

basics\basics.py

```
1  # Python Basics
2  # Mehedi Hasan
3  # mehedihasan9339@gmail.com
4  # ===== #
5
6
7  # How to write comments in Python
8
9  """
10 This
11 is
12 a
13 multi-line
14 comment
15 """
16
17 # Type conversion examples
18 a = int(5) # Converting the number 5 to an integer
19 b = str(5) # Converting the number 5 to a string
20 c = float(5) # Converting the number 5 to a float
21
22 # Printing the values of the variables
23 print(a) # Prints the integer value
24 print(b) # Prints the string value
25 print(c) # Prints the float value
26
27 # Printing the data types of the variables
28 print(type(a)) # Shows the type of 'a' (int)
29 print(type(b)) # Shows the type of 'b' (str)
30 print(type(c)) # Shows the type of 'c' (float)
31
32 # Assigning values to variables
33 a = 5 # Integer assignment
34 A = "Mehedi" # String assignment
35
36 # Printing the values of the variables
37 print(a) # Prints the value of 'a' (5)
38 print(A) # Prints the value of 'A' ("Mehedi")
39
40 # Multiple variable assignment
41 x, y, z = "Orange", "Banana", "Cherry" # Assigning multiple values to variables in a
single line
42
43 # Printing the values of the variables
44 print(x) # Prints the value of 'x' ("Orange")
45 print(y) # Prints the value of 'y' ("Banana")
46 print(z) # Prints the value of 'z' ("Cherry")
47
48
49 x = y = z = "Orange" # Assigning the same value to multiple variables in a single line
50 print(x) # Prints the value of 'x' ("Orange")
51 print(y) # Prints the value of 'y' ("Orange")
```

```

52 print(z) # Prints the value of 'z' ("Orange")
53
54 # Global variables
55 x = "awesome" # Declaring a global variable
56 def my_function(): # Defining a function
57     print("Python is " + x) # Accessing the global variable inside the function
58 my_function() # Calling the function (Python is awesome)
59
60 # Local variables
61 def my_function(): # Defining a function
62     y = "fantastic" # Declaring a local variable
63     print("Python is " + y) # Accessing the local variable inside the function
64 my_function() # Calling the function (Python is fantastic)
65
66 # Global and local variables
67 x = "awesome" # Declaring a global
68 def my_function(): # Defining a function
69     x = "fantastic" # Declaring a local variable with the same name as the global variable
70     print("Python is " + x) # Accessing the local variable inside the function
71 my_function() # Calling the function (Python is fantastic)
72 print("Python is " + x) # Accessing the global variable (Python is awesome)
73
74 # Using the global keyword to change a global variable inside a function
75 x = "awesome" # Declaring a global variable
76 def my_function(): # Defining a function
77     global x # Using the global keyword to access the global variable
78     x = "fantastic" # Changing the value of the global variable
79 my_function() # Calling the function (Python is fantastic)
80 print("Python is " + x) # Accessing the global variable (Python is fantastic)
81
82 # Unpack a collection
83 fruits = ["apple", "banana", "cherry"] # Creating a list
84 x, y, z = fruits # Unpacking the list into three variables
85 print(x) # Prints the value of 'x' ("apple")
86 print(y) # Prints the value of 'y' ("banana")
87 print(z) # Prints the value of 'z' ("cherry")
88
89 # Output of multiple variables
90 x = "Python"
91 y = "is"
92 z = "awesome"
93 print(x, y, z) # Prints the values of the variables separated by a space ("Python is
    awesome")
94
95 # Output of multiple variables with a separator
96 x = "Python"
97 y = "is"
98 z = "awesome"
99 print(x, y, z, sep="-") # Prints the values of the variables separated by a hyphen
    ("Python-is-awesome")
100
101 # Output of multiple variables with an end parameter
102 x = "Python"
103 y = "is"

```

```
104 z = "awesome"
105 print(x, end=" ") # Prints the value of 'x' followed by a space ("Python ")
106
107
108 # Output of multiple variables with a plus operator
109 x = "Python "
110 y = "is "
111 z = "awesome"
112 print(x + y + z) # Prints the values of the variables concatenated together ("Python is
    awesome")
113
114 # Output of multiple variables with a plus operator (it works as a math operator)
115 x = 5
116 y = 10
117 print(x + y) # Prints the sum of 'x' and 'y' (15)
118
119 # Output of multiple variables separates with a comma
120 x = 5
121 y = "John"
122 print(x, y) # Prints the values of 'x' and 'y' separated by a comma (5 John)
123
124 # Python data types
125 # Integers
126 x = 5
127 print(type(x)) # Shows the type of 'x' (int)
128 # Floats
129 x = 5.5
130 print(type(x)) # Shows the type of 'x' (float)
131 # Complex numbers
132 x = 1j
133 print(type(x)) # Shows the type of 'x' (complex)
134 # Strings
135 x = "Hello, World!"
136 print(type(x)) # Shows the type of 'x' (str)
137 # Lists
138 x = ["apple", "banana", "cherry"]
139 print(type(x)) # Shows the type of 'x' (list)
140 # Tuples
141 x = ("apple", "banana", "cherry")
142 print(type(x)) # Shows the type of 'x' (tuple)
143 # Dictionaries
144 x = {"name" : "John", "age" : 36}
145 print(type(x)) # Shows the type of 'x' (dict)
146 # Sets
147 x = {"apple", "banana", "cherry"}
148 print(type(x)) # Shows the type of 'x' (set)
149 # Booleans
150 x = True
151 print(type(x)) # Shows the type of 'x' (bool)
152 # Bytes
153 x = b"Hello"
154 print(type(x)) # Shows the type of 'x' (bytes)
155 # Byte Arrays
156 x = bytearray(5)
```

```
157 print(type(x)) # Shows the type of 'x' (bytearray)
158 # Memory Views
159 x = memoryview(bytes(5))
160 print(type(x)) # Shows the type of 'x' (memoryview)
161
162
163 # Python Numbers
164 # Integers
165 x = 1
166 print(type(x)) # Shows the type of 'x' (int)
167 # Floats
168 x = 1.1
169 print(type(x)) # Shows the type of 'x' (float)
170 # Complex numbers
171 x = 1j
172 print(type(x)) # Shows the type of 'x' (complex)
173
174 # Python Casting
175 # Integers
176 x = int(1)
177 print(type(x)) # Shows the type of 'x' (int)
178 # Floats
179 x = float(1)
180 print(type(x)) # Shows the type of 'x' (float)
181 # Strings
182 x = str(1)
183 print(type(x)) # Shows the type of 'x' (str)
184
185 # Python Strings
186 # Strings
187 x = "Hello, World!"
188 print(x) # Prints the string (Hello, World!)
189
190 # Multiline Strings
191 x = """Lorem ipsum dolor sit amet,
192 consectetur adipiscing elit,
193 sed do eiusmod tempor incididunt ut labore et dolore magna aliqua."""
194 print(x) # Prints the multiline string
195
196 # Strings are Arrays
197 x = "Hello, World!"
198 print(x[1]) # Prints the second character of the string (e)
199
200 # Slicing
201 x = "Hello, World!"
202 print(x[2:5]) # Prints the characters from position 2 to 5 (not included) (llo)
203
204 # Negative Indexing
205 x = "Hello, World!"
206 print(x[-1]) # Prints the last character of the string (!)
207
208 # String Length
209 x = "Hello, World!"
210 print(len(x)) # Prints the length of the string (13)
```

```
211
212 # String Methods
213 x = " Hello, World! "
214 print(x.strip()) # Removes any whitespace from the beginning or the end of the string
(Hello, World!)
215 print(x.lower()) # Converts a string into lowercase (hello, world!)
216 print(x.upper()) # Converts a string into uppercase (HELLO, WORLD!)
217 print(x.replace("H", "J")) # Replaces a string with another string (Jello, World!)
218 print(x.split(",")) # Splits the string into substrings if it finds instances of the
separator ([' Hello', ' World! '])
219
220 # Check String
221 x = "Hello, World!"
222 print("World" in x) # Checks if a certain phrase is present in the string (True)
223
224 # String Concatenation
225 x = "Hello"
226 y = "World"
227 z = x + y
228 print(z) # Prints the concatenated string (HelloWorld)
229
230 # String Format
231 age = 36
232 name = "Mehedi"
233 txt = "My name is {} and I am {}"
234 print(txt.format(name, age)) # Formats the string with the variables (My name is Mehedi and
I am 36)
235
236 # Escape Characters
237 txt = "We are the so-called \"Vikings\" from the north."
238 print(txt) # Prints the string with escape characters (We are the so-called "Vikings" from
the north.)
239
240 # String Methods
241 a = "Hello, World!"
242 print(a[1]) # Returns the character at position 1 (e)
243 print(a[2:5]) # Returns the characters from position 2 to 5 (not included) (llo)
244 print(a.strip()) # Removes any whitespace from the beginning or the end (Hello, World!)
245 print(a.lower()) # Converts a string into lowercase (hello, world!)
246 print(a.upper()) # Converts a string into uppercase (HELLO, WORLD!)
247 print(a.replace("H", "J")) # Replaces a string with another string (Jello, World!)
248 print(a.split(",")) # Splits the string into substrings if it finds instances of the
separator (['Hello', ' World!'])
249
250 # Check String
251 txt = "The rain in Spain stays mainly in the plain"
252 x = "ain" in txt
253 print(x) # Returns True if the phrase is present in the string, otherwise False (True)
254
255 # String Concatenation
256 a = "Hello"
257 b = "World"
258 c = a + b
259 print(c) # Concatenates two strings (HelloWorld)
```

```
260
261 # Python Booleans
262 # Booleans
263 print(10 > 9) # Returns True because 10 is greater than 9 (True)
264 print(10 == 9) # Returns False because 10 is not equal to 9 (False)
265 print(10 < 9) # Returns False because 10 is not less than 9 (False)
266
267 # Evaluate Values and Variables
268 print(5 > 9) # Returns False because 5 is not greater than 9 (False)
269
270 # Python Operators
271 # Arithmetic Operators
272 x = 5
273 y = 3
274 print(x + y) # Addition (8)
275 print(x - y) # Subtraction (2)
276 print(x * y) # Multiplication (15)
277 print(x / y) # Division (1.6666666666666667)
278 print(x % y) # Modulus (Remainder) (2)
279 print(x ** y) # Exponentiation (Power) (125)
280 print(x // y) # Floor division (1)
281
282 # Assignment Operators
283 x = 5
284 y = 10
285 print(x + y) # Addition (15)
286 x += 3
287 print(x) # Addition (8)
288 x -= 3
289 print(x) # Subtraction (5)
290
291 # Comparison Operators
292 x = 5
293 y = 3
294 print(x == y) # Equal to (False)
295 print(x != y) # Not equal to (True)
296 print(x > y) # Greater than (True)
297 print(x < y) # Less than (False)
298 print(x >= y) # Greater than or equal to (True)
299 print(x <= y) # Less than or equal to (False)
300
301 # Logical Operators
302 x = 5
303 y = 3
304 print(x > 2 and x < 10) # Returns True because 5 is greater than (True)
305 print(x > 2 or x < 4) # Returns True because one of the conditions are True (True)
306 print(not(x > 2 and x < 10)) # Reverse the result, returns False (False)
307
308 # Identity Operators
309 x = 5
310 y = 5
311 print(x is y) # Returns True because both variables are the same object (True)
312 print(x is not y) # Returns False because both variables are the same object (False)
313
```

```

314 # Membership Operators
315 x = 5
316 y = 3
317 list = [1, 2, 3, 4, 5]
318 print(x in list) # Returns True because a sequence with the value is present in the list
    (True)
319 print(y not in list) # Returns True because a sequence with the value is not present in the
    list (True)
320
321 # Bitwise Operators
322 x = 5
323 y = 3
324 print(x & y) # Bitwise AND (1)
325 print(x | y) # Bitwise OR (7)
326 print(x ^ y) # Bitwise XOR (6)
327 print(~x) # Bitwise NOT (-6)
328 print(x << 2) # Bitwise left shift (20)
329 print(x >> 2) # Bitwise right shift (1)
330
331 # Python Lists
332 # Lists
333 thislist = ["apple", "banana", "cherry"]
334 print(thislist) # Prints the list (['apple', 'banana', 'cherry'])
335
336 # List Indexing
337 print(thislist[0]) # Prints the first item in the list (apple)
338
339 # Negative Indexing
340 print(thislist[-1]) # Prints the last item in the list (cherry)
341
342 # Range of Indexes
343 print(thislist[1:3]) # Prints the second and third items in the list (['banana', 'cherry'])
344
345 # Change Item Value
346 thislist[1] = "blackcurrant"
347 print(thislist) # Changes the second item in the list (['apple', 'blackcurrant', 'cherry'])
348
349 # Loop Through a List
350 for x in thislist:
351     print(x) # Prints each item in the list (apple, blackcurrant, cherry)
352
353 # Check if Item Exists
354 if "apple" in thislist:
355     print("Yes, 'apple' is in the list") # (Yes, 'apple' is in the list)
356
357 # List Length
358 print(len(thislist)) # Prints the number of items in the list (3)
359
360 # Add Items
361 thislist.append("orange")
362 print(thislist) # Adds an item to the end of the list (['apple', 'blackcurrant', 'cherry',
    'orange'])
363
364 # Add Items at a Specific Index

```

```
365 thislist.insert(1, "orange")
366 print(thislist) # Adds an item at the specified index (['apple', 'orange', 'blackcurrant',
    'cherry'])
367
368 # Remove Item
369 thislist.remove("banana")
370 print(thislist) # Removes the specified item (['apple', 'blackcurrant', 'cherry'])
371
372 # Remove Item by Index
373 thislist.pop(1)
374 print(thislist) # Removes the specified index (['apple', 'cherry'])
375
376 # Empty the List
377 thislist.clear()
378 print(thislist) # Clears the list ([])
379
380 # Copy a List
381 mylist = thislist.copy()
382 print(mylist) # Copies the list ([])
383
384 # Join Two Lists
385 list1 = ["a", "b", "c"]
386 list2 = ["d", "e", "f"]
387 list3 = list1 + list2
388 print(list3) # Joins two lists (['a', 'b', 'c', 'd', 'e', 'f'])
389
390 # Append List2 to List1
391 list1 = ["a", "b", "c"]
392 list1.extend(list2)
393 print(list1) # Appends list2 to list1 (['a', 'b', 'c', 'd', 'e', 'f'])
394
395 # List Methods
396 # List Methods
397 thislist = ["apple", "banana", "cherry"]
398 thislist.append("orange") # Adds an item to the end of the list
399 thislist.insert(1, "orange") # Adds an item at the specified index
400 thislist.remove("banana") # Removes the specified item
401 thislist.pop(1) # Removes the specified index
402 thislist.clear() # Clears the list
403 mylist = thislist.copy() # Copies the list
404 list1 = ["a", "b", "c"]
405 list2 = ["d", "e", "f"]
406 list1.extend(list2) # Appends list2 to list1
407 # List Comprehension
408 fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
409 newlist = [x for x in fruits if "a" in x]
410 print(newlist) # Returns the fruits containing the letter "a" (['apple', 'banana',
    'cherry'])
411
412 # Python Tuples
413 # Tuples
414 thistuple = ("apple", "banana", "cherry")
415 print(thistuple) # Prints the tuple (['apple', 'banana', 'cherry'])
416
```



```
417 # Tuple Indexing
418 print(thistuple[1]) # Prints the second item in the tuple (banana)
419
420 # Negative Indexing
421 print(thistuple[-1]) # Prints the last item in the tuple (cherry)
422
423 # Range of Indexes
424 print(thistuple[1:3]) # Prints the second and third items in the tuple (['banana',
    'cherry'])
425
426 # Change Tuple Values
427 thistuple = ("apple", "banana", "cherry")
428 y = list(thistuple)
429 y[1] = "kiwi"
430 thistuple = tuple(y)
431 print(thistuple) # Changes the second item in the tuple (['apple', 'kiwi', 'cherry'])
432
433 # Loop Through a Tuple
434 thistuple = ("apple", "banana", "cherry")
435 for x in thistuple:
436     print(x) # Prints each item in the tuple (apple, banana, cherry)
437
438 # Check if Item Exists
439 thistuple = ("apple", "banana", "cherry")
440 if "apple" in thistuple:
441     print("Yes, 'apple' is in the tuple") # (Yes, 'apple' is in the tuple)
442
443 # Tuple Length
444 thistuple = ("apple", "banana", "cherry")
445 print(len(thistuple)) # Prints the number of items in the tuple (3)
446
447 # Add Items
448 thistuple = ("apple", "banana", "cherry")
449 y = list(thistuple)
450 y.append("orange")
451 thistuple = tuple(y)
452 print(thistuple) # Adds an item to the end of the tuple (['apple', 'banana', 'cherry',
    'orange'])
453
454 # Remove Items
455 thistuple = ("apple", "banana", "cherry")
456 y = list(thistuple)
457 y.remove("apple")
458 thistuple = tuple(y)
459 print(thistuple) # Removes the specified item (['banana', 'cherry'])
460
461 # Delete Tuple
462 thistuple = ("apple", "banana", "cherry")
463 del thistuple # Deletes the tuple completely
464
465 # Join Two Tuples
466 tuple1 = ("a", "b", "c")
467 tuple2 = ("d", "e", "f")
468 tuple3 = tuple1 + tuple2
```

```
469 print(tuple3) # Joins two tuples (('a', 'b', 'c', 'd', 'e', 'f'))
470
471 # Tuple Methods
472 # Tuple Methods
473 # count()
474 # index()
475
476 # Python Sets
477 # Sets (unordered and unindexed)
478 thisset = {"apple", "banana", "cherry"}
479 print(thisset) # Prints the set (unordered) ({'apple', 'banana', 'cherry'})
480
481 # Access Items
482 for x in thisset:
483     print(x) # Prints each item in the set (apple, banana, cherry)
484
485 # Check if Item Exists
486 if "apple" in thisset:
487     print("Yes, 'apple' is in the set") # (Yes, 'apple' is in the set)
488
489 # Add Items
490 thisset.add("orange")
491 print(thisset) # Adds an item to the set ({'apple', 'banana', 'cherry', 'orange'})
492
493 # Add Multiple Items
494 thisset.update(["orange", "mango", "grapes"])
495 print(thisset) # Adds multiple items to the set ({'apple', 'banana', 'cherry', 'orange',
'mango', 'grapes'})
496
497 # Get the Length of a Set
498 thisset = {"apple", "banana", "cherry"}
499 print(len(thisset)) # Prints the number of items in the set (3)
500
501 # Remove Item
502 thisset.remove("banana")
503 print(thisset) # Removes the specified item ({'apple', 'cherry'})
504
505 # Remove Item with discard()
506 thisset.discard("banana")
507 print(thisset) # Removes the specified item ({'apple', 'cherry'})
508
509 # Remove the Last Item
510 x = thisset.pop()
511 print(x) # Removes the last item in the set (apple)
512
513 # Empty the Set
514 thisset.clear()
515 print(thisset) # Removes all items from the set (set())
516
517 # Delete the Set
518 del thisset # Deletes the set completely
519
520 # Join Two Sets
521 set1 = {"a", "b", "c"}
```

```
522 set2 = {1, 2, 3}
523 set3 = set1.union(set2)
524 print(set3) # Joins two sets ({1, 2, 3, 'a', 'b', 'c'})
525
526 # Join Two Sets with update()
527 set1 = {"a", "b", "c"}
528 set2 = {1, 2, 3}
529 set1.update(set2)
530 print(set1) # Joins two sets ({1, 2, 3, 'a', 'b', 'c'})
531
532 # Set Methods
533 # add()
534 # clear()
535 # copy()
536 # discard()
537 # difference()
538 # difference_update()
539 # discard()
540 # intersection()
541 # intersection_update()
542 # isdisjoint()
543 # issubset()
544 # issuperset()
545 # pop()
546 # remove()
547 # symmetric_difference()
548 # symmetric_difference_update()
549 # union()
550 # update()
551
552 # Python Dictionaries
553 # Dictionaries
554 thisdict = {
555     "brand": "Ford",
556     "model": "Mustang",
557     "year": 1964
558 }
559 print(thisdict) # Prints the dictionary ({'brand': 'Ford', 'year': 1964, 'model': 'Mustang'})
560
561 # Accessing Items
562 x = thisdict["model"]
563 print(x) # Accesses the value of the specified key (Mustang)
564
565 # Get the Value of the "model" Key
566 x = thisdict.get("model")
567 print(x) # Accesses the value of the specified key (Mustang)
568
569 # Change Values
570 thisdict["year"] = 2018
571 print(thisdict) # Changes the value of the specified key ({'brand': 'Ford', 'year': 2018, 'model': 'Mustang'})
572
573 # Loop Through a Dictionary
574 for x in thisdict:
```

```
575     print(x) # Prints all key names in the dictionary
576 for x in thisdict:
577     print(thisdict[x]) # Prints all values in the dictionary
578 for x in thisdict.values():
579     print(x) # Prints all values in the dictionary
580 for x, y in thisdict.items():
581     print(x, y) # Prints all key-value pairs in the dictionary
582
583 # Check if Key Exists
584 if "model" in thisdict:
585     print("Yes, 'model' is present in the dictionary") # Checks if the specified key is
present in the dictionary (Yes, 'model' is present in the dictionary)
586
587 # Dictionary Length
588 print(len(thisdict)) # Prints the number of items in the dictionary (3)
589
590 # Adding Items
591 thisdict["color"] = "red"
592 print(thisdict) # Adds a new key-value pair to the dictionary ({'brand': 'Ford', 'year':
2018, 'model': 'Mustang', 'color': 'red'})
593
594 # Removing Items
595 thisdict.pop("model")
596 print(thisdict) # Removes the specified key ({'brand': 'Ford', 'year': 2018, 'color':
'red'})
597
598 # Copying a Dictionary
599 thisdict = thisdict.copy()
600 print(thisdict) # Copies the dictionary ({'brand': 'Ford', 'year': 2018, 'color': 'red'})
601
602 # Nested Dictionaries
603 thisdict = {
604     "name": "John",
605     "age": 36,
606     "city": {
607         "country": "USA",
608         "state": "California"
609     }
610 }
611 print(thisdict) # Creates a nested dictionary ({'name': 'John', 'age': 36, 'city':
{'country': 'USA', 'state': 'California'}})
612
613 # Dictionary Methods
614 # clear()
615 # copy()
616 # fromkeys()
617 # get()
618 # items()
619 # keys()
620 # pop()
621 # popitem()
622 # setdefault()
623 # update()
624 # values()
```

```
625
626 # Python If...Else
627 # If Statement
628 a = 33
629 if a > 10:
630     print("a is greater than 10") # Checks if the condition is true
631
632 # If-Else Statement
633 a = 33
634 b = 200
635 if b > a:
636     print("b is greater than a")
637 else:
638     print("b is not greater than a") # Checks if the condition is true
639
640 # Elif Statement
641 a = 33
642 b = 33
643 if b > a:
644     print("b is greater than a")
645 elif a == b:
646     print("a and b are equal")
647
648 # Else Statement
649 a = 33
650 b = 200
651 if b > a:
652     print("b is greater than a")
653 else:
654     print("b is not greater than a")
655
656 # Short Hand If
657 a = 200
658 b = 33
659 print("A") if a > b else print("B") # Checks if the condition is
660
661 # If-Else Ladder
662 a = 33
663 b = 33
664 if b > a:
665     print("b is greater than a")
666 elif a == b:
667     print("a and b are equal")
668 else:
669     print("a is greater than b")
670
671 # And
672 a = 33
673 b = 33
674 if a > b and b == a:
675     print("Both conditions are True") # (True)
676
677 # Or
678 a = 33
```

```
679 b = 33
680 if a > b or b == a:
681     print("At least one condition is True") # (True)
682
683 # Nested If
684 a = 33
685 b = 33
686 if a > b:
687     print("a is greater than b")
688     if a == b:
689         print("a and b are equal")
690
691 # Pass Statement
692 a = 33
693 b = 33
694 if b > a:
695     pass # Does nothing
696 else:
697     print("b is not greater than a")
698
699 # Python While Loops
700 # While Loop
701 i = 1
702 while i < 6:
703     print(i)
704     i += 1
705
706 # Break Statement
707 i = 1
708 while i < 6:
709     if i == 3:
710         break
711     print(i)
712     i += 1
713
714 # Continue Statement
715 i = 0
716 while i < 6:
717     if i == 3:
718         i += 1
719         continue
720     print(i)
721     i += 1
722
723 # Else Statement
724 i = 1
725 while i < 6:
726     print(i)
727     i += 1
728 else:
729     print("i is no longer less than 6")
730
731 # Nested Loops
732 adj = ["red", "big", "tasty"]
```

```
733 fruits = ["apple", "banana", "cherry"]
734 for x in adj:
735     for y in fruits:
736         print(x, y)
737
738 # Pass Statement
739 for x in [
740     "apple",
741     "banana",
742     "cherry"
743 ]:
744     pass
745
746 # Python For Loops
747 # For Loop
748 # For Loop with Range
749 for x in range(6):
750     print(x)
751
752 # Nested Loops
753 adj = ["red", "big", "tasty"]
754 fruits = ["apple", "banana", "cherry"]
755 for x in adj:
756     for y in fruits:
757         print(x, y)
758
759 # Break Statement
760 fruits = ["apple", "banana", "cherry"]
761 for x in fruits:
762     if x == "banana":
763         break
764
765 # Continue Statement
766 i = 0
767 while i < 6:
768     i += 1
769     if i == 3:
770         continue
771     print(i)
772
773 # Else Statement
774 for x in range(6):
775     print(x)
776 else:
777     print("Finally finished!")
778
779 # Pass Statement
780 for x in [
781     "apple",
782     "banana",
783     "cherry"
784 ]:
785     pass
786
```

```
787 # Python Functions
788 # Creating a Function
789 def greet(name):
790     print("Hello, " + name)
791
792 # Calling a Function
793 greet("John") # (Hello John)
794
795 # Return Values
796 def greet(name):
797     return "Hello, " + name
798 print(greet("John")) # (Hello John)
799
800 # Lambda Functions
801 def greet(name):
802     return "Hello, " + name
803 print(greet("John"))
804 # Lambda Function
805 x = lambda a: a + 10 # (15)
806
807 # Python Lambda
808 # Lambda Function
809 x = lambda a: a + 10
810 print(x(5)) # (15)
811
812 # Lambda Function with Multiple Arguments
813 x = lambda a, b: a + b
814 print(x(5, 10)) # (15)
815
816 # Lambda Function with Multiple Arguments
817 x = lambda a, b, c: a + b + c
818 print(x(5, 10, 15)) # (30)
819
820 # Lambda Function with Default Argument
821 x = lambda a, b, c=1: a + b + c
822 print(x(5, 10)) # (16)
823
824 # Python Classes and Objects
825 # Creating a Class
826 class Dog:
827     def __init__(self, name, age):
828         self.name = name
829         self.age = age
830     def bark(self):
831         print("Woof!")
832
833 # Creating an Object
834 class Dog:
835     def __init__(self, name, age):
836         self.name = name
837         self.age = age
838     def bark(self):
839         print("Woof!")
840
```



```

841 # Python Inheritance
842 class Animal:
843     def __init__(self, name, age):
844         self.name = name
845         self.age = age
846     def bark(self):
847         print("Woof!")
848 class Dog(Animal):
849     def __init__(self, name, age):
850         super().__init__(name, age)
851     def bark(self):
852         print("Woof!")
853
854 # Python Iterators
855 # Creating an Iterator
856 class MyIterator:
857     def __init__(self, data):
858         self.data = data
859         self.index = 0
860     def __iter__(self):
861         return self
862     def __next__(self):
863         if self.index < len(self.data):
864             result = self.data[self.index]
865             self.index += 1
866             return result
867         else:
868             raise StopIteration()
869
870 # Using the Iterator
871 data = ["apple", "banana", "cherry"]
872 my_iterator = MyIterator(data)
873 for item in my_iterator:
874     print(item)
875
876 # Python Modules
877 # Importing a Module
878 import math
879 print(math.pi)
880 # Using the Module
881 # Creating a Module
882 def greet(name):
883     print("Hello, " + name)
884
885 # Python Dates
886 # Importing the datetime Module
887 import datetime
888 x = datetime.datetime.now()
889 print(x) # (2021-07-29 12:00:00.000000)
890
891 # Python Math
892 # Importing the math Module
893 import math
894 print(math.pi) # 3.141592653589793

```

```
895 # Math Functions
896 # abs() - Returns the absolute value of a number
897 # ceil() - Returns the smallest integer not less than the given number
898 # floor() - Returns the largest integer not greater than the given number
899 # log() - Returns the natural logarithm of a number
900 # max() - Returns the largest number in a list
901 # min() - Returns the smallest number in a list
902 # pow() - Returns the value of x to the power of y
903
904 # Python JSON
905 # Importing the json Module
906 import json
907 # JSON Data
908 data = {'name': 'John', 'age': 30, 'city': 'New York' }
909 # Converting Python to JSON
910 json_data = json.dumps(data)
911 print(json_data) # {"name": "John", "age": 30, "city": "New York"}
912 # Converting JSON to Python
913 python_data = json.loads(json_data)
914 print(python_data) # {'name': 'John', 'age': 30, 'city': 'New York'}
915
916 # Python RegEx
917 # Importing the re Module
918 import re
919 # RegEx Pattern
920 # ^ matches the start of a string
921 # $ matches the end of a string
922 # . matches any character
923 # \d matches any digit
924 # \D matches any non-digit
925 # \s matches any whitespace
926 # \S matches any non-whitespace
927 # \w matches any alphanumeric character
928 # \W matches any non-alphanumeric character
929 # [abc] matches 'a' or 'b' or 'c'
930 # [a-z] matches any lowercase letter
931 # [A-Z] matches any uppercase letter
932 # [a-zA-Z] matches any letter
933 # [0-9] matches any digit
934 # [^abc] matches any character except 'a' or 'b' or 'c'
935
936 # Python XML
937 # Importing the xml.etree.ElementTree Module
938 import xml.etree.ElementTree as ET
939 # Parsing an XML File
940 # root = ET.parse('example.xml').getroot()
941 # root = ET.fromstring(xml_string)
942 # root = ET.ElementTree(xml_string)
943 # root = ET.fromstring(xml_string)
944
945 # Python PIP
946 # Installing a Package
947 # pip install package_name
948 # Uninstalling a Package
```

```
949 # pip uninstall package_name
950 # Listing Installed Packages
951 # pip list
952 # Using a Requirements File
953 # pip install -r requirements.txt
954 # Creating a Requirements File
955 # pip freeze > requirements.txt
956
957 # Python Try...Except
958 try:
959     # Code to be executed
960     print(x)
961 except:
962     # Code to be executed if an error occurs
963     print("An exception occurred")
964
965 # Python User Input
966 username = input("Enter username:")
967 print("Username is: " + username) # (Username is: John)
968
969 # Python String Formatting
970 price = 49
971 txt = "The price is {} dollars"
972 print(txt.format(price)) # (The price is 49 dollars)
973
974 # Python File Handling
975 # Opening a File
976 f = open("demofile.txt", "r")
977 print(f.read()) # (Hello, World!)
978
979 # Closing a File
980 f.close()
981
982 # Python Read Files
983 # Reading a File
984 f = open("demofile.txt", "r")
985 print(f.read()) # (Hello, World!)
986
987 # Reading Only Parts of a File
988 f = open("demofile.txt", "r")
989 print(f.read(5)) # (Hello)
990
991 # Reading Lines
992 f = open("demofile.txt", "r")
993 print(f.readline()) # (Hello, World!)
994
995 # Reading Multiple Lines
996 f = open("demofile.txt", "r")
997 print(f.readline()) # (Hello, World!)
998
999 # Python Write/Create Files
1000 # Writing to an Existing File
1001 f = open("demofile.txt", "a")
1002 f.write("Now the file has more content!")
```

```
1003
1004 # Closing the File
1005 f.close()
1006
1007 # Python Delete Files
1008 # Deleting a File
1009 import os
1010 # Delete a file
1011 os.remove("demofile.txt")
1012 # Check if file exists
1013 try:
1014     os.remove("demofile.txt")
1015 except:
1016     print("The file does not exist")
1017
1018
1019
```