✖

Quiz 1

10 questions

1 point

1.
An `if` statement can have how many `elif` parts?

- ○ Unlimited, i.e., 0 or more
- ○ 1
- ○ 0

1 point

2.
Consider the Boolean expression `not (p or not q)`. Give the four following values in order, separated only by spaces:

the value of the expression when `p` is `True`, and `q` is `True`,

the value of the expression when `p` is `True`, and `q` is `False`,

the value of the expression when `p` is `False`, and `q` is `True`,

the value of the expression when `p` is `False`, and `q` is `False`,

Remember, each of the four results you provide should be `True` or `False` with the proper capitalization.

Enter answer here

1
point

3.
A common error for beginning programmers is to confuse the behavior of `print` statements and `return` statements.

- `print` statements can appear anywhere in your program and print a specified value(s) in the console. Note that execution of your Python program continues onward to the following statement. Remember that executing a `print` statement inside a function definition does not return a value from the function.
- `return` statements appear inside functions. The value associated with the `return` statement is substituted for the expression that called the function. Note that executing a `return` statement terminates execution of the function definition immediately. Any statements in the function definition following the `return` statement are ignored. Execution of your Python code resumes with the execution of the statement after the function call.

As an example to illustrate these points, consider the following piece of code:

```
def do_stuff():
    print "Hello world"
    return "Is it over yet?"
    print "Goodbye cruel world!"

print do_stuff()
```

Note that this code calls the function `do_stuff` in the last `print` statement. The definition of `do_stuff` includes two `print` statements and one `return` statement.

Which of the following is the console output that results from executing this piece of code? While it is trivial to solve this question by cutting and pasting this code into CodeSkulptor, we suggest that you first attempt this problem by attempting to execute this code in your mind.

○
```
Hello world
Is it over yet?
```

○
```
Hello world
```

○
```
Hello world
Is it over yet?
Goodbye cruel world!
Is it over yet?
```

○
```
Hello world
Is it over yet?
Goodbye cruel world!
```

1
point

4.
Given a non-negative integer `n`, which of the following expressions computes the ten's digit of `n`? For example, if `n` is 123, then we want the expression to evaluate to 2.

Think about each expression mathematically, but also try each in CodeSkulptor.

☐ (`n` % 10) / 10

☐ (`n` // 10) % 10

☐ (`n` % 100 - `n` % 10) / 10

1
point

5.

The function calls `random.randint(0, 10)` and

The function calls `random.randint(0, 10)` and `random.randrange(0, 10)` generate random numbers in different ranges. What number can be generated by one of these functions, but not the other?

(Refer to the CodeSkulptor documentation.)

By the way, we (and most Python programmers) always prefer to use `random.randrange()` since it handles numerical ranges in a way that is more consistent with the rest of Python.

Enter answer here

1
point

6.
Implement the mathematical function $f(x) = -5x^5 + 69x^2 - 47$ as a Python function. Then use Python to compute the function values $f(0)$, $f(1)$, $f(2)$, and $f(3)$. Enter the maximum of these four values calculated.

Enter answer here

1
point

7.

When investing money, an important concept to know is compound

When investing money, an important concept to know is compound interest.

The equation $FV = PV(1 + rate)^{periods}$ relates the following four quantities.

- The *present value* (*PV*) of your money is how much money you have now.
- The *future value* (*FV*) of your money is how much money you will have in the future.
- The *nominal interest rate per period* (*rate*) is how much interest you earn during a particular length of time, **before** accounting for compounding. This is typically expressed as a percentage.
- The *number of periods* (*periods*) is how many periods in the future this calculation is for.

Finish the following code, run it, and submit the printed number. Provide at least four digits of precision after the decimal point.

```
def future_value(present_value, annual_rate, periods_per_ye
ar, years):
    rate_per_period = annual_rate / periods_per_year
    periods = periods_per_year * years

    # Put your code here.

print "$1000 at 2% compounded daily for 3 years yields $",
future_value(1000, .02, 365, 3)
```

Before submitting your answer, test your function on the following example.

`future_value(500, .04, 10, 10)` should return 745.317442824

Enter answer here

1
point

8.

There are several ways to calculate the area of a regular polygon. Given the number of sides, n, and the length of each side, s, the polygon's area is

$$\frac{ns^2}{4\tan(\frac{\pi}{n})}$$

For example, a regular polygon with 5 sides, each of length 7 inches, has area 84.3033926289 square inches.

Write a function that calculates the area of a regular polygon, given the number of sides and length of each side. Submit the area of a regular polygon with 7 sides each of length 3 inches. Enter a number (and not the units) with at least four digits of precision after the decimal point.

Note that the use of inches as the unit of measurement in these examples is arbitrary. Python only keeps track of the numerical values, not the units.

Enter answer here

1
point

9.

Running the following program results in the error

`SyntaxError: bad input on line 8 ('return').`

Which of the following describes the problem?

```
def max_of_2(a, b):
    if a > b:
        return a
    else:
        return b

def max_of_3(a, b, c):
return max_of_2(a, max_of_2(b, c))
```

○ Extra parenthesis

○ Misspelled keyword

- ○ Misspelled variable name
- ○ Misspelled function name
- ○ Incorrect indentation
- ○ Wrong number of arguments in function call
- ○ Missing colon
- ○ Missing parenthesis

1
point

10.
The following code has a number of syntactic errors in it. The intended math calculations are correct, so the only errors are syntactic. Fix the syntactic errors.

Once the code has been fully corrected, it should print out two numbers. The first should be 1.09888451159. Submit the **second** number printed in CodeSkulptor. Provide at least four digits of precision after the decimal point.

```
define project_to_distance(point_x point_y distance):
    dist_to_origin = math.square_root(pointx ** 2 + pointy
** 2)
     scale == distance / dist_to_origin
    print point_x * scale, point_y * scale

project-to-distance(2, 7, 4)
```

Enter answer here

---

9 questions unanswered

Upgrade to submit