from the datetime module.

264. Using the datetime module, write a code that prints out your current time (Hour: Minutes: Seconds as format). (Use the now() function and its format attribute). Create a variable called x for your time.

265. Using datetime module and now() function and its format attribute, print today's date, just date. Create a variable called x for the date. Print date in this format - month: date: year.

266. Using the datetime module and today() function, create a variable x and print your current time. (Use this format - Hour: Minute)

267. Using the date module and the today() function, create a variable x and print the current date.

268. Which one of the following datetime methods is used to format time?

    a. strptime()
    b. strftime()
    c. fromtimestamp()
    d. timestamp()

269. This is Jack's code:

```
import datetime
y = datetime.datetime(2021, 7, 1)
print(y.strftime("%G"))
```

What is the result of running this code by Jack?

a. July
b. 2021
c. 1
d. 2020

270. Jack makes a little change to the code:

```
import datetime

y = datetime.datetime(2021, 7, 1)

print(y.strftime("%A"))
```

What is the result of running this code?

a. Thursday
b. Wednesday
c. 2021
d. July

271. Challenge. Write a function called age that calculates a user's age. The function should ask the user to enter their year of birth and it should return their age. The function should print 'You are:' + user's age +. 'years old.'

Lambda Functions

272. What is a lambda function?
    a. A name of a function
    b. An expression that has no particular meaning
    c. An anonymous function or a function without a name
    d. All functions with names are lambda functions

273. Which one of the following is not a characteristic of a lambda function?
    a. It can only take a limited number of arguments
    b. It can only have one expression
    c. It can be used in another function
    d. It can take an unlimited number of arguments

274. Martha is using a lambda function in her code below:

```
v = lambda a,b,c: a*b*c*2
print(v(12,3,6))
```

What is the result of running this code?
    a. 216
    b. 432
    c. 108
    d. 864

275. Create a lambda function, variable name x, that takes two arguments a and b. It adds the two arguments and multiplies them by 2. Print out the result of the code. (a = 10, b = 15). Your results should be 50.

276. What is the result of running this code?

```
v = lambda a:a/2
print(int(v(12)))
```

a. 6.0
b. 12
c. 0
d. 6

277. What is the result of running this code?

```
v = lambda a:a%2
print(int(v(12)))
```

a. 6
b. 6.0
c. 0
d. 4

278.Kelvin has written a code that uses a lambda function inside a function:

```
def divide(num):
    return lambda y : y / num

 divide1 = divide(5)
 print(divide1(20))
```

What is the result of this code?

a. 0.25
b. 4.0
c. 5
d. 6

279. Modify the code (question 278). Change the value of y to 50 and run the code. What is the new output?

280. Here is a list of exam results of students at a certain school:

```
results = [ 32, 45, 78, 67, 88, 97, 76, 84, 44, 66, 63, 86, 90, 80]
```

John wants the list filtered so he can know how many students got over 60 marks. Using the filter() function and lambda function, write a code that will show John how many students got over 60 marks.

281. Write another code for John (question 280) that will show how many students got less than 50 marks.

Python Classes

282. Python is usually referred to as an object-oriented programming language. What does that mean?
    a. It means everything in Python is a function with its own properties
    b. It is structured into objects of similar properties and behaviors
    c. It is not structured into objects of similar properties and behaviors
    d. It means it's an easy language to learn

283. What does it mean to create a class in Python?
    a. It means to create a function
    b. It means to create an object
    c. It means to create a variable
    d. It means to create a database for your code

284. Jane wants to understand what an instance of a class is, so she comes to you for help. Explain to Jane what an instance of a class is.

285. What is the correct syntax for creating a class called car?
    a. class Car:
    b. (class car):
    c. class car():
    d. class_car:

286.What is the correct syntax to print out the variable i in the code below?

```
class Cat:
  i = 2
```

a. print(cat)
b. print(Cat)
c. print(i)
d. print(Cat.i)

287. What is the purpose of the constructor (__init__) function?

a. To create a class
b. To create an instance for a class
c. To assign values to class members
d. To delete class objects

288.Which one of the following is a valid way to create a class named Car with parameters - brand and model?

a.

```
class Car:
    def __init__(self,brand, model):
```

b.

```
class Car():
    def __init__(self,brand, model):
```

c.

```
class car:
    def __init__(self,brand, model)
```

d.

```
class Car:
    def (self, brand, model)
```

289. Which of the following is the correct way to assign values to class objects?

a.

```
class Car:
        def __init__(self,brand, model):
            self = model
            self = model
```

b.

```
class Car:
        def __init__(self,brand, model):
            self.(brand) = model
            self.(model) = model
```

c.

```
class Car:
        def __init__(self,brand, model):
            self(brand) = model
            self(model) = model
```

d.

```
class Car:
        def __init__(self,brand, model):
            self.brand = brand
            self.model = model
```

290. Continue with the code (from questions 288 and 289). Create an object called car1. The brand of the car is Nissan and the model is Murano. Print the model of the car1 object.

291. Create a second object (question 290) called car2. The car brand is Ford and the model is Focus. Print the brand of the car2 object.

292. Modify the car2 object you created (question 291). Change the brand name from Ford to Honda and the model name from Focus to Civic. Print the car2 model.

293. Write a code to delete car2 you created (question 292).

294. Jonathan and John cannot agree on whether you can include a function inside a class. Jonathan is saying you cannot put a function inside a class and John is arguing the opposite. Who is telling the truth?

    a. Jonathan
    b. John
    c. None

295. What is the purpose of the destructor __del__() function?

    a. To add an object
    b. It is called when an object gets destroyed
    c. To modify/collect the deleted object
    d. To lock the object

296. What is inheritance in Python classes?

    a. When a child class has a similar name to the parent class
    b. When one class inherits the methods and properties of another class
    c. When a child class is smaller than the parent class
    d. When a parent class has the properties of a child class

297. Which one of the following functions are used to by the child class to inherit from the parent class?

a. __init__()
b. super()
c. def()
d. self()

298. Mary writes the following code. She wants you to add something to the code.

```
class Father:
    def __init__(self,firstname,lastname,age):
        self.firstname = firstname
        self.lastname = lastname
        self.age = age

    def print_father(self):
        print("Your first name is {} and your last name is {}. "
              "You are {} years old".
              format(self.firstname, self.lastname, self.age))

# Creating the first object
father1 = Father("Richard", "Kelly", 42)

print(father1.print_father())
```

Create the second object - father2. The first name is Benson, the Last name is Carter and the age is 52. Print out father2.

299. Which one of the following codes is the correct way of creating a child class named Son that inherits from the Father class (question 298)?

a.

```
class Son(Father):
```

b.

```
class Father(Son):
```

c.

```
class Son():(Father):
```

d.

```
class son(father):
```

300. Add a child class to the code (question 298). The name of the child class is Son. The child class inherits properties from the Father class. Create the child class and do not add any content to it (pass). Once the class is created, create an object of the child class called son1(First name: Shawn, Last name: Kelly, Age: 21) and use the print_father method to print out son1.

301. Modify your code (question 300) by adding properties to the child class (Son class). Use the __init__() function, to assign 'height' to the Son class. Ensure that the Son class still inherits properties and methods from the Father class. Create a new method for the Son class called print_son (similar to print_father in the Father class). The print_son function should print what is printed by the print_father method but you should add "and your height is" at the end. Modify the son1 object you just created (question 299) (First name: Shawn, Last name: Kelly, Age: 21, height 5'9). Use the print_son method to print this out.

File handling

302. Which one of the following keywords are used to open text files in Python?
    a. append()
    b. read()
    c. open()
    d. extend()

303. You have a text file called names. You want to open and read this file. Write a code to open and read this file using the with statement.

304. You want to create a text file called animals. Write a code to create the file using the open() method and the with statement. Add an animal called Elephant to your file.

305. You want to add two names to the animals' file you have created (question 304). Write a code to open and add names to your file. Add Lion and Tiger to the file. Ensure that each animal is added to a separate line. Use the with statement. Write another code to open and read the updated file.

306. Which one of the following is true about the 'w' mode in the open() file method?
    a. It will overwrite an existing file
    b. It will not overwrite an existing file
    c. It will append to an existing file
    d. It will delete an existing file

307. Using the with statement, write a code to create a text file called my_first_file. Write 'This is my first file' on to the file. Then write another code to open and read your file.

308. Write a code to open the file you created in (question 307). Append a line at the end of the file, 'This is my last line'. Use the with statement. Once you append, write another code to open and read your file.

309. Write a code to delete the file you have created (question 308). Import os for this operation.

Extra Challenges

310. Write a code that finds and prints out odd numbers in the variable a.

```
a = [1, 2, 4, 6, 7, 9, 11, 17, 23, 26, 27]
```

311. Write a code that finds and prints out even numbers from the list.

```
a = [1, 2, 4, 6, 7, 9, 11, 17, 23, 26, 27]
```

312. Write a code that finds duplicates in the following list.

```
a = [1, 1, 3, 5, 6, 1, 3, 6, 7, 8, 9, 6, 8, 9,]
```

Print out the duplicates.

313. Write a function called convert that converts hours to minutes. The function should ask the user for input. The function should return the number of hours converted into minutes. Ensure that the code does not crash when a user inputs something that is not a number.

314. Create a simple calculator, that adds, subtracts, divides, and multiplies inputs from the user. Ensure that your code can handle Zero Division Errors and Value Errors.

315. Write a function called guess_num that asks a user to guess a number between 0 and 10. The code should generate a number and a user should guess that number. If the user guesses a number that is higher than the generated number, the code should tell the user that their guess is too high. If the guess is lower, the code should tell the user that the guess is too low. When the user guesses right, the code should tell the user that they have won. The code should run until the user guesses the right number.

316.Write a code that sums all the elements in the following list. Your code should add up to 78.

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,12]
```

317. Write a code that finds how many capital letters are in the string below. Print out the number of capital letters in the string.

```
str = 'This Is noT how wE writE A sentence'
```

318.Write a code that sums all the values in the following dictionaries.

```
a = {"Men": 20,"women": 25}

b = {"children": 23,"students": 44}
```

319.Write a function called smallest_odd that returns the smallest odd number from a list. The function should take one argument, a list. Use the list below. Your function should return 13.

```
a = [17, 23, 13, 24, 25, 33, 71, 81, 30]
```

320.Write a function called center that returns the middle character of a string. The function should take a string as an argument. If the string has no middle character, it should return “This string has no middle character”. When you pass it the string below, it should return ‘c’.

```
a = 'welcome'
```

321.Jane has written the following code:

```
names = ['Peter', 'Peter', 'Peter', "Peter"]
```

Jane wants to write a code that will be able to check if all the names in the list are the same. If the names are the same, the code should return, ‘All the names in the list are the same ’. If they are different, the code should return ‘The names in the list are not the same ’. Write a code for Jane.

Answers

Answers

1.a

It's a set of rules on how a Python code will be written and interpreted.

Think of syntax as the grammar of a programming language. For Python to understand what you want to do; you have to write a code in a 'grammar' that it will understand. If the syntax or grammar is not right, the program will generate an error because it will not understand what you want it to do.

2.a

A reserved memory location for an object.

When you create a variable, you are reserving some space where the object you have created can be saved. You assign it a point of reference, which is the variable name. To access or use the object you have to call the variable. In the code below to use 2, we need to call on the variable x.

```
x = 2
```

3.b

False

You can start a variable name with either an upper letter or a lower letter.

4.c

myname

A variable name is always on the left side of the equal sign.

5.d

-fruit- = "Orange"

You cannot start a variable name with a dash. Python allows you to start a name with an underscore (_) but not (-).

6.b

False

Indentation tells Python that something is part of a block of code. If you do not indent a code that is part of a block of code, your code will generate an error. Indentation is more than a readability thing.

7.a

True

The indentation of the print(x) is telling Python that the code is part of a block of code. However, these two lines of code are not a block of code. They are independent lines of code. For this reason, the print(x) should not be indented.

8.c

Number-seven

You cannot use a dash (-) in a variable name. However, you can use an underscore (_).

9.c

The variable name should not start with a number.

You cannot start a variable name with a digit. This is forbidden in Python. You can use digits inside the variable name, but not as the first character.

10.c

The name Peter should have been put in quotation marks.

The object should have been put in quotation marks to tell Python that it is a string. That's a proper syntax. The only thing that does not need quotation marks is digits and variable names.

11.c, e

My-name
My name

You cannot use a dash or leave space between characters when naming a variable.

12.a

False

False is a reserved word in Python. Reserved words are not to be used as variables. If you do not believe me, let us run the code below. We will use the reserved word as a name of a variable.

```
False = 67
print(False)
```

In the output, we get a syntax error because an object cannot be assigned to a reserved keyword False.

Output:

```
SyntaxError: cannot assign to False
```

13. List is not one of the reserved words in Python. However, it is a built-in function. As good practice, you are not to use reserved words and built-in function names as variables. If John uses the word as a variable, the code will run but the reference to the built-in function will not work.

In the example below, we have used the keyword list as a variable name. If we try to use the list keyword to create a list after using it as a variable name, it will not work, it will generate an error. Check the output below. This is why it is bad practice to use built-in function names as variable names.

```
list = 7
print(list)

# Creating a list using the keyword list, will not work
a = list((4,4))
print(a)
```

Output:

```
a = list((4,4))
TypeError: 'int' object is not callable
```

14.a

del

del is a reserved keyword that should not be used as a variable name. If you use it, your code will generate an error.

15.b

print is a built-in function in Python so it should not be used.

If you use print as a variable name all references to the built-in function will not work. Let's try to use print as a variable name and use the print function to see that it works.

```
print = 67
print(print)
```

The code generates an error because the print function cannot work anymore. It has been 'overwritten' by the print variable.

Output:

```
TypeError: 'int' object is not callable
```

16.a

Variable names are case-sensitive so fruit is not equal to Fruit.

To print fruit, Kelly needs to use the small-letter fruit in the code.

17.c and d

Fruit = Mango
Fruit = int(Mango)

C and d are not strings. If you check the class type of c and d it will not be a string.

18.c

print(c)

In Python, you can create multiple variables and assign multiple values. The number of variables should equal the number of values, otherwise, your code will generate an error. To access the values, you need to access the variable that matches the position of the value. In our example below the variable, a will print out ‘alphabet’ and b will print out ‘banana’ and so forth.

```
a, b, c = "alphabet", "banana", "car"
print(c)
```

Output:

```
car
```

19.b

#This is a comment

In Python, comments are preceded by the hashtag# sign or written between three quotes. See below:

```
'''
 This is a comment in Python.

'''
```

20.a, b

car = "Toyota"

car = 'Toyota'

John can use either double quotes or single quotes to create a string.

21.a

A statement is a line of code that does something while an expression is a line of code that evaluates to something.

The following is an expression because it evaluates to something.

```
4 + 2
```

If we print this out, we get 6.

```
print(4 + 2)
```

Output:

```
6
```

A statement on the other hand only does something. It does not evaluate anything and it does not print out any results. The following code is an example of a statement. It creates a function (does something) but it does not evaluate anything. When you print this out you will not get any results.

```
def my_function():
```

22.c

‘firstname’ + ‘lastname’

Remember an expression always evaluates into something. And, when you print out an expression it will give results.

```
'firstname' + 'lastname'
print('firstname'+ 'lastname')
```

Output:

```
firstnamelastname
```

23.b, c, d

float
integer
complex

The three types of numbers in python – Float, Integer, and complex.

Float – Float is any number that has decimal points. The number of decimal places does not matter. It can be a positive or negative number
Integer – Integer is any number that does not have any decimal points. It can be a negative or positive number
Complex – Complex numbers have both real numbers and imaginary numbers.

24.a

20

Any number without a decimal point is an integer in Python.

25.a.

float

```
y = 1.5
print(type(y))
```

Output:

```
<class 'float'>
```

26.a

str

```
name = "John"
print(type(name))
```

Output:

```
<class 'str'>
```

27.c

list

```
fruits = ["apples", "mangoes", "Oranges"]
print(type(fruits))
```

A list is surrounded by square brackets.

Output:

```
<class 'list'>
```

28.c

set

```
apples = {"apples", "mangoes", "Oranges"}
print(type(apples))
```

Output:

```
<class 'set'>
```

29.c

tuple

```
fruits = ("mango", "orange", "lemon")
print(type(fruits))
```

Output:

```
<class 'tuple'>
```

30.b

list

```
cars = ["BMW", "Mazda", "Honda", False, 1,2,9,10]
print(type(cars))
```

Output:

```
<class 'list'>
```

31.b

bool

```
y = False
print(type(y))
```

Output:

```
<class 'bool'>
```

32.a

dict

A dictionary has a key and a value. “name” is a key and “Peter” is a value. You can access the value through the key.

```
y = {"name": "Peter", "school": "yale"}
print(type(y)
```

Output:

```
<class 'dict'>
```

33.a

Converting one data type into another using a Python function.

We can use a Python function to convert one data type into another. In the example below, we are converting a float into an integer using the function int().

```
y = 1.5
y = int(y)
print(type(y))
```

When we print the type y, we can see from the output that the float has been converted into an integer class using the function int().

output:

```
<class 'int'>
```

34.a

5.0

In this code, Mary is trying to convert an integer into a float using typecasting. If you run the code below to check x type, you will get a floating class. Remember that a floating number has a decimal point.

```
x = 5
x = float(5)
print(x)
```

Output:

```
<class 'float'>
```

35.

The code below converts x into an integer using the int function is:

```
x = "10"
x = int(x)
print(type(x))
```

Output:

```
<class 'int'>
```

36.

Using the function str() we can convert an integer into a string

```
x = 10
x = str(x)
print(type(x))
```

Output:

```
<class 'str'>
```

37.

The code to convert an int to float using the float() function is:

```
x = 23
x = float(x)
print(type(x))
```

Output:

```
<class 'float'>
```

38.b

y = str(y)

The **str()** method will convert y into a string.

39.b

4

The int() method will convert a float into an integer. It will disregard decimal points.

40.

It is possible to create a tuple with one element. However, if you create a tuple with one element like the one below, and you check the data type of your variable it will not be a tuple but a string. See below:

```
z = ("a")
print(type(z))
```

Output:

```
<class 'str'>
```

To create a one-element tuple, we need to add a comma in front of the element. Check below:

```
z = ("a",)
print(type(z))
```

Output:

```
<class 'tuple'>
```

Another way to create a one-element tuple is to use the tuple keyword if you do not want to use the coma. Check below:

```
z = tuple("a")
print(type(z))
```

Output:

```
<class 'tuple'>
```

41.

Using the round brackets, we can create a tuple like this:

```
names = ("Jane", "Mary", "Alice")
print(type(names))
```

Output:

```
<class 'tuple'>
```

42.

We can use indexing to access the elements in a tuple since tuples are ordered. To access "Alice" we can use this code:

```
names = ("Jane", "Mary", "Alice")
print(names[2])
```

Output:

```
Alice
```

43.

We can use the keyword len() to check the length of our tuple
The code will return the number of elements in our tuple.

```
names = ("Jane", "Mary", "Alice")
print(len(names))
```

Output:

```
3
```

44.

Tuples are immutable. They cannot be changed once they are created. However, you can create a new tuple by copying the contents of a tuple and concatenating it with the content of another tuple.

The first thing is to create a new tuple with a single element 'Maria'.

```
new_name = ("Maria",)
```

We can now concatenate the tuple to our old tuple and create a new tuple.

```
names = names + new_name
print(names)
```

Output:

```
('Jane', 'Mary', 'Alice', 'Maria')
```

45.

Since tuples are not mutable, we cannot remove an item from a tuple we have created. We can only create a new tuple that will not have the name Maria in it. The easy way to do this is to use indexing. Let's create a variable called names_2.

```
names_2 = names[:3]
print(names_2)
```

Output:

```
('Jane', 'Mary', 'Alice')
```

46.

Using the count() function we can check the number of times the name “Jane” appears in our tuple – names. We can see from the output that Jane only appears once. We are passing the name ‘Jane’ as the argument.

```
print(names.count("Jane"))
```

Output:

```
1
```

47.c

index()

Index() method returns a position of a particular item in the tuple. It is possible to use the indexing method because tuples are ordered. In the example below, we use the index() method to search for the position of “Alice” in the tuple.

```
names = ("Jane", "Mary", "Alice")
print(names.index("Alice"))
```

We can see from the output the position of “Alice” in our tuple. Remember that Indices start at 0. In the tuple – names, “Jane” is position 0, which means ‘Alice’ is position 2.

Output:

```
2
```

48.d

There is no limit to the number of items you can add to a tuple.

Tuples can have duplicate values and they can take Boolean values. Tuples can also combine different data types at the same time. There is no limit to the number of items a tuple can take.

49.

Here is what happens when we try to create a tuple of colors:

```
colors = tuple( "Red", "Bue", "Green")
print(colors)
```

Output:

```
colors = tuple( "Red", "Bue", "Green")
TypeError: tuple expected at most 1 argument, got 3
```

Creating a tuple using the tuple() constructor is only possible if our tuple will contain only one element. To get around this, we need to add double brackets to our tuple. Let's make changes below:

```
colors = tuple (("Red", "Bue", "Green"))
print(colors)
```

You can see below that our tuple has no errors.

Output:

```
('Red', 'Bue', 'Green')
```

50.a, d

Tuples have ordered values, while sets don't

Tuples are immutable while sets are mutable

51.a

x is a tuple, y is a set

Remember a tuple has round brackets while a set has curly brackets.

52.False

You cannot use a sort() method to sort a tuple because it is immutable. The position of the items does not change. However, there is a way around it. We can use the sorted() function to sort a tuple. This function returns a list. We Will then have to convert the list back to a tuple. See the example below:

```
# A tuple of numbers
num = (1,2,9,8,5,78,45)

# using the sorted function to sort the tuple
num2 = sorted(num)
print(num2)
```

Output:

```
[1, 2, 5, 8, 9, 45, 78]
```

We can now convert the list back to a tuple using the tuple() function.

```
# Converting a list to a tuple
num2 = tuple(num2)
print(num2)
```

Output

```
(1, 2, 5, 8, 9, 45, 78)
```

53.b

Yes, you can have duplicates in the tuple.

The tuple below has two ‘Greens’.

```
colors = tuple(( "Red", "Bue", "Green","Green"))
print(colors)
```

Output:

```
('Red', 'Bue', 'Green', 'Green')
```

54.a

print(len(animals))

55.d

Tuples are immutable

If you want to update a tuple, you will have to create a new one. You cannot update a tuple once it is created.

56.c

Impossible to delete an item from a tuple.

We cannot delete an item from a tuple because they are not mutable. Tuples are permanent. The only way around it is to convert the tuple into a list, use the list method to remove the item, and then convert the list back to a tuple.

57.

We can combine the two tuples using the + operator and create a new tuple.

```
animals = ("dog", "cat", "duck", "chicken")
fruits = ("oranges", "apples", "mangoes", "bananas")
animals_and_fruits = animals + fruits
print(animals_and_fruits)
```

Output:

```
('dog', 'cat', 'duck', 'chicken', 'oranges', 'apples', 'mangoes', 'bananas')
```

58.

We can use multiplication to double the contents of a tuple. We can create a new tuple. Multiply our tuple by 2.

```
cars = ("BMW", "Isuzu", "Nissan")
cars2 = cars*2
print(cars2)
```

Output:

```
('BMW', 'Isuzu', 'Nissan', 'BMW', 'Isuzu', 'Nissan')
```

59.c

count()

The only tuple methods are count() and index().

60.c

list

List properties:

Lists are ordered.

Lists allow for duplicate values.

Lists are not immutable, that is they can be changed once they are created.

61.

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(type(numbers))
```

Output:

```
<class 'list'>
```

62.

Using slicing we can print the numbers

```
numbers = [10, 20, 40, 70, 60, 90, 100, 55, 78, 89]

# Using slice to print out the numbers
print(numbers[3:-5])
```

Output:

```
[70, 60]
```

63.

```
numbers = [10, 20, 40, 70, 60, 90, 100, 55, 78, 89]

# Using slice to print out the numbers
print(numbers[3:-3])
```

Output:

```
[70, 60, 90, 100]
```

64.

Using the sort() method we can change the position of the items in the list because lists are mutable.

```
numbers = [10, 20, 40, 70, 60, 90, 100, 55, 78,89]

# using sort() to sort the list in ascending order
numbers.sort()
print(numbers)
```

By default, the sort() method sorts the list in ascending order.

Output:

```
[10, 20, 40, 55, 60, 70, 78, 89, 90, 100]
```

65.b

False

Even though by default the sort() method sorts the list in ascending order, you can also sort a list in descending order. You can set the sorting criteria for the method. Below is a code sorting a list in descending order using the sort() method.

```
numbers = [10, 20, 40, 70, 60, 90, 100, 55, 78,89]

#we set reverse to true to sort the list in descending order
numbers.sort(reverse=True)
print(numbers)
```

Output:

```
[100, 90, 89, 78, 70, 60, 55, 40, 20, 10]
```

The sort() method can also be used to sort out words:

```
cars = ["BMW", "Isuzu", "Nissan"]
cars.sort(reverse=True)
print(cars)
```

Output:

```
['Nissan', 'Isuzu', 'BMW']
```

If you set reverse = False, you will sort the list in ascending order.

```
cars = ["BMW", "Nissan", "Isuzu"]
cars.sort(reverse=False)
print(cars)
```

Output:

```
['BMW', 'Isuzu', 'Nissan']
```

66.c

return()

Return is not a list method in Python. The rest are list methods.

67.b

2

This code checks how many times the item 'Toyota' appears in the list. It uses the list method count(). Toyota appears in the list 2 times; hence the code returns 2.

68.d

print(cars[3])

Remember that in Python counting starts at zero(Index 0). The 'BMW' in the list is at position zero.

```
cars = ["BMW", "Honda", "Nissan", "Toyota"]
print(cars[3])
```

Output:

```
Toyota
```

69.c

It is immutable

Lists are not immutable. You can delete, change, add and sort items in a list. Lists are ordered.

70.

Using the del method and indexing we can remove the items from the list.

```
a = [1,2,4,4,5,6,7,8,9]

# Using the del method to remove the items
del a[2:5]
print(a)
```

Output:

```
[1, 2, 6, 7, 8, 9]
```

71.a

print(len(a))

```
a = [1, 2, 4, 4, 5, 6, 7, 8]
print(len(a))
```

Output:

```
8
```

72.b

drop

Drop is not a list method.

73.b

girls_names.pop()

By default, **pop()** will remove the last item in the list. You can also pass the index of the item that you want to be removed.

```
girls_names = list(("Mary", "Rose", "Kate", "Amanda"))
print(girls_names.pop())
```

Output:

```
Amanda
```

74.b

append()

Append adds an item at the end of the list.
Let's use the append() method to add another item to our list. We will add the name "jane" to our list.

```
girls_names = list(("Mary", "Rose", "Kate", "Amanda"))

#use append to add Jane to the end of the list
girls_names.append("Jane")

print(girls_names)
```

Output:

```
['Mary', 'Rose', 'Kate', 'Amanda', 'Jane']
```

75.b

append adds a single item to the list, while extend increases the list by the number of iterable items.

In the following example, we extend list a with list b using extend() method.

```
a = [1, 2, 4, 4, 5, 6, 7, 8]
b = [9, 10, 11, 12, 13, 14, 15, 16]

a.extend(b)
print(a)
```

Output:

```
[1, 2, 4, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

On the other hand, using append() will only extend a list by one item - the other list.

```
a = [1, 2, 4, 4, 5, 6, 7, 8]
b = [9, 10, 11, 12, 13, 14, 15, 16]

a.append(b)
print(a)
```

You can see from the output below that the whole b list has been appended to list a. We now have a nested list.

Output:

```
[1, 2, 4, 4, 5, 6, 7, 8, [9, 10, 11, 12, 13, 14, 15, 16]]
```

76.b

BMW and Toyota

```
cars = ["Volvo", "Honda", "BMW", "Toyota"]
cars[2:4] = ["Audi","Ford"]
print(cars)
```

Output:

```
['Volvo', 'Honda', 'Audi', 'Ford']
```

77.d

cars[-4]= "Benz"

With negative indexing, you are counting from right to left.
Here is the code below:

```
cars = ["Volvo", "Honda", "BMW", "Toyota"]

cars[-4]= "Benz"
print(cars)
```

Output:

```
['Benz', 'Honda', 'BMW', 'Toyota']
```

78.a

0

The index finds the position of 'Volvo' in the list. Remember indexing in Python starts at 0. Volvo is sitting at position zero.

79.b

sorted()

Sorted() is not a list method. It is a built-in function. It can work with lists and other things such as tuples.

80.

Sorted() is a built-in function. It's not a list method, so it can stand on its own.

```
x = [1, 3, 5, 7, 8, 9, 2, 4, 5, 7, 10]
y = sorted(x)
print(y)
```

Output:

```
[1, 2, 3, 4, 5, 5, 7, 7, 8, 9, 10]
```

81.c, d

You can change sets after they are created.
Sets are surrounded by curly brackets.

Sets are not ordered, they do not allow duplicates, but they can be modified.

82.

Here is a set below. Remember to use curly brackets for sets. Print out the data type using the type () function.

```
cars = {"Benz", "Ford","Toyota"}
print(type(cars))
```

Output:

```
<class 'set'>
```

83.

You cannot replace Toyota with Honda once the set is created. You can only add new items to the set, and remove items, but you cannot replace items in the set. The only way around it is to create a new set.

84.

We can use the update() method to merge the two sets.

```
set1 = {10, 67, 90, 91, 77, 73}
set2 = {2, 1, 4, 6, 8, 7}

#merge the set2 to set1 using the update method
set1.update(set2)
print(set1)
```

The output is a set that contains combined elements.

Output:

```
{1, 2, 67, 4, 6, 7, 73, 10, 77, 90, 91}
```

85.

Using remove() method we can remove the 77 from set1

```
set1 ={1, 2, 67, 4, 6, 7, 73, 10, 77, 90, 91}
set1.remove(77)
print(set1)
```

Output:

```
{1, 2, 67, 4, 6, 7, 73, 10, 90, 91}
```

86.b

method()

87.

We can use the clear() method to remove all elements from the set.

```
v = {12, 90, 89, 23, 56}
v.clear()
print(v)
```

The output is an empty set.

Output:

```
set()
```

88.

Using the set() constructor we can create a set. Take note of the double brackets. The set() constructor takes one argument that's why we put everything in brackets before calling the constructor.

```
num = set((1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
print(type(num))
```

Output:

```
<class 'set'>
```

89.

We use the len() method to check the number of elements in a set

```
num = set((1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
print(len(num))
```

Output:

```
10
```

90.a

True

You cannot replace items once a set is created, but you can add new items using the add() method and the update() method. The add() method only adds one item at a time to the set. You can also remove items using the remove() method.

Here is an example below. We added 11 to the num set using the add() function.

```
num = set((1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
num.add(11)
print(num)
```

Output:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

91.

Let's use examples to demonstrate the difference between the two methods.

Update() method

Update method updates the original set with the content of the new set. See the example below.

```
set1 = {10, 67, 90, 91, 77, 73}
set2 = {2, 1, 4, 6, 8, 7}

set1.update(set2)
print(set1)
```

Output:

```
{1, 2, 67, 4, 6, 7, 8, 73, 10, 77, 90, 91}
```

Union () method

The union() method combines two sets to create a new set that contains elements from both sets. See the example below:

```
set1 = {10, 67, 90, 91, 77, 73}
set2 = {2, 1, 4, 6, 8, 7}
set3 = set1.union(set2)
print(set3)
```

Output:

```
{1, 2, 67, 4, 6, 7, 8, 73, 10, 77, 90, 91}
```

92.c

Sets have a similar bracket type to dictionaries.

Both sets and dictionaries use curly{} brackets.

93.

We can use the update() method to add cars2 to cars 1.

```
cars = {"Volvo", "Honda", "BMW", "Toyota"}
cars2 = {"Audi","Ford"}
cars.update(cars2)
print(cars)
```

We can see from the output that elements of car2 have been added to cars.

Output:

```
{'Ford', 'BMW', 'Honda', 'Toyota', 'Volvo', 'Audi'}
```

94.b

The plus sign + is used to add values together.

You can use the plus sign to add values together and concatenate strings.

95.b

8

This is an exponentiation sign (double *).
This expression is equivalent to 2 * 2 * 2

```
x = 2**3
print(x)
```

Output:

```
8
```

96.c

36

The breakdown of this code is as follows:
6 * 3 = 18
18* 2 = 36

```
y = 6 * 3
print(y*2)
```

Output:

```
36
```

97.b

/

/ This is a division operator.

The division operator will always return a floating number.
Here is an example below:

```
y = 6 / 3
print(y)
```

Output

```
2.0
```

98.d

2

// this is a floor division sign. It rounds the answer to the nearest round number.

6 / 4 = 1.5, however, since we are using the // sign the 1.5 will be rounded off to 1. When we multiply the 1 by 2 (using the * sign) we get a 2.

```
x = 6//4
print(x*2)
```

Output:

```
2
```

99.c

3.0

This is a division sign /. When you divide 6 / 4 you get 1.5. Multiply 1.5 by 2, you get 3.0. Remember that a / sign will always return a floating number that is why we are getting a 3.0.

```
x = 6/4
print(x*2)
```

Output:

```
3.0
```

100.b

//operator returns the result without decimal places, while / operator returns the results with decimal places.

// operator rounds off the division to the nearest whole number.

```
x = 6//4
print(x)
```

Output:

```
1
```

/ Operator returns the result of division with decimal points.

```
x = 6/4
print(x)
```

Output:

```
1.5
```

101.b

c += 5

This is the shorthand method of c = c+ 5.

102.b

0

The modulus % operator returns the remainder of the division. In this case, the result is zero.

```
b = 12%3
print(b)
```

Output:

```
0
```

103.d

equal

This is not a logical operator. The rest are logical operators in Python.

104.a

True

The expression is simply evaluating **if 10 is NOT equal to 5**, and, indeed, **10 is not equal to 5**.

105.c

Y //=5

106.b

A sequence of characters surrounded by quotation marks

A string in Python is surrounded by quotation marks. You can use double or single quotation marks. The codes below are the same.

```
x = 'car'
x = "car"
```

107.b

False

Split() splits the string at a specified separator and returns a list of strings from the string. Check the example below.

```
fruits = "mangoes oranges"
print(fruits.split())
```

Output:

```
['mangoes', 'oranges']
```

Strip() trims a string by removing any space or specified characters at the beginning and the end of a string.

```
fruits = "$mangoes oranges$"
print(fruits.strip("$"))
```

Output:

```
mangoes oranges
```

108.d

y = 5

A string is created by using double quotation marks, single quotation marks, and the keyword **str().** From the codes below, you can see that the output for y = 5 is **int().**

```
y = str(5)
print(type(y))
y = '5'
print(type(y))
y = "5"
print(type(y))
y = 5
print(type(y))
```

Output:

```
<class 'str'>
<class 'str'>
<class 'str'>
<class 'int'>
```

109.c

print(y[5: 13])

Here is the code below:

```
y = "I am learning python"
print(y[5: 13])
```

Output:

```
learning
```

110.a

print(y[-6:])

Here is the code below. Remember to start counting from right to left.

```
y = "I am learning Python"
print(y[-6:])
```

Output:

```
Python
```

111.

We are going to use the replace() method to replace the word in our string.

```
y = "I am learning python"
y = y.replace("learning", "studying")
print(y)
```

Output:

```
I am studying python
```

112.

We can use the strip() method to remove the characters**.**

```
fruits = "#mangoes, oranges#"
print(fruits.strip('#'))
```

Output:

```
mangoes, oranges
```

113.b

He cannot use double quotes inside a string surrounded by double-quotes.

Dwayne should have used single quotes inside the string. Check the example below:

```
y = "I don’t like 'lazy' people"
print(y)
```

Output:

```
I don’t like 'lazy' people
```

Another way around it is to use escape characters inside the string. That way you can still use double quotes inside the string See the example below:

```
y = "I don’t like \"lazy\" people"
print(y)
```

Output:

```
I don’t like "lazy" people
```

114.a

capitalize()

Here is an example to capitalize a y variable

```
y = "ben"
print(y.capitalize())
```

Output:

```
Ben
```

115.a

lower()

Example below:

```
y = "Ben"
print(y.lower())
```

Output:

```
ben
```

116.a

upper()

The code below demonstrates how the upper() method converts the whole string to the upper case.

```
y = "i love ben"
print(y.upper())
```

Output:

```
I LOVE BEN
```

117.b

len()

Remember Python counts the characters. The space between words is counted as a character.

```
y = "i love ben"
print(len(y))
```

Output:

```
10
```

118.c

\n

Here is the example below. In this code, we are telling Python to write “ben” in another line. You can see in the output that Ben is in another line.

```
y = "I love \nBen"
print(y)
```

Output:

```
I love
Ben
```

119.b

True

The code will return a True because a is a bool() value.

120.a

True

7 is greater than (7-1).

121.b

True

Insinstance() is checking if the variable home is a string or not. In this instance, it is a string that is why it returns True. If we replace 'str' with 'int', it will return False because the variable is not an integer. See below.

```
home = "The place where I relax"
print(isinstance(home, int))
```

Output:

False

122.a

student = {"name": "John", "age": 23}

A dictionary has curly brackets and it has a key and a value. In the dictionary above "name" is a key and "John" is a value.

123.a

Immutable

Dictionaries are not immutable. You can change the values of a dictionary.

124.a

print(cars.values())

Here is the code below. The code prints out the values of the dictionary.

```
cars = {"name": "Ford", "year": 1992}
print(cars.values())
```

Output:

```
dict_values(['Ford', 1992])
```

125.

Here is the code below. The code prints out the keys of the dictionary.

```
cars = {"name": "Ford", "year": 1992}

for keys, values in cars.items():
    print(keys, values)
```

Output:

```
name Ford
year 1992
```

126.a

cars["name"] = "BMW"

You can see from the output of this code that the car name has changed from Ford to BMW. The new dictionary has BMW as a value.

```
cars = {"name": "Ford", "year": 1992}
cars["name"] = "BMW"
print(cars)
```

Output:

```
{'name': 'BMW', 'year': 1992}
```

127.d

Keys cannot be changed

Dictionary keys cannot be changed once they are made. Values of the keys can be changed.

128.

Here is the code below:

```
cars = {"name": "Ford", "year": 1992}
cars ["color"] = "White"
print(cars)
```

Output:

```
{'name': 'Ford', 'year': 1992, 'Color': 'White'}
```

129.

Here is the code below:

```
cars = {"name": "Ford", "year": 1992}
cars ["year"] = 2004
print(cars)
```

Output:

```
{'name': 'Ford', 'year': 2004}
```

130.b

update()

131.

You can remove a key from a dictionary using the pop() method. Pass the key that you want to remove as an argument.

```
cars = {"names": "Nissan", "year": 2007}
cars.pop('year')
print(cars)
```

You can see from the output that the key and its value have been removed from the dictionary.

Output:

```
{'names': 'Nissan'}
```

132.

Alison can use the keyword del() to delete the dictionary. You can see below that trying to print cars after running del generates an error because the dictionary cars do not exist.

```
cars = {"names": "Nissan", "year": 2007}
del cars
print(cars)
```

Output:

```
NameError: name 'cars' is not defined
```

133.

We are going to create a dictionary called students, that will be nested. When we run print students, you can see that we get a dictionary with dictionaries in it. This is a nested dictionary.

```
Student1 = {"name": "John", "grade" : "F"}
Student2 ={ "name" : "Alison", "grade" : "B"}
Student3 = {"name" : "Ben", "grade" : "C"}

students ={"Student1": Student1, "student2":
                Student2,"student3":Student3}
print(students)
```

Output:

```
{'Student1': {'name': 'John', 'grade': 'F'}, 'student2': {'name': 'Alison', 'grade': 'B'},     'student3': {'name': 'Ben', 'grade': 'C'}}
```

134.

Using the keyword pop() Mary can specify which dictionary she wants to remove from the nested dictionary. In this case, she wants to remove student3. Here is the code below:

```
students = {'Student1': {'name': 'John', 'grade': 'F'},
            'student2': {'name': 'Alison', 'grade': 'B'},
            'student3': {'name': 'Ben', 'grade': 'C'}}

students.pop('student3')
print(students)
```

If we print this dictionary now, you will see that student3 has been deleted.

Output:

```
{'Student1': {'name': 'John', 'grade': 'F'}, 'student2': {'name': 'Alison', 'grade': 'B'}}
```

135.d

clear()

Clear() will remove contents from the dictionary. Let's use our previous question example to demonstrate this. When we try to print the dictionary after running clear(), we get an empty dictionary.

```
students = {'Student1': {'name': 'John', 'grade': 'F'},
        'student2': {'name': 'Alison', 'grade': 'B'},
        'student3': {'name': 'Ben', 'grade': 'C'}}


students.clear()
print(students)
```

Output:

```
{}
```

136.b

newdict = mydict.copy()

If we run the code below, we can see that a copy of the mydict dictionary has been created.

```
mydict = {"name": "Mary", "occupation": "Singer"}
newdict = mydict.copy()
print(newdict)
```

Output:

```
{'name': 'Mary', 'occupation': 'Singer'}
```

137.

The code below uses the get() method to access the value. You will have to pass the key of the value you are trying to access to the get() method.

```
mydict = {"name": "Mary", "occupation": "Singer"}
print(mydict.get("occupation"))
```

Output:

```
Singer
```

138.b

delete()

delete() is not dictionary method.

139.b

Dictionaries allow duplicate values.

Dictionaries allow duplicate values, but not duplicate keys. If you have two keys with the same name in a dictionary, then one of the keys will be ignored. Let's see the example below. The dictionary below has two keys called 'name'. When we print the dictionary, only the second key called name and its value are printed the first one is ignored.

```
car = {'name': 'Honda', 'name':
      'Ford', 'city': 'Tokyo'}
print(car)
```

Output:

```
{'name': 'Ford', 'city': 'Tokyo'}
```

140.b

Items in a dict are accessed by keys while items in a list are accessed by referencing their position.

141.

```
x = { "name": "carol", "age": 23, "school": "Yale"}
print(x['school'])
```

Output:

```
Yale
```

142.b

They are both mutable.

Dictionaries and lists can be changed after they are created.

143.a

Dictionary is mutable while a tuple is immutable.

144.

Here is the code below to loop through the dictionary.

```
x = { "name": "carol", "age": 23, "school": "Yale"}
for items in x.values():
    print(items)
```

Output:

```
carol
23
Yale
```

145.

```
x = { "name": "carol", "age": 23, "school": "Yale"}

for items in x.keys():
    print(items)
```

Output:

```
name
age
school
```

146.

The expression ‘name in x’ will return a Boolean value. True if the key is in the dictionary and False if the key is not in the dictionary.

```
x = { "name": "carol", "age": 23, "school": "Yale"}
if 'name' in x:
    print('name is one of the keys')
else:
    print('name is not one of the keys')
```

Output:

```
name is one of the keys
```

147. We can combine the two dictionaries using the **Bitwise OR operator |.**

```
a = {'name': 'John', 'age': 25}
b = {'school': 'Yale', 'location': 'USA'}

c = a | b
print(c)
```

Output:

```
{'name': 'John', 'Grade': 11, 'school': 'yale', 'location': 'USA'}
```

148.b

A block of code that performs a specific task and runs when called.

A function will only run when it is called.

149.b

def

Every function block in Python will start with the keyword def.

150.b

def my_car():

151.a

Values that are passed into a function when it is called.

Arguments are passed into the brackets of the function. The code below has two parameters between the round brackets, a and b. When you call the function, you will have to pass two arguments for a and b.

```
def add(a,b):
```

152.b

def car(carname):

153.

The parameters of the function will be put between the brackets.

```
def add(number1,number2):
    sum = number1 + number2
    return sum
print(add(5,7))
```

Output

```
12
```

154.c

def my_function (car = "BMW"):

The significance of a default argument is that whenever you call the function and you do not pass any arguments it will use the default argument.
In the code below, when we call the function, in the first code we pass “Honda” as an argument, and the code prints out Honda. In the second one, we do not pass any argument, and the function prints out BMW, which is our default argument.

```
def my_function (car="BMW"):
    print("My favorite car is:" + car)


# Calling the function
my_function("Honda")
my_function()
```

Output:

```
My favorite car is: Honda
My favorite car is: BMW
```

155.a

To tell the function to return the results of the function.

In the example below the return statement is telling the function to print the results of adding number1 and number2.

```
def add(number1,number2):
    return number1 + number2
print(add(12,65))
```

Output:

```
77
```

156.d

All the above.

157.

Here is the code below:

```
def MultiplyAndDivide(a,b,c):
    results = a*b/c
    return results
print(MultiplyAndDivide(12,24,8))
```

Output:

```
36.0
```

158.c

A function can accept zero parameters.

A function does not require parameters to work. It is possible to create a function with zero parameters. The function below has zero parameters.

```
def myFunc():
    print("My name is John ")
myFunc()
```

Output:

```
My name is John
```

159.b

She's right we can use different arguments for one function.

A function is powerful because you can use different arguments for its parameters every time it is called. For example, for the function below, we can use different arguments for a, b, and c every time we call the function and it will still work. Functions help in writing clean code by ensuring we avoid repeating code. Instead of repeating code we just call the same function and we can use different arguments.

```
def MultiplyAndDivide(a,b,c):
    results = a*b/c
    return results
print(MultiplyAndDivide(12,24,8))
```

Example 2

```
def MultiplyAndDivide(a,b,c):
    results = a*b/c
    return results
print(MultiplyAndDivide(16,14,6))
```

160.b

Pass *args as the argument of the function.

By using *args as a parameter name, you can pass any number of arguments into the function.
In the example below, we have passed 3 arguments into the function. You can change the number of arguments and it will still work. When you use *args as an argument, Python will store the input into a tuple.

```
def myFunc(*args):
  return "His cars are " + str(args).strip('()')

print(myFunc("BMW","Honda","Benz"))
```

Output:

```
His cars are 'BMW ', 'Honda ', 'Benz '
```

161.d

There is no limit to the number of parameters a function can take.

You can pass as many parameters as you want into a function.

162.b

The sub1 and return should have been indented.

In Python, indentation is a way of telling the interpreter that a code is part of a block of code. All codes that are part of the function must be indented. If not, the code will generate an indentation error. Python uses four spaces for indentation.

Below is how the code should be written with proper indentation.

```
def subtract(num1,num2):
    sub1 = num1 - num2
    return sub1
print(subtract(24,26))
```

This code is no longer generating an error.

Output:

```
-2
```

163.b

Declare the variable, a global variable.

Here is an example of that below. In this example, we have a variable f inside a function.

```
def my_fun():
    f = 12
    sum = f + b
    return sum
print(my_fun())
```

Let's see what happens when we try to use the f variable outside the function.

```
sub = 23 - f
print(sub)
```

The code generates an error because the variable f is inside a function. See below.

Output:

```
NameError: name 'f' is not defined
```

To use the f variable outside the function we need to use the global keyword inside the function. Let's add global to the function.

```
def my_fun():
    global f
    f = 12
    sum = f + b
    return sum
print(my_fun())
```

Now let's use it outside the function.

```
sub = 23 - f
print(sub)
```

Now it works. Check the output below.

Output:

```
11
```

164.b

: is missing at the end of the function name.

This is how the code should have been:

```
def my_fun():
    sum = 11 + 11
    return sum
print(my_fun())
```

165.b

A variable declared inside a function.

In the code below the x variable is a local variable because it is declared inside a function and can only be accessed inside the function.

```
def my_fun():
    x = 1
```

166.c

print(subtract(10,5))

Here is the code below. Subtract is the name of the function, so we need to call it every time we want to run the function.

```
def subtract(num1, num2):
    sub1 = num1 - num2
    return sub1
print(subtract(10, 15))
```

Output:

```
-5
```

167.a

Add the keyword pass to the function.

Use the keyword pass when you want to create a function but are not yet sure what to add to it. The pass keyword is simply telling the code to do nothing. When Python sees a pass statement it will not run the code.

```
def my_function():
  pass
```

168.

Here is the code below:

```
def convert(km):
  miles = km * 0.621371
  return str(round(miles)) + ' miles'
print(convert(109))
```

Output:

```
68 miles
```

169.

Here is the code below:

```
def rectangle_area(a,b):
  area = (a * b) * 2
  return area

print(rectangle_area(15,67))
```

Output:

```
2010
```

170.

Here is the code below:

```
def area_of_triangle (base, height):
    area = (base * height)/2
    return area
print(area_of_triangle(24,67))
```

Output:

```
804.0
```

171.

```
def max_num(a):
    if a < 10:
        return "Max is less than 10"
    else:
        return "The number is not less than 10"

print(max_num(12))
```

Output:

```
The number is not less than 10
```

172.b

The print statement is not indented.

Indentation is important in Python. The print line is part of the if statement block of code, that is why it should be indented.

```
a = 11
b = 10

if a != b:
    print("it's not equal")
```

Output:

```
it's not equal
```

173.b

elif statement

The first statement is the if statement**.** The second condition should use the elif statement. Elif is shorthand for 'else if'. The last conditional statement is else. Else statement is optional.

174.

```
a = 10
b = 20

if a > b:
   print("a is greater than b")
elif a < b:
   print("b is greater than a")
else:
   print("The two numbers are equal")
```

Output:

```
b is greater than a
```

175.

```
a = 22
b = 34

if b > a:
   print("B is superior to A")
elif b < a:
   print("A is superior to B")
```

Output:

```
B is superior to A
```

176.a

and

If a statement has two conditions, you can use the keyword and
Here is the example below.

```
if a > b and b < c:
```

177.

In this example, we are using the 'and' keyword in our conditional statement.

```
a = 12
b = 15
c = 18

if a > b and c > b:
   print("Life is not fair")
else:
   print("Life is beautiful")
```

Output:

```
Life is beautiful
```

178.

In this conditional statement, we are using the or operator.

```
a = 22
b = 34
c = 56

if a > b or b > c:
   print("Life is good")
else:
   print("Life is great")
```

Output:

```
Life is great
```

179.

```
if a > b or a > c: print("Life is good")
```

180.b

or

or is a reserved keyword used in conditional statements.

181.c

A loop that iterates over a given sequence of items

A for loop will iterate over a string, a list, and so forth. Anything that has a sequence of items can be iterated over by a for loop.

182.

Here is the code below:

```
a = "string"

for letters in a:
    print(letters)
```

Output:

```
s
t
r
i
n
g
```

183.

Here is the code below:

```
animals = ("dog", "cat", "duck", "chicken")

for animals in animals:
   print(animals)
```

Output:

```
dog
cat
duck
chicken
```

184.c

break

The break statement breaks the loop. It stops the loop.

185.

Remember that a for loop iterates from left to right. In the following code, we are telling the loop to stop the iteration once it prints 'Honda'.

```
cars = ["Volvo", "Honda", "BMW", "Toyota","Toyota"]

for car in cars:
   print(car)
   if car == "Honda":
      break
```

Output:

```
Volvo
Honda
```

186.b

continue

187.b

To stop the current iteration and begin a new one.

The continue statement tells the loop that once it reaches a particular item, it should stop, skip that item and proceed to the next one.

188.

In the following code, we are using the continue statement to tell the loop that if it reaches “BMW” it should skip it and continue to the next item.

```
cars = ["Volvo", "Honda", "BMW", "Toyota","Toyota"]

for car in cars:
   if car == "BMW":
      continue
   print(car)
```

You can see from the output below that we do not have a BMW in our output.

Output:

```
Volvo
Honda
Toyota
Toyota
```

189.d

0,1,2,3,4

We are telling the loop to break or stop once it finds 5 in the iteration. So, 5 will not be printed.

190.d

0,1,2,3,4,6,7,8,9

In this loop, the continue statement is telling the loop that once it finds 5 in the iteration, it should skip it and continue with the next item. So, 5 will not be printed**.**

191.a

Mary, John, Peter

We are printing the values from the dictionary, which is Mary, John, Peter.

```
students ={"Student1": "Mary", "student2":
          "John","student3":"Peter"}
for x in students.values():
   print(x)
```

Output:

```
Mary
John
Peter
```

192.b

hllo

The loop will skip the letter ‘e’ and print out the rest.

193.c

Red, Green

The loop will break once it finds blue in its path. It will not print blue.

194.b

red, green

Since the print statement has come before the if statement, this code will print red and green. If the print statement came after the break statement, the code would print ‘red’ only.

Example 1

```
colors = ["red", "green", "blue"]
for x in colors:
   print(x)
   if x == "green":
     break
```

Output:

```
red
green
```

Example 2

```
colors = ["red", "green", "blue"]
for x in colors:
   if x == "green":
     break
   print(x)
```

Output:

```
red
```

195.a

red, green, and blue

Since the print statement has come before the if statement, this code will print out all the items in the iteration. For the condition to work, the print statement should come after the continue statement.

```
colors = ["red", "green", "blue"]
for x in colors:
   print(x)
   if x == "green":
     continue
```

Output:

```
red
green
blue
```

In the code below, we have moved the print statement after the continue statement and the out has changed.

```
colors = ["red", "green", "blue"]
for x in colors:
   if x == "green":
      continue
   print(x)
```

Output:

```
red
blue
```

196.d

2,10

Here is a simple way to understand the code:
2 is where the range starts from.
16 is the end of the range.
8 is how many numbers the code should skip after printing 2, so after 2 it will print 10. After 10 it is supposed to skip another 8 numbers, but since our range ends at 16, the code will stop at 10.

```
for x in range (2, 16, 8):
  print(x)
```

Output:

```
2
10
```

197.a

2,6,10,14

```
for x in range(2, 16, 4):
  print(x)
```

Output:

```
2
6
10
14
```

198.b

A loop within another loop.

When you have a loop inside a loop, it is a nested loop. Below is a simple illustration of a nested loop. There is a loop inside a loop

```
for a in b:
    for b in c:
        print(c)
```

199.

Here is the code below:

```
a = [ "yellow", "green", "red"]
b = [1,2,3,4]

for colors in a:
    for numbers in b:
        print(colors,numbers)
```

Output:

```
yellow 1
yellow 2
yellow 3
yellow 4
green 1
green 2
green 3
green 4
red 1
red 2
red 3
red 4
```

200.a

You cannot use a for loop in a function.

You can easily combine a function with a for loop. Below is a simple code that combines a function with a for loop.

```
b = [ "yellow", "green", "red"]

def colors(b):
    for color in b:
        print(color)
colors(b=b)
```

Output:

```
yellow
green
red
```

201.a

A for loop iterates through a given number of items or range of objects, while a while loop iterates through items or objects as long as the condition is true.

A while loop iterates when a certain condition is true, a for loop iterates through a sequence of items.

202.b

Yes

This is because, in the given code, the condition will always be true. The variable a will always be less than 10. As long as that condition is true, the code will run forever. Try it, but be careful it will crash your computer.

203.

Here is the code below:

```
a = 1
while a < 10:
    print("This number is less than 10")
    a += 2
```

Output:

```
This number is less than 10
This number is less than 10
This number is less than 10
This number is less than 10
This number is less than 10
```

204.a, b, d

The while loop will stop when a condition is false.
You can stop the while loop by using the break statement.
A while loop will stop when the computer runs out of memory

Any of the above scenarios will stop a while loop.

205.c

0,1,2,4,5

The continue statement is simply telling the while loop that once it finds 3 in the iteration, it should stop, skip 3 and continue.

```
a = -1
while a < 5:
  a += 1
  if a == 3:
    continue
  print(a)
```

Output:

```
0
1
2
4
5
```

206.d

0,1,2

The while loop will break once the iteration finds a 3. It will not print the 3.

```
a = -1
while a < 5:
  a += 1
  if a == 3:
    break
  print(a)
```

Output:

```
0
1
2
```

207.

Here is the loop below:

```
a = 1
while a < 5:
    a += 1
    print("This number is less than 5")
else:
    print("End of the loop")
```

Output:

```
This number is less than 5
This number is less than 5
This number is less than 5
This number is less than 5
End of the loop
```

208.a

John is right, you can use a while loop to iterate through the dictionary.

A while loop iterates while a certain condition is true. As long as we can set that condition, we can use it to iterate through a dictionary.

Below we have a while loop, looping through a dictionary.

```
cars = {"name": "Ford", "year": 1992}

a = 2
while a < 3:
    x = cars.items()
    print(x)
    a += 1
```

Output:

```
dict_items([('name', 'Ford'), ('year', 1992)])
```

209.

Yes, you can use a for loop inside a while loop. Here is a simple example below.

```
cars = {"name": "Ford", "year": 1992}

a =1
while a < 10:
    for items in cars.values():
        print(items)
        a += 5
```

Output:

```
Ford
1992
```

210.b

If statements can be used with while loops and for loops.

Both while and for loops can incorporate conditional statements.

211.

```
a = 23
b = 45
def add(a, b):
    sum1 = a + b
    return sum1
print(add(23,45))
```

Output:

```
68
```

212.

```
a = 23
b = 45

def multiply(a, b):
    c = a * b
    return c
print(multiply(23,45))
```

Output:

```
1035
```

213.

```
a = 45
b = 66
c = 12

def multiply_and_divide(a,b,c):
    d = (a * b)/c
    return d
print(multiply_and_divide(23,45, 12))
```

Output:

```
86.25
```

214.

```
x = (10, 20, 21, 23, 24, 56, 78, 89)
y = min(x)
print(y)
```

Output:

```
10
```

215.

```
x = (10, 20, 21, 23, 24, 56, 78, 89)

for num in x:
    if num == max(x):
        print(num)
```

Output:

```
89
```

216.c

math.divide()

217.

You will have to import the math module.

```
import math
x = 64


def square_root(x):
    y = math.sqrt(x)
    return y
print(square_root (64))
```

Output:

```
8.0
```

218.b

math.isnan()

Math.isnan() checks if a given item is not a number. If it is a number, it will return False. Here is an example below:

```
x = 3
print(math.isnan(x))
```

It returns False because x is a number.

Output:

```
False
```

219.

Using the abs() function, you can convert any number into an absolute number.

```
a = -5.91
print(abs(a))
```

Output:

```
5.91
```

220.b

23

This method **rounds down** the number to the nearest integer.

Here is the code below:

```
z = math.floor(23.3)
print(z)
```

Output:

```
23
```

221.a

24

This method **rounds up** the number to the nearest integer.

Here is a code below:

```
a = math.ceil(23.4)
print(a)
```

Output:

```
24
```

222.a

368

```
p = max (12,13,16)
s = 23
print(p*s)
```

Output:

```
368
```

223.b

0

The modulus sign % gives the remainder of the division. When you divide 16 by 8, there is no remainder.

224.

Using exponentiation, Mary can write the code in this way.

```
a = 2**5
```

We can even check if these expressions are equal, with the code below.

```
a = 2 * 2* 2 * 2* 2
b = 2**5
print(a==b)
```

Output:

```
True
```

225.

8 is the remainder when you divide 89 by 9.

```
a = 89
b = 9

c = a % b
print(c)
```

Output:

```
8
```

226.

Using pow(), Mary can write the code below.

```
a = pow(5,4)
```

If we compare the two expressions, we get a True statement, meaning they are equal.

```
b = 5 * 5 * 5 * 5
a = pow(5,4)
print(b==a)
```

Output:

```
True
```

227.c

18

The trunc() method will remove the decimal point and return a whole number**.**

228.d

5

In this code, we are simply getting 5 to the power of 3, and 8 is a modulus. (5 * 5 * 5) % 8.

```
x = pow(5, 3, 8)
print(x)
```

Output:

```
5
```

229.a

6.238

We are multiplying pi by 2 in this expression.

230.d

1080

This code is simply multiplying the min of variable x, with the max of variable y using the math method math.prod().

231.b

input()

232.

```
my_name = input("Please enter your name:")
```

When you run this code, it will prompt the user to enter their name.

233.

Here is the code below.

```
def add():
    a = int(input("Enter a number:"))
    b = int(input("Enter another number"))
    c = a + b
    return c
print(add())
```

Output:

```
75
```

234.

Here is the code below.

```
def multiplyAndDivide():
    a = int(input("Enter a number:"))
    b = int(input("Enter another number"))
    c = int(input("Enter another number"))
    d = (a*b)/c
    return d
print(multiplyAndDivide())
```

Output:

```
180.0
```

235.

```
def your_age(age):
    age = input("Enter your age:")
    print('Your age is ' + age + ' years old')
your_age(25)
```

Output:

```
Your age is 21 years old
```

236.b

A variable that can be accessed within the local and global scope.

A variable that can be accessed inside and outside a function is global.

237.b

Use the global keyword within the function to declare c a global variable.

By using the global keyword, you are making variable c accessible within the local and global scope.
Janet should write the following code:

```
def my_Function():
    global c
    c = 200
```

238.a

A variable that is declared inside a function and can only be used in that function.

In the code below, c is a local variable because it can only be used in this function.

```
def my_Function():
    c = 200
```

239.a

True

By default, any variable declared outside a function is global. It can be used in a function and outside of a function. In the code below, the variable num can be used within and outside the function. This makes the variable num a global variable.

```
num = 12
def my_Function():
    c = 200 + num
    return c
print(my_Function())
```

Output:

```
212
```

240.b

False

When you declare a variable within a function, you do not need to use the word 'local' to make it a local variable. By default, a variable declared within a function is a local variable.

241.b

False

To make a variable declared inside a function global, you have to declare it global by using the global keyword in the function.

242.b

try

The try keyword is used to handle errors in the code.

243.a

True

An error that is not handled will stop the program from running.

244.b

It handles the error raised in the try block.

In the code below, the try block will raise a NameError since 'name' is not defined. However, the except block will handle this error to prevent the program from crashing. We use the except block to tell the program what to do with the errors in the program.

```
try:
    print(name)
except NameError:
    print("name is not defined")
```

Output:

```
name is not defined
```

245.a

raise

In the code below, using the keyword raise, we are telling the code what kind of error to raise when x is less than 20 This will let the user know if they enter any number below 20.

```
x = 10
if x < 20:
    raise ValueError ("no numbers below 20")
```

Output:

```
raise ValueError ("no numbers below 20")
ValueError: no numbers below 20
```

246.b

A code that runs after a try block whether an exception is handled or not.

A finally block is the code that we want to run whether the error has been handled or not.
In the code below, we have handled the error with the except block.

```
try:
   print(name)
except NameError:
   print("x is not defined")
finally:
   print("End of code")
```

Output:

```
x is not defined
End of code
```

In the code below, we have not handled the error with the except block, but even with the error, the finally block still prints out "End of code". **So, the finally block is the code that we want to run whether we have handled our errors or not**.

```
try:
   print(name)
finally:
   print("End of code")
```

Output:

```
  print(name)
NameError: name 'name' is not defined
End of code
```

247.

To prevent the error from ruining our program, we can use the try and except keywords in the code.

```
try:
    print(x)
except NameError:
    print("x is not defined")
```

Output:

```
x is not defined
```

248.

```
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    print(a / b)
except ZeroDivisionError:
    print("You cannot divide a number by zero. Try again")
```

This is what happens when you try to divide a number by 0, it prints out you cannot divide the number by zero. This prevents the program from crashing.

Output:

```
You cannot divide a number by zero. Try again
```

249.

Add the finally block after the except block.

```
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    print(a / b)
except ZeroDivisionError:
    print("You cannot divide a number by zero")
finally:
    print("End of code")
```

If a = 45 and b = 5, below will be the output of the code.

Output:

```
9.0
End of code
```

250.

Here is the modified code below.

```
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    print(a / b)
except ZeroDivisionError:
    print("You cannot divide a number by zero."
          "Try again")
except ValueError:
    print("You have not entered a valid number."
          " Try again")
finally:
    print("End of code")
```

251.

We have put everything in a function. We use the while loop to ensure that the function only stops once it returns the answer.

```
def divide ():
    while True:
        try:
            a = int(input("Enter a number: "))
            b = int(input("Enter another number: "))
            return ("The answer is :" + str(a/b))
        except ZeroDivisionError:
            print("You cannot divide a number by zero."
                  "Please try again")
        except ValueError:
            print("You have not entered a "
                  "valid number. Please try again")
        finally:
            print("End of code")

print(divide())
```

252.

Here is the code below:

```
def your_age():
    while True:
        age = int(input("Enter your age : "))
        if age < 1:
            print("You cannot enter a number "
                  "that is less than one."
                  "Please try again")
        elif age > 150:
            print("You cannot enter a "
                  "number above 150."
                  " Please try again")
        else:
            return 'Your age is:' + str(age) + \
                   ' years old.'

print((your_age()))
```

253.b

A Python file that contains Python code.

254.b

Add .py at the end of the name.

Whatever name you give a module, just remember to add .py at the end.

255.

Here is how to import the module add.py

```
import add
```

256.c

A library is a collection of files with different functionalities, while a module is a file with single functionality & statement.

When you combine different modules, you create a library.

257.c

A package manager in Python.

Pip is used to manage and install a lot of python libraries.

258.

In windows type '**pip install pandas'** in the cmd line. You should have pip installed on your machine to successfully run this command. Pip is installed together with Python.

For mac run **'sudo pip install pandas'**.

259.

We are going to use the add method from the add module.

```
import add

x = 23
y = 12
print(add.add(23,12))
```

Output:

```
35
```

260.b

from add import sum.

To import a function from a module, we start with the keyword from.

261.c

import myfunction as mf.

262.a

dir()

Here is an example below. We will check the attributes and methods off add.py.

```
import add
print(dir(add))
```

Output:

```
['__builtins__', '__cached__', '__doc__', '__file__',
'__loader__', '__name__', '__package__', '__spec__', 'add']
```

263.

```
from datetime import time, date, datetime
```

264.

```
from datetime import datetime

x = datetime.now().strftime("%H:%M:%S")
print(x)
```

The time when I ran the code was:

Output:

```
21:43:42
```

265.

```
from datetime import datetime

x = datetime.now().strftime("%m/%d/%Y")
print(x)
```

Output:

```
09/10/2021
```

266.

This time we print out the time.

```
from datetime import datetime

x = datetime.today().strftime("%H:%M")
print(x)
```

Output:

```
10:42
```

267.

```
from datetime import date

x = date.today()
print(x)
```

Output:

```
2021-07-31
```

268.b

strftime()

This method allows us to format our time. We have used it in previous questions.

269.b

2021

When you pass the argument '%G' to the strftime() method, the output is a year.

270.a

Thursday

When you pass the argument '%A' to the strftime() method, the output is a day of the week.

271.

```
def age():
    a = input("Enter your year of birth:")
    b = datetime.today().strftime("%Y")
    your_age = int(b) - int(a)
    return "You are " + str(your_age) + " years old."
print(age())
```

If the user enters 1966 as their year of birth, below will be the output (Remember to import datetime module from datetime).

Output:

```
You are 55 years old.
```

272.c

An anonymous function or a function without a name.

While a normal function requires a name, a Lambda function does not require a name.

273.a

It can only take a limited number of arguments.

A lambda expression can take an unlimited number of arguments, but it can only have one expression.

274.b

432

This lambda expression has three arguments, a, b and c. However, it has one expression. The expression is what comes after the ":". In this expression, we are simply multiplying the three arguments by 2.

```
v = lambda a,b,c: a*b*c*2
print(v(12,3,6))
```

Output:

```
432
```

275.

Remember that a lambda function can only have one expression.

```
x = lambda a,b:(a+b)*2
print(x(10,15))
```

Output:

```
50
```

276.d

6

```
v = lambda a:a/2
print(int(v(12)))
```

Output:

```
6
```

277.c

0

This expression is simply finding the remainder of the division (Using the modulus %). In this case, the division of 12 by 2 gives no remainder, that's why the answer is 0**.**

```
v = lambda a:a%2
print(int(v(12)))
```

Output:

```
0
```

278.b

4.0

We are dividing 20 by 5 in this function.

```
def divide(num):
    return lambda y : y / num

divide1 = divide(5)
print(divide1(20))
```

Output:

```
4.0
```

279.

```
def divide(num):
    return lambda y: y / num

divide1 = divide(5)
print(divide1(50))
```

Output:

```
10.0
```

280.

```
results = [ 32, 45, 78, 67, 88, 97, 76, 84,
          44, 66, 63, 86, 90, 80]


# Using list, filter, and lambda to
marks = list(filter(lambda x: (x > 60),results))
print(len(marks))
```

We can see from the output that 11 people got over 60 marks.

Output

```
11
```

281.

```
results = [ 32, 45, 78, 67, 88, 97, 76, 84,
          44, 66, 63, 86, 90, 80]


# Using list, filter, and lambda
marks = list (filter (lambda x: (x < 50), results))
print(len(marks))
```

We can see from the output that 3 people got less than 50 marks.

Output:

```
3
```

282.b

It is structured into objects of similar properties and behaviors.

Everything in Python is some kind of an object. An object with its properties and functions. The class function is used to create an object with its properties and methods.

283.b

It means to create an object.

Simply put, creating a class in Python is creating an object with its own properties and methods. Class is a blueprint for creating objects in object-oriented programming (OOP).

284.

An instance of a class is simply an object that belongs to that class. Let's create a class called Animals. Remember a class is like a box or a container that holds objects. The objects that we will add are a dog, a cat, and a fish. Since a dog belongs to the animal class, we can say it is an instance of the animal class. A cat is an instance of the animal class. A fish is an instance of the animal class (meaning a fish belongs and uses attributes from the animal class).

285.a

class Car:

When giving your class a name, the name should always start with an upper letter.

286.d

print(Cat.i)

The variable i is a method of the class.

287.c

To assign values to class members.

When you create a class, the constructor will be used to instantiate or assign values to the members of the class.

288.a

```
class Car:
    def __init__(self,brand, model):
```

289.d

```
class Car:
    def __init__(self,brand, model):
        self.brand = brand
        self.model = model
```

290.

```
class Car:
    def __init__(self,brand, model):
        self.brand = brand
        self.model = model


car1 = Car('Nissan','Murano')
print(car1.model)
```

Output:

```
Murano
```

291.

```
class Car:
    def __init__(self,brand, model):
        self.brand = brand
        self.model = model


car1 = Car('Nissan','Morano')
car2 = Car('Ford','Focus')
print(car2.brand)
```

Output:

```
Ford
```

292.

```
class Car:
    def __init__(self,brand, model):
        self.brand = brand
        self.model = model


car1 = Car('Nissan','Morano')
car2 = Car('Ford','Focus')

# assign new values to the car2 object
car2.brand= "Honda"
car2.model= "Civic"
print(car2.brand)
```

Output:

```
Honda
```

293.

```
class Car:
    def __init__(self,brand, model):
        self.brand = brand
        self.model = model


car1 = Car('Nissan','Morano')
car2 = Car('Ford','Focus')

# assign new values to the car2 object
car2.brand= "Honda"
car2.model= "Civic"
# deleting car2
del car2
print(car2)
```

When we run the code, we get an error because car2 has been deleted and no longer exists.

Output:

```
        print(car2)
NameError: name 'car2' is not defined
```

294.b

John

You can put a function inside a class. The function is a method of the class. Here is an example of a class with a function. We assign the self as a parameter to the function so that the function can access the class variables.

```
class Car:
    def __init__(self,brand, model):
        self.brand = brand
        self.model = model

    def car_print(self):
        print( "My favorite car is : " + self.brand)


car1 = Car('Nissan','Morano')
print(car1.car_print())
```

Output:

```
My favorite car is: Nissan
```

295.b

It is called when an object gets destroyed.

Simply put, when you delete a class object using del(), the destructor __del__() is called to remove the garbage. The garbage in this case is the deleted object.

296.b

When one class inherits the methods and properties of another class.

In the code below, there are two classes, the parent class, and the child class. The child class (Dog class) is inheriting methods and properties of the parent class (Animals class).

```
# parent class
class Animals:
    def __init__(self, name, color)

# child class
class Dog(Animals):
    pass
```

297.b

Super()

In the example below, the Dog class is inheriting properties and methods from the Animals class by using the super() method.

```
class Dog(Animals):
    def __init__(self, name, color)
        super().__init__(name,color)
```

298.

In the code below, we have created a second object for the Father class.

```
class Father:
  def __init__(self, firstname,lastname, age):
      self.firstname = firstname
      self.lastname = lastname
      self.age = age

  def print_father(self):
      print("Your first name is  {} and your last name is {}. You are {} years old".
        format(self.firstname, self.lastname, self.age))



# creating the first object
father1 = Father("Richard", "Kelly", 42)
# creating the second object
father2 = Father("Benson", "Carter", 52)

print(father2.print_father())
```

Output:

```
Your first name is Benson and your last name is Carter. You are 52 years old
```

299.a

```
class Son(Father):
```

When creating a child class, we put the name of the parent class in the brackets. We basically pass the parent class as an argument.

300.

```
class Father:
    def __init__(self,firstname,lastname,age):
        self.firstname = firstname
        self.lastname = lastname
        self.age = age

    def print_father(self):
        print("Your first name is : {} and your last name is : {}. You are: {} years old.".
            format(self.firstname, self.lastname, self.age))


# creating a child class
class Son(Father):
    pass


# creating an object for the child class
son1 = Son("Shawn", "Kelly", 21)
# printing the son1 using the print_father function
print(son1.print_father())
```

Output:

```
Your first name is  Shawn and your last name is  Kelly. You are 21 years old.
```

301.

```
class Father:
    def __init__(self,firstname,lastname,age):
        self.firstname = firstname
        self.lastname = lastname
        self.age = age

    def print_father(self):
        print("Your first name is: {} and your last name is "
              ": {}. You are: {} years old.".
              format(self.firstname, self.lastname, self.age,))


class Son(Father):
    def __init__(self, firstname, lastname, age, height):

     # using the name of the parent to inherit its properties
        Father.__init__(self, firstname, lastname, age)
        self.height = height

    def print_son(self):
        print("Your first name is {} and your last name is {} "
              "you are {} years old and your height is {} ."
              format(self.firstname, self.lastname,self.age,self.height))


# creating the son1 object
son1 = Son('Shawn', 'Kelly', 21, "5'9")
# print the son one using the print son function
print(son1.print_son())
```

Output:

```
Your first name is Shawn and your last name is Kelly you are
21 years old and your height is 5'9.
```

302.c

open()

303.

We can open the file using the open() method. The open method takes two arguments, the name of the file we want to open and the mode. The mode is the state we want to open the file in. In our case, we want to open the file so we can read it. So, the mode is 'r' for read.

```
with open('names', 'r') as x:
    print(x.read())
```

304.

To create a file we use the open() method. We have to use a different mode called x. This will create a file and add an Elephant to it.

```
with open('animals.txt', 'x') as x:
    x.write("Elephant")
```

We can also use the mode 'w' and if that file does not exist, it will create one. See below**.**

```
with open('animals.txt', 'w') as x:
    x.write("Elephant")
```

305.

To add another line to the file, we can use the open() method and the append 'a' mode.

```
with open('animals.txt', 'a') as x:
# Using escape characters to write in separate lines
   x.writelines(["\nLion","\nTiger"])
```

Let's check below if the information has been appended correctly to the file. We can open the file in read mode.

```
with open("animals.txt", 'r') as x :
   print(x.read())
```

Output:

```
Elephant
Lion
Tiger
```

306.a

It will overwrite an existing file.

If there is an existing file name and you use the 'w' mode, it will get overwritten**.**

307.

Below is a code you can use to create and write to the file. If you are using the 'with' statement, you do not need to use the 'close' statement at the end of your code. The 'with' statement will close the code for you.

```
with open('my_first_file.txt', 'w') as x:
    x.write("This is my first file")
```

To read the file we just created, we can use the code below.

```
with open('my_first_file.txt', 'r') as x:
    print(x.read())
```

Output:

```
This is my first file
```

308.

Here is the code below to append a line to the file. Remember to change the mode to 'a' for append.

```
with open('my_first_file.txt', 'a') as x:
    x.write('\nThis is my last line')
```

Let's open the file now.

```
with open('my_first_file.txt', 'r') as x:
    print(x.read())
```

You can see from the output below that a line has been added to our file.

Output:

```
This is my first file
This is my last line
```

309.

To delete a file, you have to import a file os module. The module will first check if the file exists and then it will delete it or remove it.

```
import os

os.path.exists("my_first_file.txt")
os.remove("my_first_file.txt")
```

310.

Here is the code below**:**

```
a = [1, 2, 4, 6, 7, 9, 11, 17, 23, 26, 27]

#  Create an empty list to append odd numbers
odd_numbers = []
for number in a:
    if number % 2 != 0:
        odd_numbers.append(number)
print(odd_numbers)
```

Output:

```
[1, 7, 9, 11, 17, 23, 27]
```

311.

```
a = [1, 2, 4, 6, 7, 9, 11, 17, 23, 26, 27]
# we create an empty list to append even numbers
even_numbers =[]
for number in a:
    if number % 2 == 0:
        even_numbers.append(number)
print(even_numbers)
```

Output:

```
[2, 4, 6, 26]
```

312.

```
c = [1, 1, 3, 5, 6, 1, 3, 6, 7, 8, 9, 6, 8, 9,]
duplicates = []
for number in c:
    if c.count(number) > 1:
        if number not in duplicates:
            duplicates.append(number)
print(duplicates)
```

Output:

```
[1, 3, 6, 8, 9]
```

313.

```
def convert_to_min():
    while True:
        try:
            hrs = input("Enter the number of hours:")
            minutes = int(hrs) * 60
        except ValueError:
            print('Please enter a valid number')
        else:
            print("The number of minutes is:" + str(minutes))
            break

convert_to_min()
```

314.

Here is a simple calculator code below.

```
def calc():
    while True:
        try:
            a = int(input("Enter a number"))
            b = input("Enter an operator")
            c = int(input("Enter another number"))
            if b == '+':
                add = a + c
                return add
            elif b == "-":
                subtract = a - c
                return subtract
            elif b == "*":
                multiply = a * c
                return multiply
            elif b == "/":
                division = a/c
                return division
        except ValueError:
            print('Please enter a valid number')
        except ZeroDivisionError:
            print('You cannot divide a number by zero. Try again')


print(calc())
```

315.

Here is the code below:

```
import random

a = random.randint(0,10)

def guess_num():
    while True:
        guess = int(input("Guess a number between 0 and 10 :"))
        if guess > a:
            print('Your number is too high.')
        elif guess < a:
            print("Your number is too low.")
        else:
            print("Correct. You have won.")
            break

guess_num()
```

316.

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,12]

total = 0
for num in a:
    total = total + num
print(total)
```

Output:

```
78
```

317.

```
str = 'This Is noT how wE writE A sentence'

con1 = []
for letter in str:
    if (letter.isupper()) == True:
        con1.append(letter)
print(len(con1))
```

Output:

```
6
```

318.

We use the sum() function to add the numbers in the dictionary.

```
a = {"Men": 20,"women": 25}
b = {"children": 23,"students": 44}

c = a.values()
d = sum(c)
e = b.values()
f = sum(e)
total = d + f
print(total)
```

Output:

```
112
```

319.

```
a = [17, 23, 13, 24, 25, 33, 71, 81, 30]
odd = []
for num in a:
    if num % 2 != 0:
        odd.append(num)
print(min(odd))
```

Output:

```
13
```

320.

```
def center(a):
    length = len(a)
    middle = length//2
    if len(a)%2==1:
        return a[middle]
    else:
        return "No middle character"
print(center("welcome"))
```

Output:

```
C
```

321.

Remember that they are different ways to solve this challenge. This is just one of them.

```
names = ['Peter', 'Peter', 'Peter', "Peter"]

def all_same(a):
    same = all(item == a[0] for item in a)
    if (same):
        return 'The names are the same'
    else:
        return 'The names are not the same.'
print(all_same(names))
```

Passing the argument 'names' into the function, we get the following outcome.

Output:

```
The names are the same.
```

What is Next?

Once you have mastered the basics of Python language, you are ready to tackle advanced uses of the Python language. Where do you go from here? What things can you do with your Python knowledge? Python is a general-purpose programming language that can be used in a variety of areas in programming.

Web Development

Popular web development frameworks such as Django and Flask are written in Python. Once you master the basics of Python you can start learning these frameworks to learn about web applications.

Gaming

Python is widely used in the gaming industry. If you are interested in game development, you can explore popular gaming frameworks such as Pygame and many more to create games.

Data Science and Analysis

Python is the number one language used in data science and machine learning and data analytics and visualizations. Some of the most popular data science libraries such as NumPy, Pandas, TensorFlow, Scikit-Learn, Pytorch, Matplotlib, and many more are written in python. Once you master the basics of Python, you can start exploring these libraries if you have an interest in data science.

Automation and Scripting

Python is also widely used in scripting to automate tasks that require repetition like sending emails and downloading files.

These are some of the areas you can explore once you master the basics of Python. Remember, there is so much to learn about Python and it all starts with mastering the basics. **Practice Python.**

Acknowledgment

Massive thanks to Andrew for the great cover design and other invaluable contributions to the writing of the book.

About Author

The author is a student of life, a father, a husband, a finance and programming enthusiast. For queries, comments, and suggestions, you can reach him at: benjaminbennettalexander@gmail.com