# Cheat Sheet

### Types (kinds of data that can be used by your programs)

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Examples** |
| int | integers | 42, -3, 92851 |
| float | real numbers | 3.14, 2.0 |
| string | text | "date", "Cat" |

### Expressions (compute a value using arithmetic operations)

|  |  |
| --- | --- |
| **Arithmetic Operators** | |
| **Operator** | **Meaning** |
| + | addition |
| - | subtraction, negation |
| \* | multiplication |
| / | division |
| // | integer division |
| % | remainder ("modulus") |
| \*\* | exponent |

* *precedence*: () before \*/% before +-
* // is integer division and % is integer remainder

|  |  |
| --- | --- |
| **1 \* 2** + 3 \* 5 // 4  2 + **3 \* 5** // 4  2 + **15 // 4**  **2 + 3**  5 | **9.0 / 4.0** + 1  **2.25** + 1  **2.26** |

### Printing numbers (convert numbers to concatenate them with strings)

print(**string** + str(**number**) + **string**)

print(The answer is ” + str(42)) # The answer is 42

### Variables (pieces of memory that can store a value of a particular type)

name = value; *assignment (stores a value into a variable)*

y = 3

x = 1 + y \* 2 # x stores the value 7

### The for loop (repeats a group of statements a fixed number of times)

for name in range(start, stop + 1):

statement

statement

...

statement

for i in range(1, 11):

print(str(i) + " squared is " + str(i \* i))

### Parameters (A way to pass information in to a function)

Declaration:

Call:

function\_name(expression, ..., expression)

Example:

**line(10)** # \*\*\*\*\*\*\*\*\*\*

**line(5)** # \*\*\*\*\*

def name(name, ..., name):

statements

Example:

def line(**width**):

print("\*" \* **width**)

### Class constants (unchangeable global values that can be seen throughout your program)

NAME = value DAYS\_PER\_WEEK = 7  
**Problems**

### Expressions

1. Compute the value of each expression below.  
Be sure to list a literal of appropriate type (e.g., 7.0 rather than 7 for a float, string literals in quotes).

|  |  |
| --- | --- |
| Expression | Expression |
| 4 \* 3//8 + 2.5 \* 2  26 % 10 % 4 \* 3  (5 \* 7.0//2 - 2.5)//5 \* 2  12//7 \* 4.4 \* 2//4  41 % 7 \* 3//5 + 5/2 \* 2.5  22 + 4 \* 2  10.0/2/4  23 % 8 % 3  17 % 10//4  8//5 + 13//2/3.0  12 - 2 - 3  6/2 + 7//3  6 \* 7%4 | (2.5 + 3.5)//2  9//4 \* 2.0 - 5//4  3 \* 4 + 2 \* 3  177 % 100 % 10//2  9/2.0 + 7//3 - 3.0/2  813 % 100//3 + 2.4  27//2/2.0 \* (4.3 + 1.7) - 8/3  89 % (5 + 5) % 5  4.0/2 \* 9//2  392//10 % 10//2  53//5//(0.6 + 1.4)/2 + 13//2  8 \* 2 - 7//4  37 % 20 % 3 \* 4  2.5 \* 2 + 8/5.0 + 10//3  2 \* 3//4 \* 2/4.0 + 4.5 - 1  89 % 10//4 \* 2.0/5 + (1.5 + 1.0/2) \* 2 |

### Variables

|  |  |
| --- | --- |
| 2. What is the output from the following code?  min = 10  max = 17 - 4 // 10  max = max + 6  min = max - min  print(max \* 2)  print(max + min)  print(max)  print(min) | 3. What are the values of a, b, and c after the following code? (What is the code really doing?)  a = 2  b = 3  c = 4  a = a \* b \* c  b = a // b // c  c = a // b // c  a = a // b // c |

### Constants

4. Assume that you have a constant called SIZE that will take on the values 1, 2, 3, 4, and so on. You are going to formulate expressions in terms of SIZE that will yield different sequences. The first row is filled in for you as an example. Fill in the table below, indicating an expression that will generate each sequence.

|  |  |  |
| --- | --- | --- |
| SIZE **value** | **Sequence** | **Expression** |
| 1, 2, 3, 4, 5, 6 | 2, 4, 6, 8, 10, 12, ... | 2 \* SIZE |
| 1, 2, 3, 4, 5, 6, ... | 4, 19, 34, 49, 64, 79, ... |  |
| 1, 2, 3, 4, 5, 6, ... | 30, 20, 10, 0, -10, -20, ... |  |
| 1, 2, 3, 4, 5, 6, ... | -7, -3, 1, 5, 9, 13, ... |  |

# Problems (continued)

### Parameters

5. Write a function called print\_by that takes a number as a parameter and prints all of the numbers between 1 and 100 counting by that number. For example print\_by(10) would produce the following:

10 20 30 40 50 60 70 80 90 100

6. Write a main program that calls print\_by from the above question with all numbers from 1 – 50.

7. Write a function called repeat that takes a string as a parameter and outputs that string two times. For example, repeat(“meow”) would output meowmeow and repeat(“meowmeow”) would output meowmeowmeowmeow

8. Write a main program that calls repeat.

9. Add a second parameter to the repeat function from question 7. This parameter should represent the number of times the function should repeat the first parameter.

# Solutions

1.

Expression Value Expression Value

4 \* 3//8 + 2.5 \* 2 6.0 (2.5 + 3.5)//2 3.0

26 % 10 % 4 \* 3 6 9//4 \* 2.0 - 5//4 3.0

(5 \* 7.0//2 - 2.5)//5 \* 2 4.0 3 \* 4 + 2 \* 3 18

12//7 \* 4.4 \* 2//4 2.0 177 % 100 % 10//2 3

41 % 7 \* 3//5 + 5/2 \* 2.5 9.25 9/2.0 + 7//3 - 3.0/2 5.0

22 + 4 \* 2 30 813 % 100//3 + 2.4 6.4

10.0/2/4 1.25 27//2/2.0 \* (4.3 + 1.7) - 8//3 37.0

23 % 8 % 3 1 89 % (5 + 5) % 5 4

17 % 10//4 1 4.0/2 \* 9//2 9.0

8//5 + 13//2/3.0 3.0 392//10 % 10//2 4

12 - 2 - 3 7 53//5//(0.6 + 1.4)/2 + 13//2 8.5

6/2 + 7//3 5.0 8 \* 2 - 7//4 15

6 \* 7%4 2 37 % 20 % 3 \* 4 8

2.5 \* 2 + 8/5.0 + 10//3 9.6

2 \* 3//4 \* 2/4.0 + 4.5 - 1 4.0

89 % 10//4 \* 2.0/5 + (1.5 + 1.0/2) \* 2 4.8

2.

46

36

23

13

3.

a has value 4

b has value 2

c has value 3 (The code is rotating the values of the variables.)

4.

|  |  |  |
| --- | --- | --- |
| SIZE **value** | **Sequence** | **Expression** |
| 1, 2, 3, 4, 5, 6, ... | 2, 4, 6, 8, 10, 12, ... | 2 \* SIZE |
| 1, 2, 3, 4, 5, 6, ... | 4, 19, 34, 49, 64, 79, ... | 15 \* SIZE - 11 |
| 1, 2, 3, 4, 5, 6, ... | 30, 20, 10, 0, -10, -20, ... | -10 \* SIZE + 40 |
| 1, 2, 3, 4, 5, 6, ... | -7, -3, 1, 5, 9, 13, ... | 4 \* SIZE - 11 |

5.

def print\_by(n):

for i in range(n, 100, n):

print(i, " ", end="")

print()

9.

def repeat(s, count):

print(s \* count)

def main():

repeat("meow", 4)

repeat("meowmeow", 2)

6.

def main():

for i in range(1, 51):

print\_by(i)

7.

def repeat(s):

print(s \* 2)

8.

def main():

repeat("meow")

repeat("meowmeow")