Questions sheet 1

on (python basics, control flows and different datatypes)

Created by: Eng. Maryam Mostafa

Basic questions:

1. What is the output of the following code?

```
valueOne = 5 ** 2
valueTwo = 5 ** 3
print(valueOne)
print(valueTwo)
```

- 10
  15
- **25**
  125
- Error: invalid syntax

Explanation: Using two multiplication symbols, we can make a power relationship in Python. We call `**` operator an exponent operator. For example, the result of expression `5 ** 3` is 125.

2. What is the Output of the following code?

```
for x in range(0.5, 5.5, 0.5):
  print(x)
```

- [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5]
- [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]
- **The Program executed with errors**

Explanation: We cannot use float numbers in `range()` function. Please refer to How to generate a range of float numbers.

3. What is the output of the following code?

```
sampleList = ["Jon", "Kelly", "Jessa"]
sampleList.append(2, "Scott")
print(sampleList)
```

- **The program executed with errors**
- ['Jon', 'Kelly', 'Scott', 'Jessa']
- ['Jon', 'Kelly', 'Jessa', 'Scott']
- ['Jon', 'Scott', 'Kelly', 'Jessa']

Explanation: The `append()` method appends an item to the end of the list. Therefore, we cannot pass the index number to it.

4. What is the output of the following code?

```
sampleSet = {"Jodi", "Eric", "Garry"}
sampleSet.add(1, "Vicki")
print(sampleSet)
```

- {'Vicki', 'Jodi', 'Garry', 'Eric'}
- {'Jodi', 'Vicki', 'Garry', 'Eric'}
- **The program executed with error**

Explanation: The set is an unordered data structure. Therefore, we cannot access/add/remove its elements by index number.

5. What is the output of the following code?

```
var1 = 1
var2 = 2
var3 = "3"
print(var1 + var2 + var3)
```

- 6
- 33
- 123
- **Error. Mixing operators between numbers and strings are not supported**

Explanation

We cannot add strings and numbers together using the `+` operator. Either we can use the `+` operator to concatenate strings or add numbers.

6. A string is immutable in Python?

Every time when we modify the string, Python Always create a new String and assign a new string to that variable.

- **True**
- False

Explanation: Yes, strings are immutable in Python. You cannot modify a string once created. If you change a string, Python builds a new string with the updated value and assigns it to the variable.

7. What is the output of the following code?

```
var= "James Bond"
print(var[2::-1])
```

- Jam
- dno
- **maJ**
- dnoB semaJ

Explanation: Pick a range of items starting in the reverse direction starting from index 2 with step 1.

8. What is the output of the following code?

```
for i in range(10, 15, 1):
  print( i, end=', ')
```

- **10, 11, 12, 13, 14,**
- 10, 11, 12, 13, 14, 15,

Explanation: Remember, the range doesn't include the stop number in the output.
Read Python range function for more details.

9. The `in` operator is used to check if a value exists within an iterable object container such as a list. Evaluate to `True` if it finds a variable in the specified sequence and `False` otherwise.

- **True**
- False

10. What is the output of the following code?

```
str = "pynative"
print (str[1:3])
```

- py
- **yn**
- pyn
- yna

11. What is the output of the following

```
x = 36 / 4 * (3 + 2) * 4 + 2
print(x)
```

- **182.0**
- 37
- 117
- The Program executed with errors

Explanation: To choose the correct answer, You must know the operator precedence and associativity.

12. What is the output of the following code?

```
var = "James" * 2  * 3
print(var)
```

- **JamesJamesJamesJamesJamesJames**
- JamesJamesJamesJamesJames
- Error: invalid syntax

Explanation: We can use `*` operator to repeat the string `n` number of times. For example, in the above question, First, we repeated the string two times, and again we repeated the output string three times.

Control flow (operators):

1. What is the output of the following addition (`+`) operator

```
a = [10, 20]
b = a
b += [30, 40]
print(a)
print(b)
```

- **[10, 20, 30, 40]**
  [10, 20, 30, 40]
- [10, 20]
  [10, 20, 30, 40]

Explanation: Because both `b` and `a` refer to the same object, when we use addition assignment `+=` on `b`, it changes both `a` and `b`

2. What is the output of the expression `print(-18 // 4)`

- -4
- 4
- **-5**
- 5

Explanation: In the case of **floor division** operator (`//`), when the result is negative, the result is rounded down to the next smallest (big negative) integer.

3. What is the output of the following code

```
print(bool(0), bool(3.14159), bool(-3), bool(1.0+1j))
```

- True True False True
- **False True True True**
- True True False True
- False True False True

Explanation:

- If we pass A zero value to `bool()` constructor, it will treat it as a boolean `False`.
- Any non-zero value will be treated as a boolean `True`.

4. What is the value of the following Python Expression `print(36 / 4)`

- **9.0**
- 9

Explanation: Remember the result of a **division operator(`/`)**, is always float value.

5. What is the output of the following Python code

```
x = 10
y = 50
if x ** 2 > 100 and y < 100:
    print(x, y)
```

- 100 500
- 10 50
- **None**

6. What is the output of the following assignment operator

```
y = 10
x = y += 2
print(x)
```

- 12
- 10
- **SynatxError**

Explanation: `x = y += 2` expression is Invalid

7. What is the output of the following code

```
x = 100
y = 50
print(x and y)
```

- True
- 100
- False
- **50**

Explanation: In Python, when we join two non-Boolean values using a `and` operator, the value of the expression is the second operands, not `True` or `False`.

8. What is the output of `print(2 * 3 ** 3 * 4)`

- **216**
- 864

Explanation: The exponent (`**`) operator has higher precedence than multiplication (`*`). Therefore the statement `print(2 * 3 ** 3 * 4)` evaluates to `print(2 * 27 * 4)`

9. What is the data type of `print(type(10))`

- float
- integer
- **int**

10. What is the result of `print(type([]) is list)`

- False
- **True**

11. What is the output of the following variable assignment?

```
X = x + 1
print(x)
```

- **Error**
- 76
- 1
- None

Control flow (if statements and loops):

1. Given the nested `if-else` below, what will be the value `x` when the code executed successfully

```
x = 0
a = 5
b = 5
if a > 0:
    if b < 0:
        x = x + 5
    elif a > 5:
        x = x + 4
    else:
        x = x + 3
else:
    x = x + 2
print(x)
```

- 0
- 4
- 2
- **3**

2. What is the output of the following `if` statement

```
a, b = 12, 5
if a + b:
    print('True')
else:
  print('False')
```

- False
- **True**

Explanation: In Python, any non-zero value is considered **TRUE**. So it will evaluate to true

3. Select which is true for `for` loop

- **Python's for loop used to iterates over the items of list, tuple, dictionary, set, or string**
- **else clause of for loop is executed when the loop terminates naturally**
- else clause of for loop is executed when the loop terminates abruptly
- We use for loop when we want to perform a task indefinitely until a particular condition is met

Explanation: We use while loop when we want to perform a task indefinitely until a particular condition is true.

4. What is the output of the following nested loop

```
numbers = [10, 20]
items = ["Chair", "Table"]
for x in numbers:
  for y in items:
    print(x, y)
```

- **10 Chair**
  10 Table
  20 Chair
  20 Table
- 10 Chair
  10 Table

5. What is the value of the `var` after the `for` loop completes its execution

```
var = 10
for i in range(10):
    for j in range(2, 10, 1):
        if var % 2 == 0:
            continue
            var += 1
    var+=1
print(var)
```

- **20**
- 21
- 10
- 30

Explanation:

- The `continue` statement returns the control to the beginning of the loop
- else block of a `for` loop is executed when the loop terminates naturally

6. What is the value of `x` after the following nested for loop completes its execution

```
x = 0
for i in range(10):
  for j in range(-1, -10, -1):
    x += 1
print(x)
```

- 99
- **90**
- 100

7. What is the output of the following loop

```
for l in 'Jhon':
   if l == 'o':
      pass
   print(l, end=", ")
```

- J, h, n,
- **J, h, o, n,**

Explanation: in Python, the **`pass`** is a null operation. The Python interpreter executes the `pass` statement without any activity. The `pass` statement is useful when you want to write the pseudo code that you want to implement in the future.

8. What is the output of the following nested loop?

```
for num in range(10, 14):
   for i in range(2, num):
       if num%i == 1:
          print(num)
          break
```

- **10**
  11
  12
  13
- 11
  13

Explanation: We use a `break` statement to terminate the loop and transfer execution to the statement immediately following the loop.

Tuple datatype:

1. A Python tuple can also be created without using parentheses

- False
- **True**

Explanation: A tuple can also be created without using parentheses. It is called tuple packing.

2. What is the output of the following tuple operation

```
aTuple = (100,)
print(aTuple * 2)
```

- TypeError
- **(100, 100)**
- (200)

Explanation: We can use `*` operator to repeat the tuple values `n` number of times.

3. What is the output of the following

```
aTuple = (10, 20, 30, 40, 50, 60, 70, 80)
print(aTuple[2:5], aTuple[:4], aTuple[3:])
```

- **(30, 40, 50) (10, 20, 30, 40) (40, 50, 60, 70, 80)**
- (20, 30, 40, 50) (10, 20, 30, 40) (30, 40, 50, 60, 70, 80)

Explanation:

To get a sub tuple out of the tuple, we need to specify the range of indexes. We need to specify where to start and where to end the range.

Syntax: `tuple[start:end]` If the start is missing it takes 0 as the starting index.

4. What is the output of the following tuple operation

```
aTuple = (100, 200, 300, 400, 500)
aTuple.pop(2)
print(aTuple)
```

- (100, 200, 400, 500)
- (100, 300, 400, 500)
- **AttributeError**

Explanation:

A tuple is immutable. Once a tuple is created, you cannot remove its items, but you can delete the tuple completely. If you try to remove the item from the tuple, you will receive an `AttributeError: 'tuple' object has no attribute 'pop'`.

5. Select true statements regarding the Python tuple

- We can remove the item from tuple but we cannot update items of the tuple
- We cannot delete the tuple
- **We cannot remove the items from the tuple**
- **We cannot update items of the tuple.**

Explanation: A tuple is immutable.

6. Select which is true for Python tuple

- **A tuple maintains the order of items**
- A tuple is unordered
- **We cannot change the tuple once created**
- We can change the tuple once created

7. What is the output of the following

```
aTuple = "Yellow", 20, "Red"
a, b, c = aTuple
print(a)
```

- ('Yellow', 20, 'Red')
- TyepeError
- **Yellow**

Explanation: The tuple unpacking is also possible

8. What is the output of the following

```
tuple1 = (1120, 'a')
print(max(tuple1))
```

- **TypeError**
- 1120
- 'a'

Explanation: TypeError: `'>'` not supported between instances of 'str' and 'int'

9. What is the type of the following variable

```
aTuple = ("Orange")
print(type(aTuple))
```

- list
- tuple
- array
- **str**

Explanation: To create a tuple with a single item, you need to add a comma after the item. Otherwise, Python will not recognize the variable as a tuple, and it will treat it as a string type.

Set datatype:

1. What is the output of the following

```
set1 = {10, 20, 30, 40, 50}
set2 = {60, 70, 10, 30, 40, 80, 20, 50}
print(set1.issubset(set2))
print(set2.issuperset(set1))
```

- False
  False
- **True**
  True

2. What is the output of the following set operation.

```
set1 = {"Yellow", "Orange", "Black"}
set2 = {"Orange", "Blue", "Pink"}
set1.difference_update(set2)
print(set1)
```

- **{'Black', 'Yellow'}**
- {'Yellow', 'Orange', 'Black', 'Blue', 'Pink'}

Explanation: The `difference_update()` method removes the items that exist in both sets. Here `set1.difference_update(set2)` removed the unwanted items from the original set1.

3. Select all the correct ways to copy two sets

- **`set2 = set1.copy()`**
- **`set2 = set(set1)`**
- `set2.update(set1)`
- `set2 = set1`

Explanation:

When you set `set2= set11`, you are making them refer to the same object, so when you modify one of them, all references associated with that object reflect the current state of the object. So don't use the assignment operator to copy the set instead use the `copy()` method or `set()` constructor.

4. What is the output of the following code

```
aSet = {1, 'PYnative', ('abc', 'xyz'), True}
print(aSet)
```

- TypeError
- {'PYnative', 1, ('abc', 'xyz'), True}
- **{'PYnative', 1, ('abc', 'xyz')}**

Explanation: Set already has **1** as item **`True` evaluates to 1**. As you know set doesn't allow duplicate element

5. The `union()` method returns a new set with all items from both sets by removing duplicates

- **True**
- False

6. What is the output of the following

```
sampleSet = {"Yellow", "Orange", "Black"}
sampleSet.discard("Blue")
print(sampleSet)
```

- **{'Yellow', 'Orange', 'Black'}**
- KeyError: 'Blue'

Explanation:

If the item to remove does not exist in the set, the `discard()` method will **NOT** raise an error. If we use `remove()` method to perform the same operation, we will receive a `keyError`.

List datatype:

1. What is the output of the following code?

```
sampleList = [10, 20, 30, 40]
del sampleList[0:6]
print(sampleList)
```

- **[]**
- list index out of range.
- [10, 20]

2. What is the output of the following list function?

```
sampleList = [10, 20, 30, 40, 50]
sampleList.pop()
print(sampleList)
sampleList.pop(2)
print(sampleList)
```

- [20, 30, 40, 50]
  [10, 20, 40]
- [10, 20, 30, 40]
  [10, 20, 30, 50]
- **[10, 20, 30, 40]**
  [10, 20, 40]

Explanation: The list's `pop()` function is used to remove the item present at the specified index, (or the last item if the index is not specified).

3. What is the output of the following list function?

```
sampleList = [10, 20, 30, 40, 50]
sampleList.append(60)
print(sampleList)
sampleList.append(60)
print(sampleList)
```

- [10, 20, 30, 40, 50, 60]
  [10, 20, 30, 40, 50, 60]
- **[10, 20, 30, 40, 50, 60]**
  [10, 20, 30, 40, 50, 60, 60]

Explanation: The `append()` method is used to add an item at the end of a list. Also, the list allows duplicate items.

4. What is the output of the following

```
aList = [5, 10, 15, 25]
print(aList[::-2])
```

- [15, 10, 5]
- [10, 5]
- **[25, 10]**

Explanation: `aList[::-2]` Start from the end of the list with step value 2.

5. Select all the correct options to copy a list `aList = ['a', 'b', 'c', 'd']`

- `newList = copy(aList)`
- **`newList = aList.copy()`**
- `newList.copy(aList)`
- **`newList = list(aList)`**

Explanation: The `copy()` method and `list()` constructor can be used to create a copy of a list. This will create a new list and any changes made in the original list will not reflect in the new list. This is **shallow copying**.

6. In Python, `list` is mutable

- False
- **True**

Explanation: The list collection is ordered and changeable. A mutable object can be changed after it is created. So we can update or remove elements from a `list` once it is created.

7. What is the output of the following

```
l = [None] * 10
print(len(l))
```

- **10**
- 0
- Syntax Error

8. What is the output of the following list assignment

```
aList = [4, 8, 12, 16]
aList[1:4] = [20, 24, 28]
print(aList)
```

- [4, 20, 24, 28, 8, 12, 16]
- **[4, 20, 24, 28]**

Explanation: Use the assignment operator (=) to replace an item or a range of items in a List.

9. What is the output of the following code

```
list1 = ['xyz', 'zara', 'PYnative']
print (max(list1))
```

- PYnative
- **zara**

10. What is the output of the following code

```
my_list = ["Hello", "Python"]
print("-".join(my_list))
```

- HelloPython-
- **Hello-Python**
- -HelloPython

Explanation: The `join()` method will join all items in a list into a string, using a hyphen character as a separator.

11. Select all the correct options to join two lists in Python

```
listOne  =  ['a', 'b', 'c', 'd']
listTwo =  ['e', 'f', 'g']
```

- **`newList = listOne + listTwo`**
- `newList = extend(listOne, listTwo)`
- **`newList = listOne.extend(listTwo)`**
- `newList.extend(listOne, listTwo)`

Explanation:

1. The `extend()` method adds all the elements of an iterable (list, tuple, string) to the end of the list.
2. You can also use the addition operator to join two list in Python

12. What is the output of the following code

```
aList = ["PYnative", [4, 8, 12, 16]]
print(aList[0][1])
print(aList[1][3])
```

- P 8
  Y 16
- P
  12
- **Y**
  16

13. What is the output of the following list operation

```
aList = [10, 20, 30, 40, 50, 60, 70, 80]
print(aList[2:5])
print(aList[:4])
print(aList[3:])
```

- [20, 30, 40, 50]
  [10, 20, 30, 40]
  [30, 40, 50, 60, 70, 80]
- **[30, 40, 50]**
  [10, 20, 30, 40]
  [40, 50, 60, 70, 80]

Explanation:
Python list collection is ordered and changeable. The list also allows duplicate members. To get a sublist out of the list, we need to specify the range of indexes. To get a sublist, we need to specify where to start and where to end the range.

Syntax: `list[start:end]` If start is missing it takes `0` as the starting index


Best
WISHES
FOR
You